WML

wireless markup language

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

WML is an XML language used to specify content and user interface for WAP devices like PDA and Mobile Phones. The WAP forum provides a DTD for WML.

This tutorial explains how to use WML to develop WAP applications.

Audience

This tutorial is designed for Software Professionals who are in the need of learning the basics of WML.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of XML, text editor, execution of programs, etc.

Disclaimer & Copyright

Table of Contents

1. WML - OVERVIEW

The topmost layer in the WAP (Wireless Application Protocol) architecture is made up of WAE (Wireless Application Environment), which consists of WML and WML scripting language.

- WML stands for **W**ireless **M**arkup **L**anguage
- WML is an application of XML, which is defined in a document-type definition.
- WML is based on HDML and is modified so that it can be compared with HTML.
- WML takes care of the small screen and the low bandwidth of transmission.
- WML is the markup language defined in the WAP specification.
- WAP sites are written in WML, while web sites are written in HTML.
- WML is very similar to HTML. Both of them use tags and are written in plain text format.
- WML files have the extension ".wml". The MIME type of WML is "text/vnd.wap.wml".
- WML supports client-side scripting. The scripting language supported is called WML Script.

WML Versions

WAP Forum has released a latest version WAP 2.0. The markup language defined in WAP 2.0 is XHTML Mobile Profile (MP). The WML MP is a subset of the XHTML. A style sheet called **WCSS** (WAP CSS) has been introduced along with XHTML MP. The WCSS is a subset of the CSS2.

Most of the new mobile phone models released are WAP 2.0-enabled. Because WAP 2.0 is backward compatible to WAP 1.x, WAP 2.0-enabled mobile devices can display both XHTML MP and WML documents.

WML 1.x is an old technology. However, that does not mean it is of no use, since a lot of wireless devices that support only WML 1.x are still being used. The latest version of WML is 2.0 and it is backward compatible. So, WAP site developers need not worry about WML 2.0.

WML Decks and Cards

The main difference between HTML and WML is that the basic unit of navigation in HTML is a page, while that in WML is a card. A WML file can contain multiple cards and they form a deck.

When a WML page is accessed from a mobile phone, all the cards in the page are downloaded from the WAP server. So, if the user goes to another card of the same deck, the mobile browser does not have to send any requests to the server since the file that contains the deck is already stored in the wireless device.

You can put links, text, images, input fields, option boxes, and many other elements in a card.

WML Program Structure

Following is the basic structure of a WML program:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="one" title="First Card">
<p>
This is the first card in the deck
</p>
</card>

<card id="two" title="Second Card">
<p>
Ths is the second card in the deck
</p>
</card>

</wml>
```

The first line of this text says that this is an XML document and the version is 1.0. The second line selects the document type and gives the URL of the document type definition (DTD).

One WML deck (i.e. page) can have one or more cards as shown above. We will see complete details on WML document structure in subsequent chapter.

Unlike HTML 4.01 Transitional, text cannot be enclosed directly in the <card>...</card> tag pair. So, you need to put a content inside ... as shown above.

WAP Site Design Considerations

Wireless devices are limited by the size of their displays and keypads. It's therefore very important to take this into account when designing a WAP Site.

While designing a WAP site, you must ensure that you keep things simple and easy to use. You should always keep in mind that there are no standard micro browser behaviors and that the data link may be relatively slow, at around 10Kbps. However, with GPRS, EDGE, and UMTS, this may not be the case for long, depending on where you are located.

The following are general design tips that you should keep in mind when designing a service:

- Keep the WML decks and images to less than 1.5KB.
- Keep the text brief and meaningful, and as far as possible try to pre-code options to minimize the rather painful experience of user data entry.
- Keep the URLs **brief** and easy to recall.
- Minimize menu levels to prevent users from getting lost and the system from slowing down.
- Use standard layout tags such as <big> and <b>, and logically structure your information.
- Don't go overboard with the use of graphics, as many target devices may not support them.

2. WML - ENVIRONMENT

To develop WAP applications, you will need the following:

- **A WAP enabled Web Server:** You can enable your Apache or Microsoft IIS to serve all the WAP client request.

- **A WAP Gateway Simulator:** This is required to interact to your WAP server.

- **A WAP Phone Simulator:** This is required to test your WAP Pages and to show all the WAP pages.

You can write your WAP pages using the following languages:

- Wireless Markup Language (WML) to develop WAP application.
- WML Script to enhance the functionality of WAP application.

Configuring Web Server

In normal web applications, MIME type is set to text/html, designating normal HTML code. Images, on the other hand, could be specified as image/gif or image/jpeg. With this content type specification, the web browser knows the data type that the web server returns.

To make your Apache WAP compatible, you don't have to put a lot of effort. All that you need to do is to add support for the MIME types and extensions listed below.

File Extension	MIME type
WML (.wml)	text/vnd.wap.wml
WMLScript (.wmls)	text/vmd.wap.wmlscript
WMLScriptc (.wmlsx)	application/vnd.wap.wmlscriptc
WMLC (.wmlc)	application/vnd.wap.wmlc
WBMP (.wbmp)	image/vnd.wap.wbmp

Configure Apache Web Server for WAP

Let us assume you have Apache Web server installed on your machine. Now we will explain let you know tell you how to enable WAP functionality in your Apache web server.

Locate Apache's file httpd.conf which is usually in /etc/httpd/conf, and add the following lines to the file and restart the server:

```
AddType text/vnd.wap.wml .wml
AddType text/vnd.wap.wmlscript .wmls
AddType application/vnd.wap.wmlc .wmlc
AddType application/vnd.wap.wmlscriptc .wmlsc
AddType image/vnd.wap.wbmp .wbmp
```

In dynamic applications, the MIME type must be set on the fly, whereas in static WAP applications, the web server must be configured appropriately.

Configure Microsoft IIS for WAP

To configure a Microsoft IIS server to deliver WAP content, you need to perform the following exercise:

- Open the Internet Service Manager console and expand the tree to view your Website entry. You can add the WAP MIME types to the whole server or to the individual directories.

- Open the Properties dialog box by right-clicking the appropriate server or directory, then choose Properties from the menu.

- From the Properties dialog, choose the HTTP Headers tab, then select the File Types button at the bottom right.

- For each MIME type listed in the above table, supply the extension with or without the dot (it will be automatically added for you), then click OK in the Properties dialog box to accept your changes.

Installing WAP Gateway Simulator

There are many WAP Gateway Simulator available on the Internet, so download any of them and install on your PC. You would need to run this gateway before starting WAP Mobile simulator.

WAP Gateway will take your request and passes it to the Web Server and whatever response will be received from the Web server that will be passed to the Mobile Simulator.

You can download it from Nokia website:

- **Nokia WAP Gateway simulator** - Download Nokia WAP Gateway simulator.

Installing WAP Phone Simulator

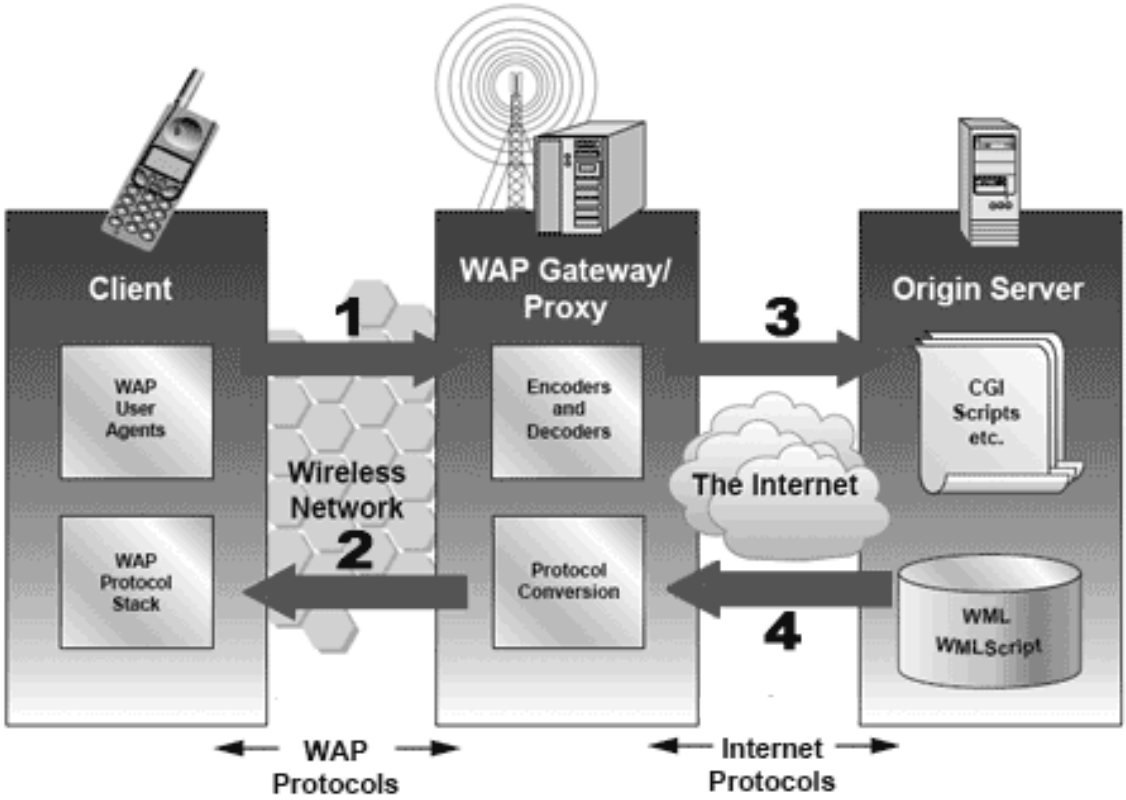
There are many WAP Simulators available on the Internet, so download any of them and install on your PC, which you will use as a WAP client. Here are popular links to download simulator:

- **Nokia WAP simulator** - Download Nokia WAP simulator.
- **WinWAP simulator** - Download WinWAP browser from their official website.

NOTE: If you have WAP enabled phone, then you do not need to install this simulator. But while doing development, it is more convenient and economic to use a simulator.

The WAP Model

The following figure shows the WAP programming model. Note the similarities with the Internet model. Without the WAP Gateway/Proxy, the two models would have been practically identical.



1 -- WSP Request (URL)
2 -- WSP Response (Binary WML)
3 -- HTTP Request (URL)
3 -- HTTP Response (WML)

WAP Gateway/Proxy is the entity that connects the wireless domain with the Internet. You should make a note that the request, which is sent from the wireless client to the WAP Gateway/Proxy uses the Wireless Session Protocol (WSP). In its essence, WSP is a binary version of HTTP.

A **markup language** - the Wireless Markup Language (WML) has been adapted to develop optimized WAP applications. In order to save valuable bandwidth in the wireless network, WML can be encoded into a compact binary format. Encoding WML is one of the tasks performed by the WAP Gateway/Proxy.

How WAP Model Works?

When it comes to actual use, WAP works like this:

- The user selects an option on their mobile device, which has a URL with Wireless Markup language (WML) content assigned to it.
- The phone sends the URL request via the phone network to a WAP gateway, using the binary encoded WAP protocol.
- The gateway translates this WAP request into a conventional HTTP request for the specified URL, and sends it on to the Internet.
- The appropriate Web server picks up the HTTP request.
- The server processes the request, just as it would any other request. If the URL refers to a static WML file, the server delivers it. If a CGI script is requested, it is processed and the content returned as usual.
- The Web server adds the HTTP header to the WML content and returns it to the gateway.
- The WAP gateway compiles the WML into binary form.
- The gateway then sends the WML response back to the phone.
- The phone receives the WML via the WAP protocol.
- The micro-browser processes the WML and displays the content on the screen.

3. WML - SYNTAX

A WML program is typically divided into two parts: the document **prolog** and the **body**. Consider the following code:

Following is the basic structure of a WML program:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="one" title="First Card">
<p>
This is the first card in the deck

</p>
</card>

<card id="two" title="Second Card">
<p>
Ths is the second card in the deck
</p>
</card>

</wml>
```

WML Document Prolog

The first line of this text says that this is an XML document and the version is 1.0. The second line selects the document type and gives the URL of the document type definition (DTD). The DTD referenced is defined in WAP 1.2, but this header changes with the versions of the WML. The header must be copied exactly so that the tool kits automatically generate this prolog.

The prolog components are not WML elements and they should not be closed, i.e. you should not give them an end tag or finish them with />.

WML Document Body

The body is enclosed within a <wml> </wml> tag pair. The body of a WML document can consist of one or more of the following:

- Deck
- Card
- Content to be shown
- Navigation instructions

Unlike HTML 4.01 Transitional, text cannot be enclosed directly in the <card>...</card> tag pair. So you need to put a content inside ... as shown above.

Testing Your Program

Put the above code in a file called test.wml file, and put this WML file locally on your hard disk, then view it using an emulator.

This is by far the most efficient way of developing and testing WML files. Since your aim is, however, to develop a service that is going to be available to WAP phone users, you should upload your WML files onto a server once you have developed them locally and test them over a real Internet connection. As you start developing more complex WAP services, this is how you will identify and rectify performance problems, which could, if left alone, lose your site visitors.

In uploading the file test.wml to a server, you will be testing your WML emulator to see how it looks and behaves, and checking your Web server to see that it is set up correctly. Now start your emulator and use it to access the URL of test.wml. For example, the URL might look something like this:

```
http://websitename.com/wapstuff/test.wml
```

NOTE: Before accessing any URL, make sure WAP Gateway Simulator is running on your PC.

When you will download your WAP program, then you will see only first card at your mobile. Following is the output of the above example on Nokia Mobile Browser 4.0. This mobile supports horizontal scrolling. You can see the text off the screen by pressing the "Left" or "Right" button.

When you press right button, then second card will be visible as follows:

4. WML - ELEMENTS

WML is defined by a set of *elements* that specify all markup and structural information for a WML deck. Elements are identified by tags, which are each enclosed in a pair of angle brackets.

Unlike HTML, WML strictly adheres to the XML hierarchical structure, and thus, elements must contain a start tag; any content such as text and/or other elements; and an end tag. Elements have one of the following two structures:

- **<tag> content </tag>** : This form is identical to HTML.
- **<tag />:** This is used when an element cannot contain visible content or is empty, such as a line break. WML document's prolog part does not have any element, which has closing element.

Following table lists the majority of valid elements. A complete detail of all these elements is given in **WML Tags Reference**.

Deck & Card Elements

WML Elements	Purpose
<!-->	Defines a WML comment
<wml>	Defines a WML deck (WML root)
<head>	Defines head information
<meta>	Defines meta information
<card>	Defines a card in a deck
<access>	Defines information about the access control of a deck
<template>	Defines a code template for all the cards in a deck

Text Elements

WML Elements	Purpose
** **	Defines a line break
<p>	Defines a paragraph
<table>	Defines a table
<td>	Defines a table cell (table data)

<tr>	Defines a table row
<pre>	Defines preformatted text

Text Formatting Tags

WML Elements	Purpose
<b>	Defines bold text
<big>	Defines big text
****	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
****	Defines strong text
<u>	Defines underlined text

Image Elements

WML Elements	Purpose
<img>	Defines an image

Anchor Elements

WML Elements	Purpose
<a>	Defines an anchor
<anchor>	Defines an anchor

Event Elements

WML Elements	Purpose
<do>	Defines a do event handler
<onevent>	Defines an onevent event handler
<postfield>	Defines a postfield event handler
<ontimer>	Defines an ontimer event handler
<onenterforward>	Defines an onenterforward handler

<onenterbackward>	Defines an onenterbackward handler
<onpick>	Defines an onpick event handler

Task Elements

WML Elements	Purpose
<go>	Represents the action of switching to a new card
<noop>	Says that nothing should be done
<prev>	Represents the action of going back to the previous card
<refresh>	Refreshes some specified card variables.

Input Elements

WML Elements	Purpose
<input>	Defines an input field
<select>	Defines a select group
<option>	Defines an option in a selectable list
<fieldset>	Defines a set of input fields
<optgroup>	Defines an option group in a selectable list

Variable Elements

WML Elements	Purpose
<setvar>	Defines and sets a variable
<timer>	Defines a timer

5. WML - COMMENTS

As with most programming languages, WML also provides a means of placing comment text within the code.

Comments are used by developers as a means of documenting programming decisions within the code to allow for easier code maintenance.

WML comments use the same format as HTML comments and use the following syntax:

```
<!-- This will be assumed as a comment -->
```

A multiline comment can be given as follows:

```
<!-- This is a multi-line
     comment -->
```

The WML author can use comments anywhere, and they are not displayed to the user by the user agent. Some emulators may complain if comments are placed before the XML prolog.

Note that comments are not compiled or sent to the user agent, and thus have no effect on the size of the compiled deck.

6. WML - VARIABLES

Because multiple cards can be contained within one deck, some mechanism needs to be in place to hold data as the user traverses from card to card. This mechanism is provided via WML variables.

WML is case sensitive. No case folding is performed when parsing a WML deck. All enumerated attribute values are case sensitive. For example, the following attribute values are all different: id="Card1", id="card1", and id="CARD1".

Variables can be created and set using several different methods. Following are the two examples:

The <setvar> element

The <setvar> element is used as a result of the user executing some task. The >setvar> element can be used to set a variable's state within the following elements: <go>, <prev>, and <refresh>.

This element supports the following attributes:

Attribute	Value	Description
name	string	Sets the name of the variable
value	string	Sets the value of the variable
class	class data	Sets the class name for the element.
id	element ID	A unique ID for the element.

The following element would create a variable named *'a'* with a value of 1000:

```
<setvar name="a" value="1000"/>
```

The input elements

Variables are also set through any input element like *input, select, option,* etc. A variable is automatically created that corresponds with the named attribute of an input element.

For example, the following element would create a variable named *b*:

```
<select name="b">
<option value="value1">Option 1</option>
```

```
<option value="value2">Option 2</option>
</select>
```

Using Variables

Variable expansion occurs at runtime, in the micro browser or emulator. This means, it can be concatenated with or embedded in other text.

Variables are referenced with a preceding dollar sign, and any single dollar sign in your WML deck is interpreted as a variable reference.

```
<p> Selected o
```

7. WML – FORMATTING

This section describes basic text formatting elements of WML.

Line Break

The <br /> element defines a line break and almost all WAP browsers support a line break tag.

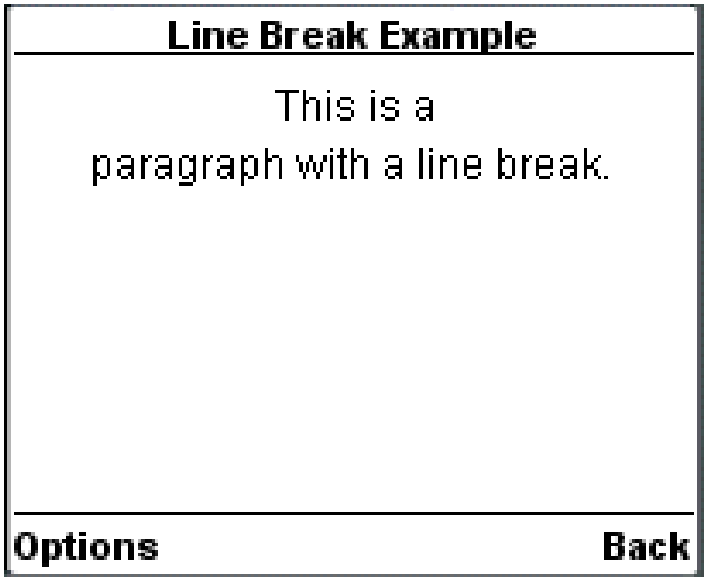
The <br /> element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <br /> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Line Break Example">
<p align="center">
This is a <br /> paragraph with a line break.
</p>
</card>

</wml>
```

It will produce the following result:



Text Paragraphs

The <p> element defines a paragraph of text and WAP browsers always render a paragraph in a new line.

A <p> element is required to define any text, image, or a table in WML.

The <p> element supports the following attributes:

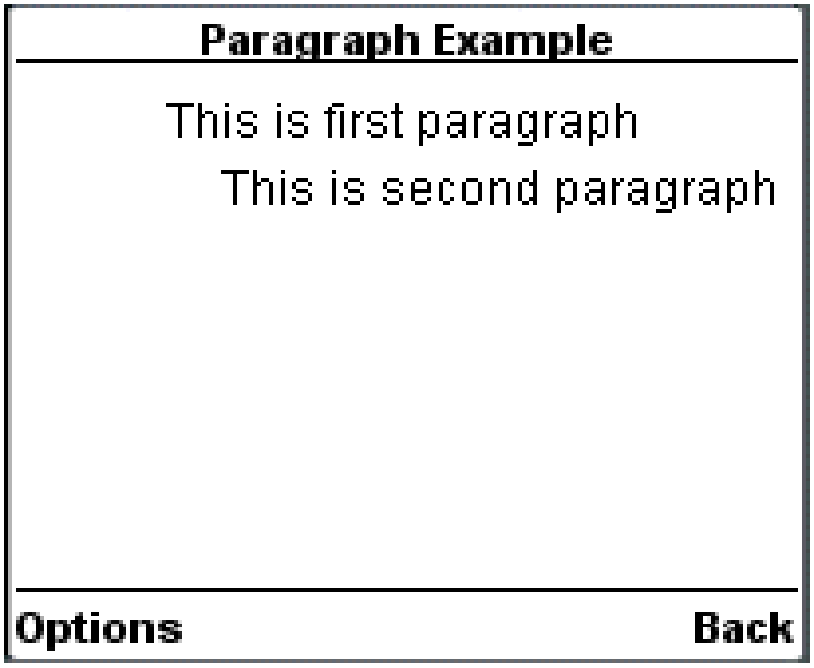
Attribute	Value	Description
align	left right center	This is used to change the horizontal alignment of a paragraph.
mode	wrap no wrap	Sets whether a paragraph should wrap lines or not.
xml:lang	language_code	Sets the language used in the element
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <p> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
```

```
<wml>

<card title="Paragraph Example">
<p align="center">
This is first  paragraph
</p>
<p align="right">
This is second  paragraph
</p>
</card>

</wml>
```

It will produce the following result:



WML Tables

The <table> element along with <tr> and <td> is used to create a table in WML. WML does not allow the nesting of tables.

A <table> element should be put with-in ... elements.

The <table /> element supports the following attributes:

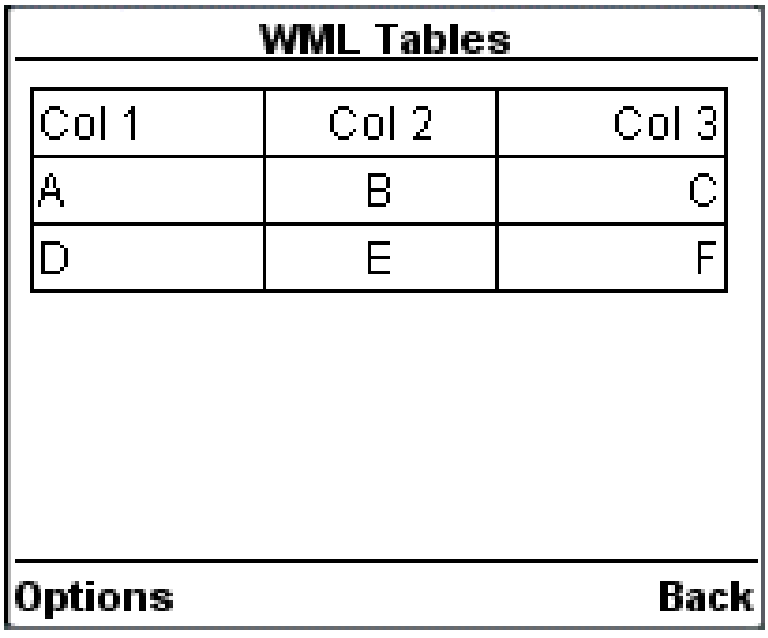
Attribute	Value	Description
columns	number	Sets the number of columns in the table
align	L C R	To specify the horizontal text alignment of the columns, you need to assign three letters to the align attribute. Each letter represents the horizontal text alignment of a column. The letter can be L, C, or R. For example, if you want the following settings to be applied to your table: First table column -- Left-aligned Second table column -- Center-aligned Third table column -- Right-aligned Then you should set the value of the *align* attribute to LCR.
xml:lang	language_code	Sets the language used in the element
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <table> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="WML Tables">
<p>
<table columns="3" align="LCR">
     <tr>
       <td>Col 1</td>
       <td>Col 2</td>
       <td>Col 3</td>
```

```
    </tr>

    <tr>
      <td>A</td>
      <td>B</td>
      <td>C</td>
    </tr>

    <tr>
      <td>D</td>
      <td>E</td>
      <td>F</td>
    </tr>
</table>
</p>
</card>

</wml>
```

It will produce the following result:



Preformatted Text

The <pre> element is used to specify preformatted text in WML. Preformatted text is text of which the format follows the way it is typed in the WML document.

This tag preserves all the white spaces enclosed inside this tag. Make sure you are not putting this tag inside ...

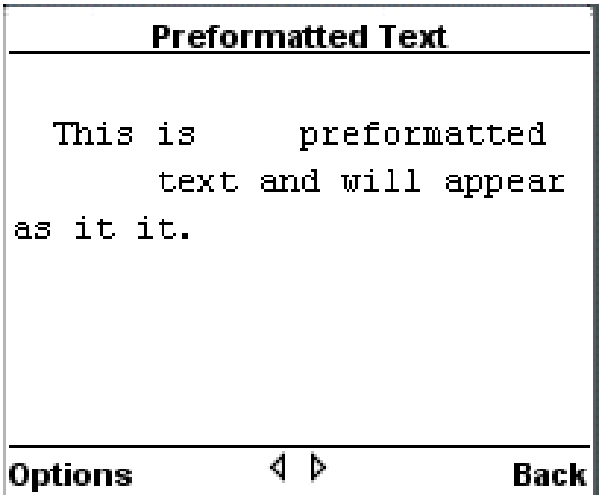
The <pre> element supports following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <pre> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Preformatted Text">
   <pre>
   This is     preformatted
        text and will appear
 as it it.
</pre>
</card>

</wml>
```

It will produce the following result:



8. WML – FONTS

WML does not support <font> element, but there are other WML elements, which you can use to create different font effects like underlined text, bold text, and italic text, etc.

These tags are given in the following table:

WML Elements	Purpose
<b>	Defines bold text
<big>	Defines big text
	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
	Defines strong text
<u>	Defines underlined text

These elements support the following attributes:

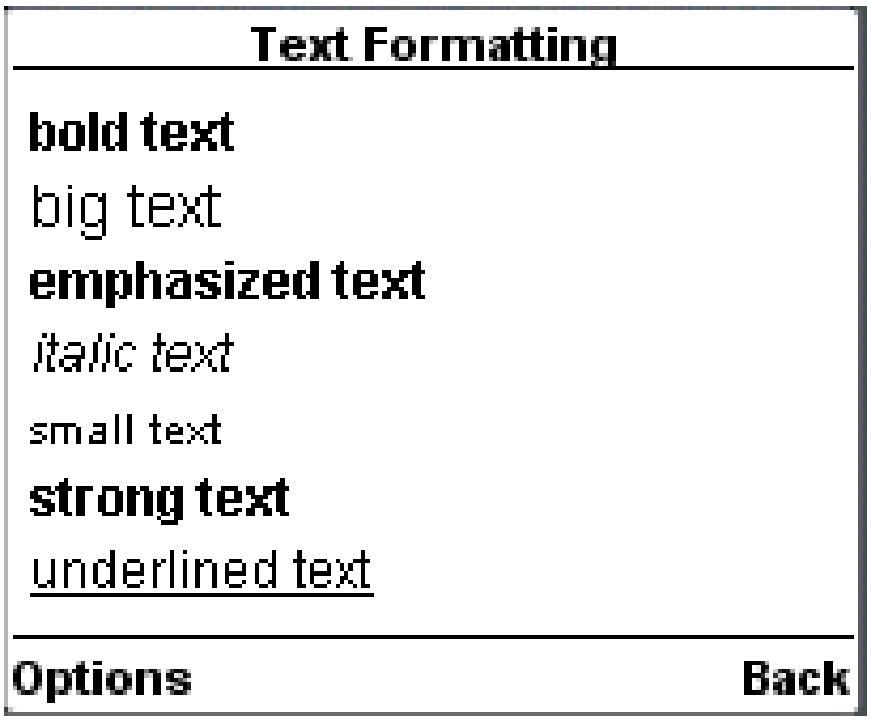
Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of these elements.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Text Formatting">
<p>
```

```
   <b>bold text</b><br/>
   <big>big text</big><br/>
   <em>emphasized text</em><br/>
   <i>italic text</i><br/>
   <small>small text</small><br/>
   <strong>strong text</strong><br/>
   <u>underlined text</u>
</p>
</card>

</wml>
```

It will produce the following result:



9. WML – IMAGES

The <img> element is used to include an image in a WAP card. WAP-enabled wireless devices only supported the Wireless Bitmap (WBMP) image format.

WBMP images can only contain two colors: black and white. The file extension of WBMP is ".wbmp" and the MIME type of WBMP is "image/vnd.wap.wbmp".

The <img> element supports the following attributes:

Attribute	Value	Description
align	top middle bottom	Alignment of the image.
alt	alternative text	Sets an alternate text to be displayed if the image is not displayed.
height	px %	Height of the image in pixels or percentage. If you specify the value in pixels, the syntax is "140", instead of "140px".
hspace	px %	Sets white space to the left and right of the image. If you specify the value in pixels, the syntax is "140", instead of "140px".
localsrc	cdata	Sets an alternate representation for the image. If this attribute is set, the browser will use it instead of the "src" attribute.
src	image url	A path to wbmp image.
vspace	px %	Sets white space above and below the image. If you specify the value in pixels, the syntax is "140", instead of "140px".
width	px %	Sets the width of the image. If you specify the value in pixels, the syntax is "140", instead of "140px".
xml:lang	language_code	Sets the language used in the element
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

How to make ".wbmp" Images

There are free tools available on the Internet to make ".wbmp" images.

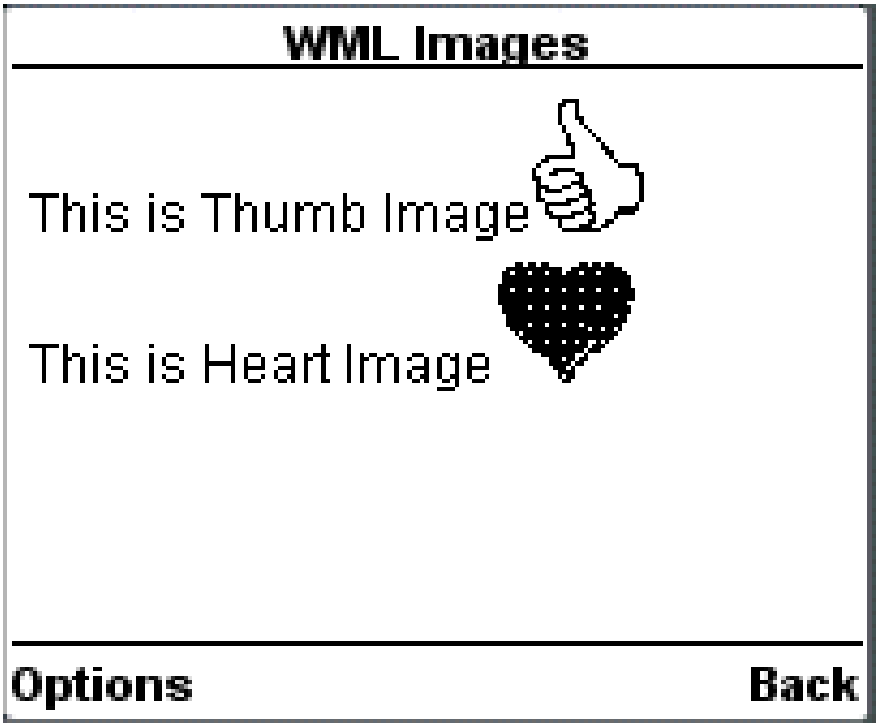
The Nokia Mobile Internet Toolkit (NMIT) comes with a WBMP image editor that you can use. You can convert existing GIF or JPG image files into WBMP file using NMIT.

Another free tool is **ImageMagick**, which can help you to create WBMP images.

The following example shows the usage of <img> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="WML Images">
<p>
This is Thumb image
<img src="/images/thumb.wbmp" alt="Thumb Image"/>
</p>

<p>
This is Heart image
<img src="/images/heart.wbmp" alt="Heart Image"/>
</p>

</card>

</wml>
```

It will produce the following result:



10. WML – TABLES

The <table> element along with <tr> and <td> is used to create a table in WML. WML does not allow the nesting of tables

A <table> element should be put with-in ... elements.

The <table /> element supports the following attributes:

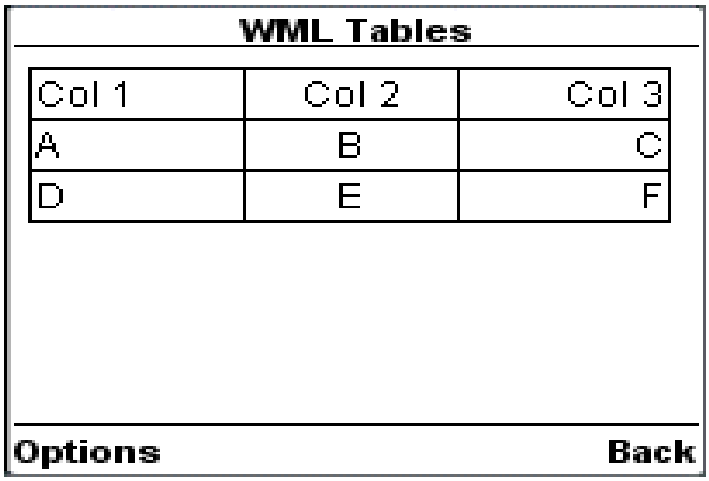
Attribute	Value	Description
columns	number	Sets the number of columns in the table
align	L C R	To specify the horizontal text alignment of the columns, you need to assign three letters to the align attribute. Each letter represents the horizontal text alignment of a column. The letter can be L, C, or R. For example, if you want the following settings to be applied to your table: First table column -- Left-aligned Second table column -- Center-aligned Third table column -- Right-aligned Then you should set the value of the *align* attribute to LCR.
xml:lang	language_code	Sets the language used in the element
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <table> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="WML Tables">
<p>
```

```
<table columns="3" align="LCR">
     <tr>
       <td>Col 1</td>
       <td>Col 2</td>
       <td>Col 3</td>
     </tr>

     <tr>
       <td>A</td>
       <td>B</td>
       <td>C</td>
     </tr>

     <tr>
       <td>D</td>
       <td>E</td>
       <td>F</td>
     </tr>
</table>
</p>
</card>

</wml>
```

It will produce the following result:



11. WML – LINKS

WML provides you an option to link various cards using links and then navigate through different cards.

There are two WML elements, <anchor> and <a>, which can be used to create WML links.

WML <anchor> Element

The <anchor>...</anchor> tag pair is used to create an anchor link. It is used together with other WML elements called <go/>, <refresh> or <prev/>. These elements are called task elements and tell WAP browsers what to do when a user selects the anchor link.

You can enclose Text or image along with a task tag inside <anchor>...</anchor> tag pair.

The <anchor> element supports the following attributes:

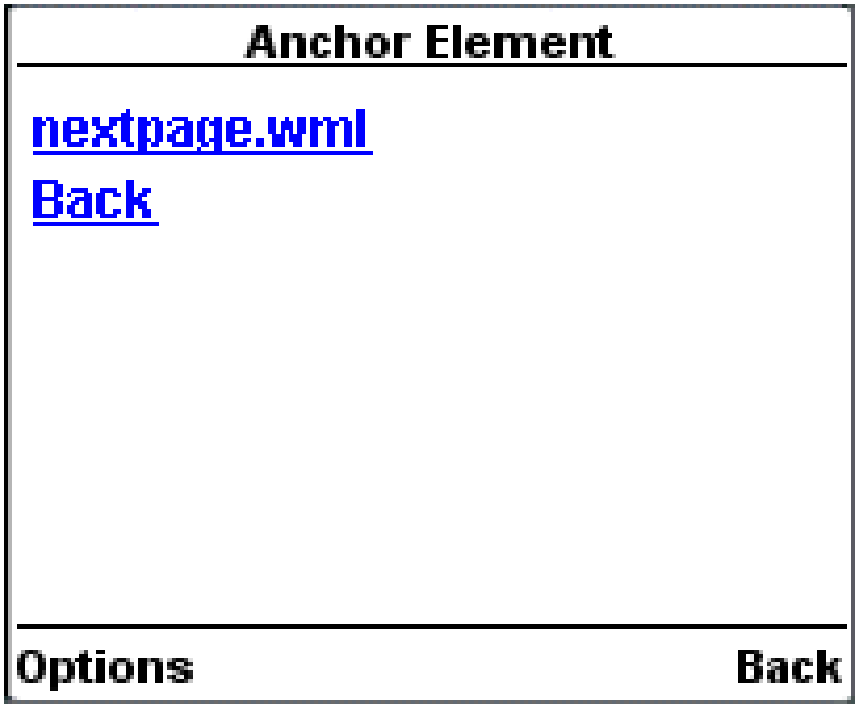
Attribute	Value	Description
title	cdata	Defines a text identifying the link
xml:lang	language_code	Sets the language used in the element
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <anchor> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Anchor Element">
<p>
   <anchor>
      <go href="nextpage.wml"/>
```

```
   </anchor>
</p>
<p>
   <anchor>
      <prev/>
   </anchor>
</p>
</card>

</wml>
```

It will produce the following result:



WML <a> Element

The <a>... tag pair can also be used to create an anchor link and always a preferred way of creating links.

You can enclose Text or image inside <a>... tags.

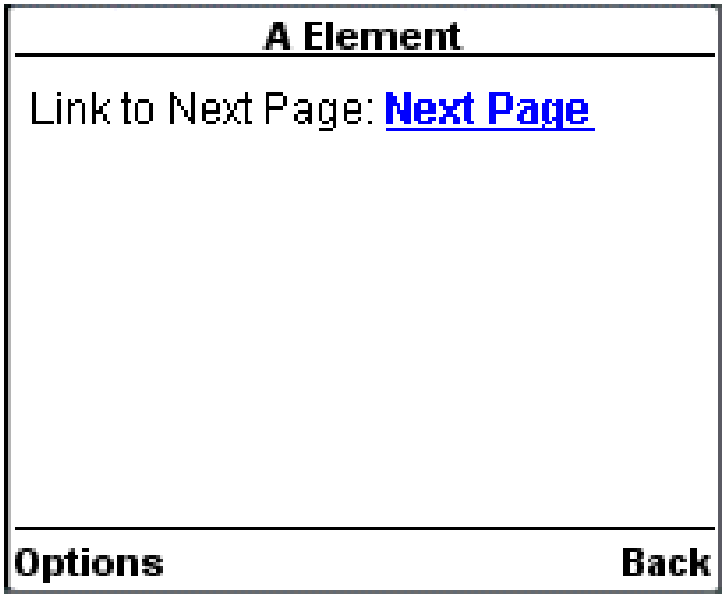
The <a> element supports the following attributes:

Attribute	Value	Description
href	URL	Defines URL of the liked page.
title	cdata	Defines a text identifying the link.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <a> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="A Element">
<p> Link to Next Page:
   <a href="nextpage.wml">Next Page</a>
</p>
</card>

</wml>
```

It will produce the following result:



12. WML – TASKS

A WML task is an element that specifies an action to be performed by the browser, rather than something to be displayed. For example, the action of changing to a new card is represented by a <go> task element, and the action of returning to the previous card visited is represented by a <prev> task element. Task elements encapsulate all the information required to perform the action.

WML provides the following four elements to handle four WML tasks called as:

- **The <go> task**
- **The <pre> task**
- **The <refresh> task**
- **The <noop> taks**

The <go> Task

As the name suggests, the <go> task represents the action of going to a new card.

The <go> element supports the following attributes:

Attribute	Value	Description
href	URL	Gives the URL of the new card. Relative URLs are resolved relative to the current card.
method	get post	Specifies the method that should be used to fetch the deck. This must be one of the values get or post, corresponding to the GET and POST methods of HTTP. When using method="get", the data is sent as a request with? Data appended to the url. The method has a disadvantage that it can be used only for a limited amount of data, and if you send sensitive information, it will be displayed on the screen and saved in the web server's logs. So, do not use this method if you are sending password etc. With method="post", the data is sent as a request with the data sent in the body of the request. This method has no limit, and sensitive information is not visible in the URL

sendreferer	true false	If set to true, the browser sends the URL of the current deck along with the request. This URL is sent as a relative URL if possible. The purpose of this, is to allow servers to perform simple access control on decks, based on which decks are linking to them. For example, using HTTP, this attribute is sent in the HTTP Referer header.
accept-charset	charset_list	Specifies a comma- or space-separated list of character sets that can encode data sent to the server in a POST request. The default value is "unknown".
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <go> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
   <anchor>
       Chapter 2 : <go href="chapter2.wml"/>
   </anchor>
</p>
</card>
</wml>
```

Another example showing how to upload data using Get Method:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
```

```
<wml>

<card title="GO Element">
<p>
   <anchor>
      Using Get Method
      <go href="chapter2.wml?x=17&y=42" method="get"/>
   </anchor>
</p>
</card>
</wml>
```

Another example showing how to upload data using <setvar> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
   <anchor>
      Using setvar:
       <go href="chapter2.wml">
           <setvar name="x" value="17"/>
           <setvar name="y" value="42"/>
       </go>
   </anchor>
</p>
</card>
</wml>
```

Another example showing how to upload data using <postfiled> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
   <anchor>
      Using setvar:
       <go href="chapter2.wml" method="get">
              <postfield name="x" value="17"/>
              <postfield name="y" value="42"/>
       </go>
   </anchor>
</p>
</card>
</wml>
```

The <prev> Task

The <prev> task represents the action of returning to the previously visited card on the history stack. When this action is performed, the top entry is removed from the history stack, and that card is displayed again, after any <setvar> variable assignments in the <prev> task have taken effect.

If no previous URL exists, specifying <prev> has no effect.

The <prev> element supports the following attributes:

Attribute	Value	Description
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <prev> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Prev Element">
<p>
   <anchor>
        Previous Page :<prev/>
   </anchor>
</p>
</card>
</wml>
```

One situation where it can be useful to include variables in a <prev> task is a login page, which prompts for a username and password. In some situations, you may want to clear out the password field when returning to the login card, forcing the user to reenter it. This can be done with a construct such as:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Prev Element">
<p>
   <anchor>
        <prev>
           <setvar name="password" value=""/>
        </prev>
   </anchor>
</p>
```

```
</card>
</wml>
```

The <refresh> Task

The <refresh> task is the simplest task that actually does something. Its effect is simply to perform the variable assignments specified by its <setvar> elements, then redisplay the current card with the new values. The <go> and <prev> tasks perform the same action just before displaying the new card.

The <refresh> task is most often used to perform some sort of "reset" action on the card.

The <refresh> element supports the following attributes:

Attribute	Value	Description
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <refresh> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Referesh Element">
<p>
   <anchor>
         Refresh this page:
          <go href="test.wml"/>
          <refresh>
            <setvar name="x" value="100"/>
          </refresh>
   </anchor>
</p>
```

```
</card>
</wml>
```

The <noop> Task

The purpose of the <noop> task is to do nothing (no operation).

The only real use for this task is in connection with templates.

The <noop> element supports the following attributes:

Attribute	Value	Description
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <noop> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Noop Element">
<p>
  <do type="prev" label="Back">
      <noop/>
  </do>
</p>
</card>
</wml>
```

13. WML - INPUTS

WML provides various options to let a user enter information through WAP application.

First of all, we are going to look at the different options for allowing the user to make straight choices between items. These are usually in the form of menus and submenus, allowing users to drill down to the exact data that they want.

WML <select> Element

The <select>...</select> WML elements are used to define a selection list and the <option>...</option> tags are used to define an item in a selection list. Items are presented as radio buttons in some WAP browsers. The <option>...</option> tag pair should be enclosed within the <select>...</select> tags.

This element supports the following attributes:

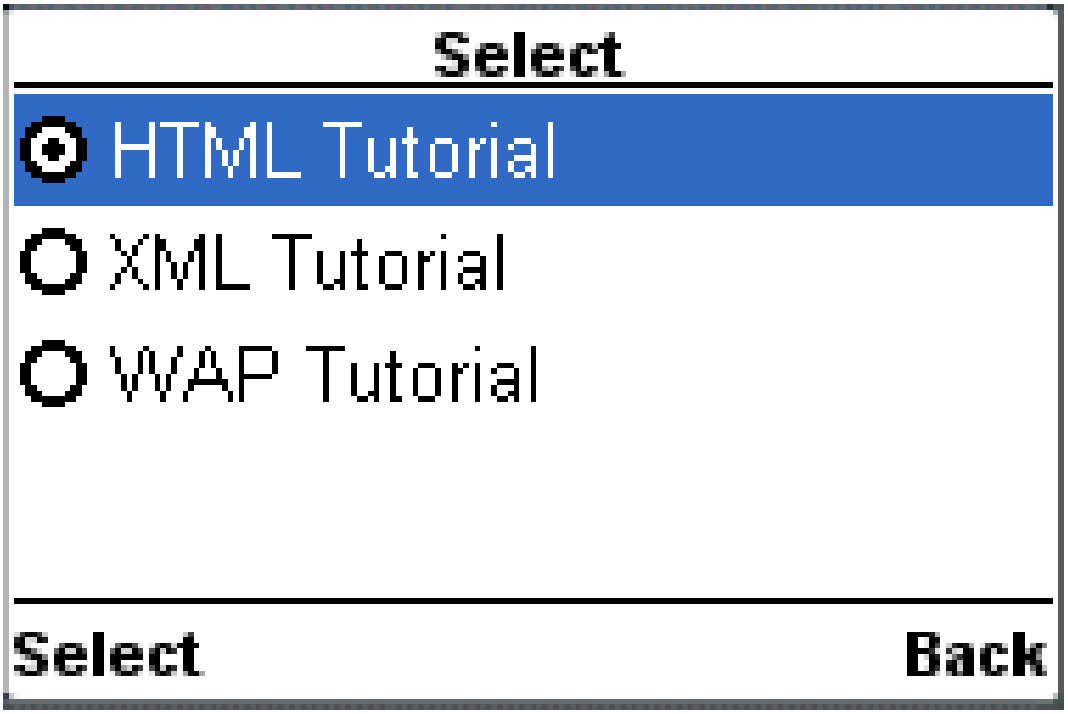
Attribute	Value	Description
iname	text	Names the variable that is set with the index result of the selection.
ivalue	text	Sets the pre-selected option element.
multiple	true false	Sets whether multiple items can be selected. Default is "false."
name	text	Names the variable that is set with the result of the selection.
tabindex	number	Sets the tabbing position for the select element.
title	text	Sets a title for the list.
value	text	Sets the default value of the variable in the "name" attribute.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of these two elements.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Selectable List">
<p> Select a Tutorial :
 <select>
  <option value="htm">HTML Tutorial</option>
  <option value="xml">XML Tutorial</option>
  <option value="wap">WAP Tutorial</option>
 </select>
</p>
</card>

</wml>
```

When you will load this program, it will show you the following screen:

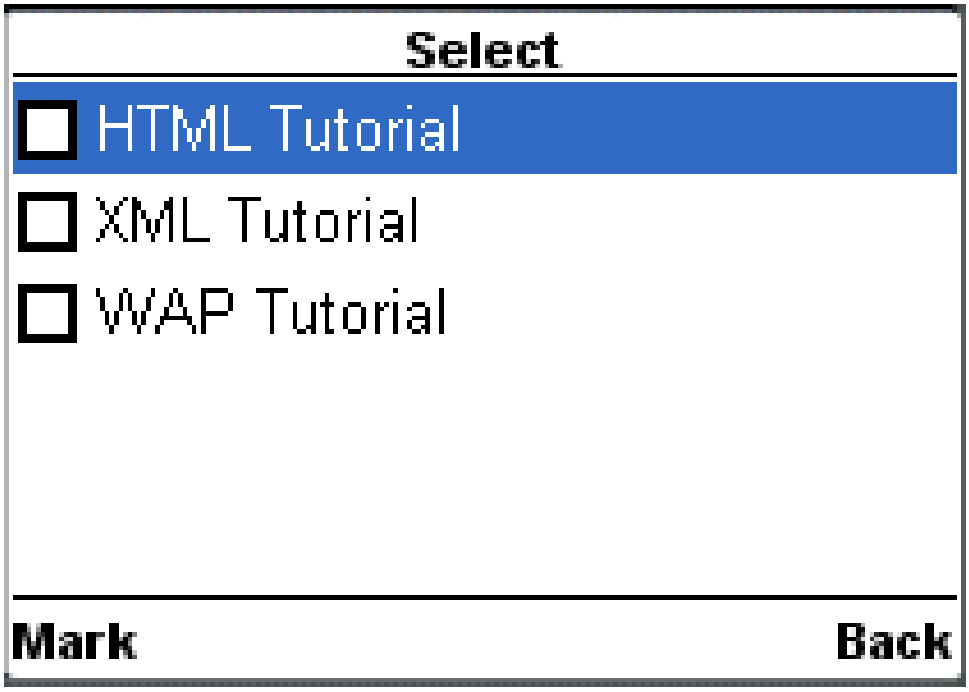
Once you highlight and enter the options, it will display the following screen:



You want to provide option to select multiple options, then set *multiple* attribute to *true* as follows:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Selectable List">
<p> Select a Tutorial :
 <select multiple="true">
  <option value="htm">HTML Tutorial</option>
  <option value="xml">XML Tutorial</option>
  <option value="wap">WAP Tutorial</option>
 </select>
</p>
</card>

</wml>
```

It will give you a screen to select multiple options as follows:



WML <input> Element

The <input/> element is used to create input fields and input fields are used to obtain alphanumeric data from users.

This element supports the following attributes:

Attribute	Value	Description
name	text	The name of the variable that is set with the result of the user's input.
maxlength	number	Sets the maximum number of characters the user can enter in the field.
emptyok	true false	Sets whether the user can leave the input field blank or not. Default is "false."
format	A a N X x M m *f *nf*	Sets the data format for the input field. Default is "*M." A = uppercase alphabetic or punctuation characters. a = lowercase alphabetic or punctuation characters. N = numeric characters. X = uppercase characters. x = lowercase characters. M = all characters.

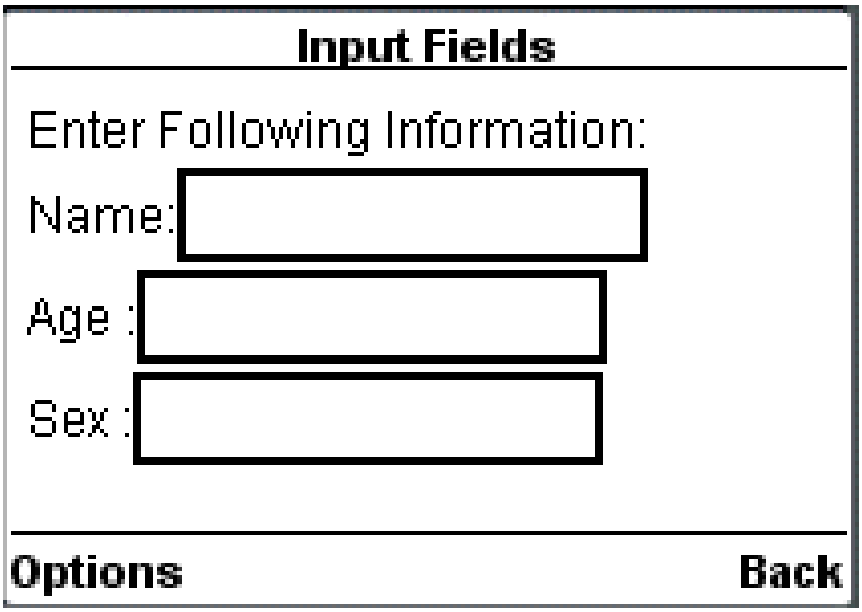
		m = all characters. *f* = Any number of characters. Replace the *f* with one of the letters above to specify what characters the user can enter. *nf* = Replace the *n* with a number from 1 to 9 to specify the number of characters, the user can enter. Replace the *f* with one of the letters above to specify what characters, the user can enter.
size	number	Sets the width of the input field.
tabindex	number	Sets the tabbing position for the select element.
title	text	Sets a title for the list.
type	text password	Indicates the type of the input field. The default value is "text". Password field is used to take password for authentication purpose.
value	text	Sets the default value of the variable in the "name" attribute.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Input Fields">
<p> Enter Following Information:<br/>
 Name: <input name="name" size="12"/>
 Age :  <input name="age" size="12" format="*N"/>
 Sex :  <input name="sex" size="12"/>
</p>
</card>
```

```
</wml>
```

This will provide you the following screen to enter required information:



WML <fieldset> Element

The <fieldset/> element is used to group various input fields or selectable lists.

This element supports the following attributes:

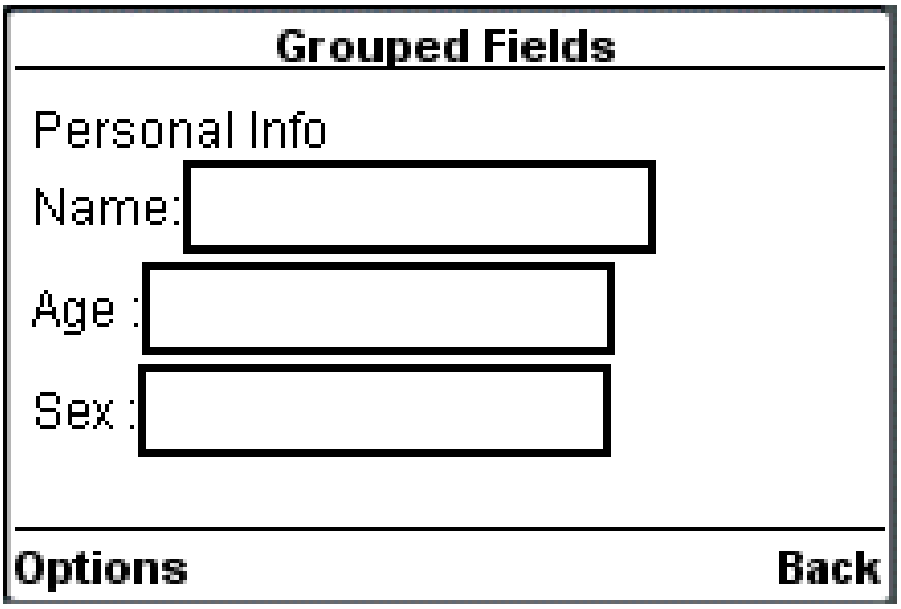
Attribute		Value	Description
title		text	Sets a title for the list.
xml:lang		language_code	Sets the language used in the element.
class		class data	Sets a class name for the element.
id		element ID	A unique ID for the element.

The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Grouped Fields">
```

```
<p>
<fieldset title="Personal Info">
 Name: <input name="name" size="12"/>
 Age :  <input name="age" size="12" format="*N"/>
 Sex :  <input name="sex" size="12"/>
</fieldset>
</p>
</card>


</wml>
```

It will provide you the following screen to enter required information. This result may differ browser to browser.



WML <optgroup> Element

The <optgroup/> element is used to group various options together inside a selectable list.

This element supports the following attributes:

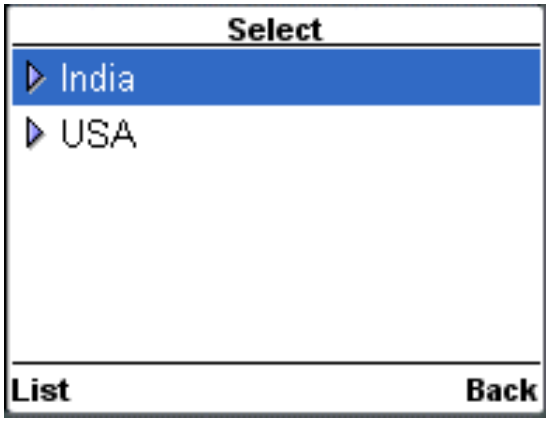
Attribute	Value	Description
title	text	Sets a title for the list.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.

id	element ID	A unique ID for the element.

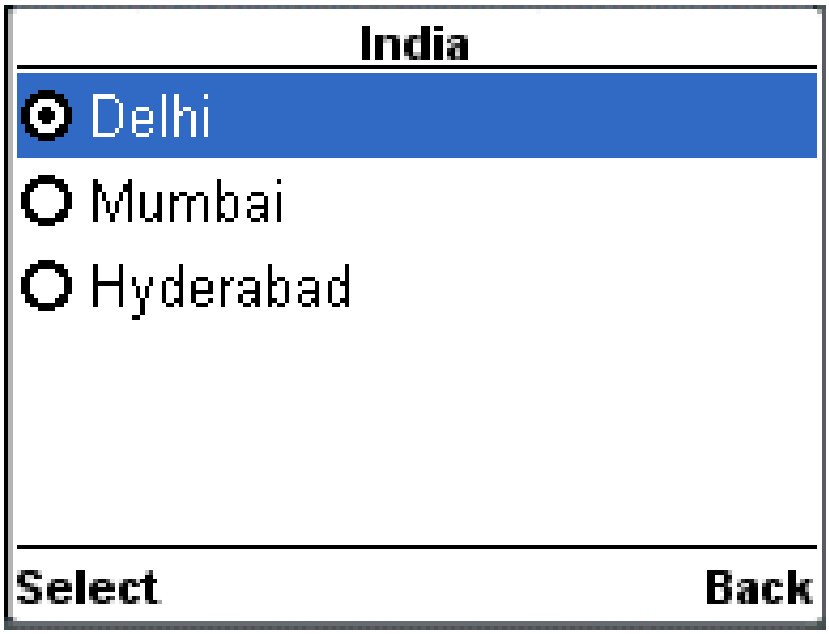
The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Selectable List">
<p>
 <select>
   <optgroup title="India">
    <option value="delhi">Delhi</option>
    <option value="mumbai">Mumbai</option>
    <option value="hyderabad">Hyderabad</option>
   </optgroup>
   <optgroup title="USA">
    <option value="ohio">Ohio</option>
    <option value="maryland">Maryland</option>
    <option value="washington">Washingtone</option>
   </optgroup>
 </select>
</p>
</card>

</wml>
```

When a user loads above code, then it will give two options to be selected:



When a user selects any of the options, then only it will give final options to be selected. So, if user selects India, then it will show you following options to be selected:



14. WML – SUBMIT DATA TO SERVER

Many times, you will want your users to submit some data to your server. Similar to *HTML Form* WML also provide a mechanism to submit user data to web server.

To submit data to the server in WML, you need the <go>...</go> along with <postfield/> tags. The <postfield/> tag should be enclosed in the <go>...</go> tag pair.

To submit data to a server, we collect all the set WML variables and use <postfield> elements to send them to the server. The <go>...</go> elements are used to set posting method to either POST or GET and to specify a server side script to handle uploaded data.

In the previous chapters, we have explained various ways of taking inputs form the users. These input elements sets WML variables to the entered values. We also know how to take values from WML variables. So, now following example shows how to submit three fields *name, age,* and *sex* to the server.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="card1" title="WML Form">
<p>
   Name: <input name="name" size="12"/>
   Sex : <select name="sex">
      <option value="male">Male</option>
      <option value="female">Female</option>
      </select>
   Age :  <input name="age" size="12" format="*N"/>
   <anchor>
      <go method="get" href="process.php">
          <postfield name="name" value="$(name)"/>
          <postfield name="age" value="$(age)"/>
          <postfield name="sex" value="$(sex)"/>
```

```
      </go>
      Submit Data
    </anchor>
</p>
</card>


</wml>
```

When you download above code on your WAP device, it will provide you option to enter three fields *name, age,* and *sex* and one link *Submit Data*. You will enter three fields and then finally you will select *Submit Data* link to send entered data to the server.

The *method* attribute of the <go> tag specifies, which HTTP method should be used to send the form data.

If the HTTP POST method is used, the form data to be sent will be placed in the message body of the request. If the HTTP GET method is used, the form data to be sent will be appended to the URL. Since a URL can only contain a limited number of characters, the GET method has the disadvantage that there is a size limit for the data to be sent. If the user data contains non-ASCII characters, you should make use of the POST method to avoid encoding problems.

There is one major difference between HTML and WML. In HTML, the name attribute of the <input> and <select> tags is used to specify the name of the parameter to be sent, while in WML the name attribute of the <postfield> tag is used to do the same thing. In WML, the name attribute of <input> and <select> is used to specify the name of the variable for storing the form data.

Next chapter will teach you how to handle uploaded data at the server end.

15. WML – SERVER SIDE SCRIPTS

If you already know how to write server side scripts for Web Application, then for you, this is very simple to write Server Side program for WML applications. You can use your favorite server-side technology to do the processing required by your mobile Internet application.

At the server side, the parameter name will be used to retrieve the form data.

Consider the following example from the previous chapter to submit name, age, and sex of a person:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="card1" title="WML Form">
<p>
   Name: <input name="name" size="12"/>
   Sex : <select name="sex">
      <option value="male">Male</option>
      <option value="female">Female</option>
      </select>
   Age :  <input name="age" size="12" format="*N"/>
   <anchor>
      <go method="get" href="process.php">
          <postfield name="name" value="$(name)"/>
          <postfield name="age" value="$(age)"/>
          <postfield name="sex" value="$(sex)"/>
      </go>
      Submit Data
    </anchor>
</p>
</card>
```

```
</wml>
```

WML and PHP

Now, we can write a server side script to handle this submitted data in using either PHP, PERL, ASP or JSP. I will show you a server side script written in PHP with HTTP GET method.

Put the following PHP code in process.php file in the same directory where you have your WML file.

```
<?php echo 'Content-type: text/vnd.wap.wml'; ?>
<?php echo '<?xml version="1.0"?'.'>'; ?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

  <card id="card1" title="WML Response">
    <p>
      Data received at the server:<br/>
      Name: <?php echo $_GET["name"]; ?><br/>
      Age: <?php echo $_GET["age"]; ?><br/>
      Sex: <?php echo $_GET["sex"]; ?><br/>
    </p>
  </card>

</wml>
```

If you are using HTTP POST method, then you have to write PHP script accordingly to handle received data. While sending output back to the browser, remember to set the MIME type of the document to "text/vnd.wap.wml".

This way, you can write full-fledged Web Application where lots of database transactions are involved.

You can use **PERL CGI Concepts** to write a dynamic WAP site.

16. WML – EVENTS

Event in ordinary language can be defined as something happened. In programming, **event** is identical in meaning, but with one major difference. When something happens in a computer system, the system itself has to **(1)** detect that something has happened and **(2)** know what to do about it.

WML language also supports events and you can specify an action to be taken whenever an event occurs. This action could be in terms of WMLScript or simply in terms of WML.

WML supports following four event types:

- **onenterbackward**: This event occurs when the user hits a card by normal backward navigational means. That is, user presses the Back key on a later card and arrives back at this card in the history stack.
- **onenterforward**: This event occurs when the user hits a card by normal forward navigational means.
- **onpick**: This is more like an attribute, but it is being used like an event. This event occurs when an item of a selection list is selected or deselected.
- **ontimer**: This event is used to trigger an event after a given time period.

These event names are case sensitive and they must be in lowercase.

WML - onenterbackward Event

This event occurs when the user hits a card by normal backward navigational means. That is, user presses the Back key on a later card and arrives back at this card in the history stack.

Here is the syntax to define an event handler for **onenterbackward** event:

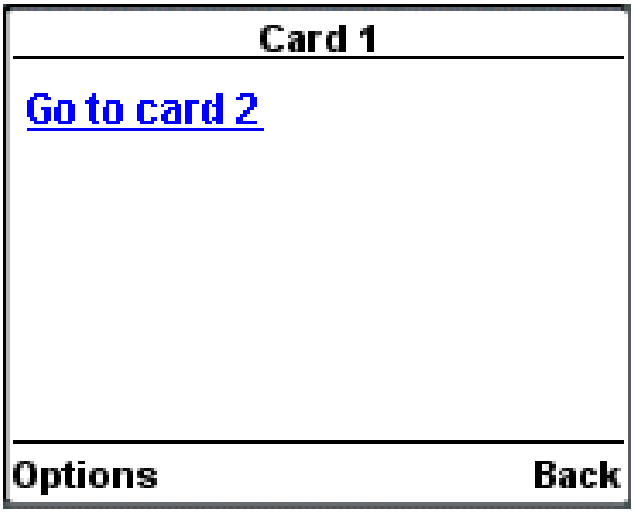
```
<onevent type="onenterbackward">
   A task to be performed.
</onevent>
```

Following is the example showing how **onenterbackward** event occurs whenever you try to go back from the second card to the first card and defined event handler takes you to card number three instead of card number 1. Copy and paste this program and try to play with it to understand **onenterbackward** event type.

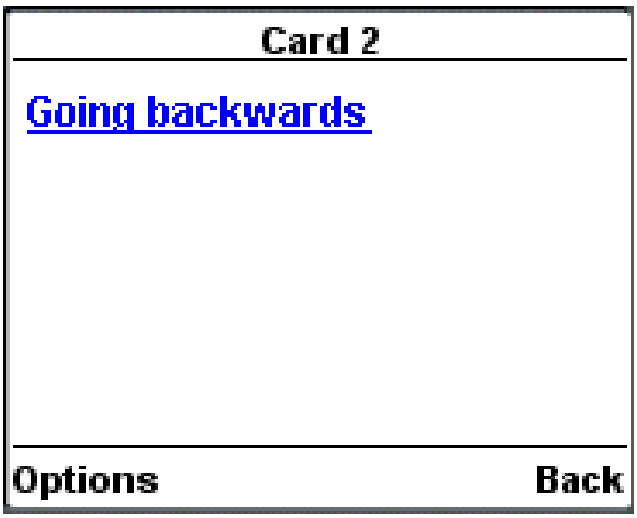
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
```

```
<wml>
<card id="card1" title="Card 1">
<onevent type="onenterbackward">
  <go href="#card3"/>
</onevent>
<p>
  <anchor>
     <go href="#card2"/>
     Go to card 2
  </anchor>
</p>
</card>
<card id="card2" title="Card 2">
<p>
   <anchor>
   <prev/>
      Going backwards
   </anchor>
</p>
</card>
<card id="card3" title="Card 3">
<p>
Hello World!
</p>
</card>
</wml>
```

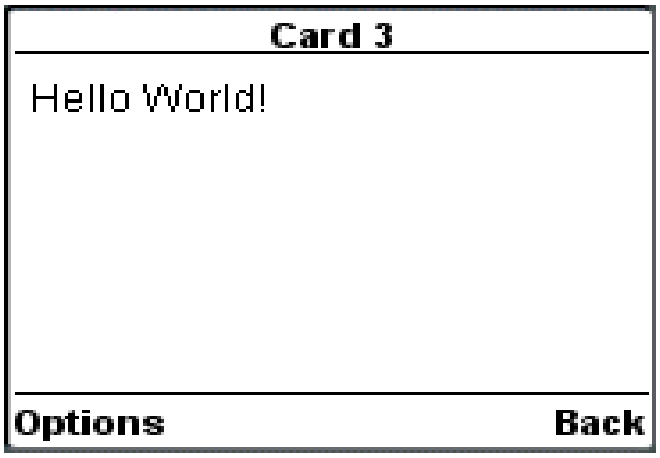
When you load this program, you will get the following screen:



Now, press option **Go to card 2** to go to the second card this will take you to the following screen:



Now, you are on second page. Now, when you try to go back on card number, **onenterbackward** event is executed and it takes you on card number 3 instead of card number 2 and you see the following screen.



WML - onenterforward Event

The onenterforward event is triggered when a user goes to a card in the forward direction. For example, if you go to a card by entering the URL directly or by following an anchor link of which the action is <go>, the onenterforward event will be triggered and the WML code associated with the event will be executed.

The **onenterforward** event will be useful to you if you want to do something before a card is displayed. For example, you need the onenterforward event if you want to assign a value to a variable before a card is displayed.

Here is the syntax to define an event handler for **onenterforward** event:

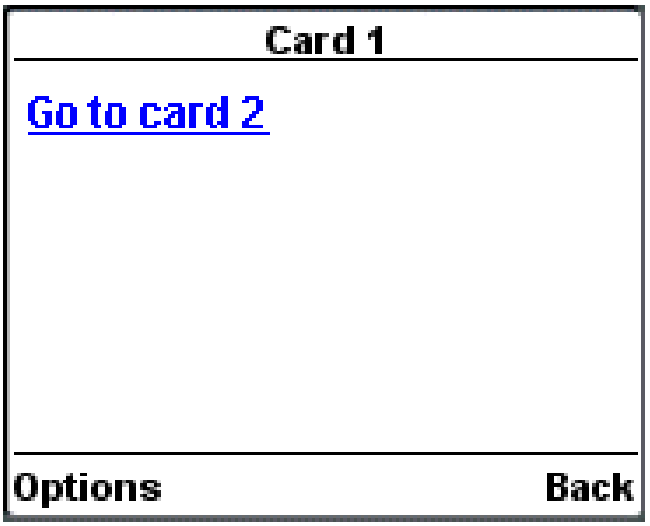
```
<onevent type="onenterforward">
   A task to be performed.
</onevent>
```

Following is the example showing how **onenterbackward** event occurs whenever you try to go on second card from the first card and defined event handler takes you to card number three instead of card number 2. Copy and paste this program and try to play with it to understand **onenterforward** event type.

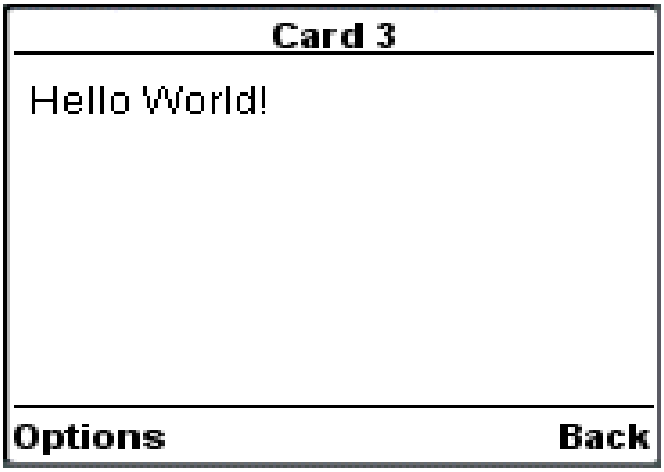
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="card1" title="Card 1">
<p>
  <anchor>
     <go href="#card2"/>
     Go to card 2
  </anchor>
</p>
</card>

<card id="card2" title="Card 2">
<onevent type="onenterforward">
  <go href="#card3"/>
```

```
</onevent>
<p>
  This is card 2
</p>
</card>

<card id="card3" title="Card 3">
<p>
Hello World!
</p>
</card>
</wml>
```

When you load this program, you will get the following screen:



Now, press option **Go to card 2** to go to the second card, because of this, **onenterforward** event will occur and it will take on card number 3 instead of card number 2 and you will see the following screen:



WML - onpick Attribute

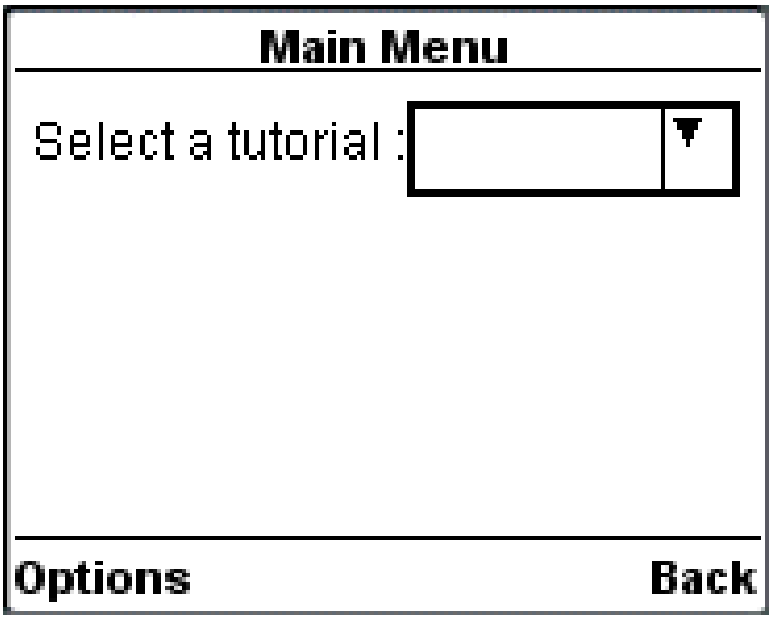
The **onpick** attribute is a great shortcut if you are using a select menu. Instead of writing a lot of codes that allow the user to go to another card if an option is selected, you can simply place the destination into the **onpick** attribute. Here is a code fragment without the onpick attribute:

The following example shows the usage of **onpick** attribute along with <option> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="tutorials" title="Main Menu">
<p>
  Select a tutorial :
  <select title="tutorials" name="selection_list">
    <option onpick="#xhtml">XHTML Tutorial</option>
    <option onpick="#wap">WAP Tutorial</option>
  </select>
</p>
</card>
<card id="xhtml" title="XHTML Tutorial">
<p>
```

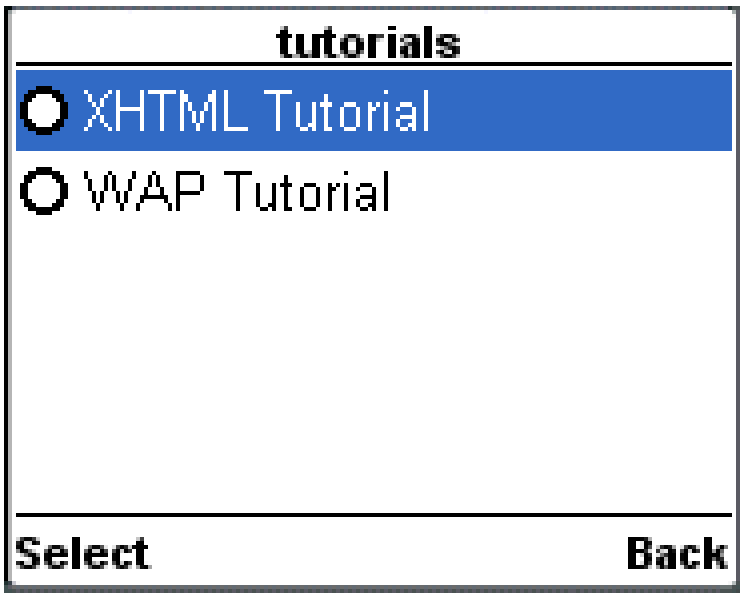
```
Go through our XHTML tutorial
</p>
</card>
<card id="wap" title="WAP Tutorial">
<p>
Go through our WAP tutorial
</p>
</card>
</wml>
```

When you load this program, it will display the following screen:

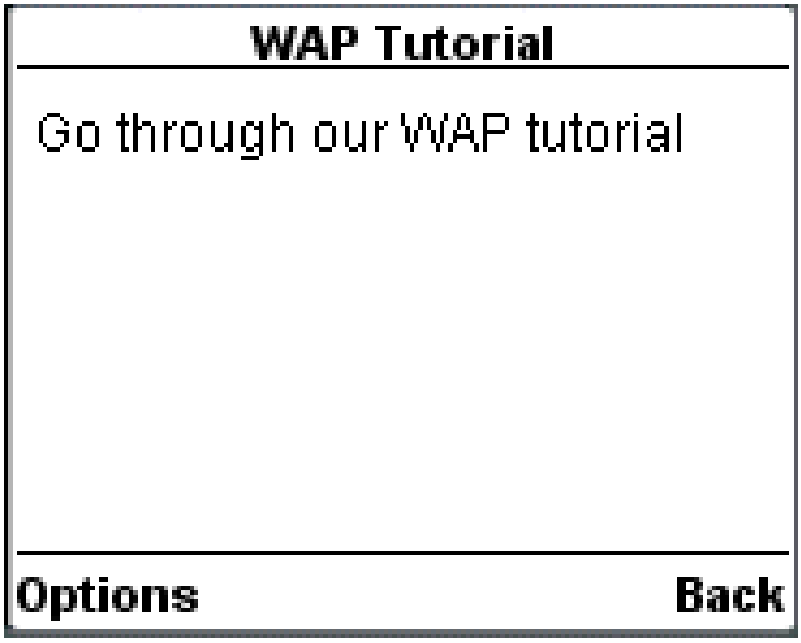


Now, highlight the dropdown box and select it. It will give you two options as follows:

When you load this program, it will display the following screen:



Now, assume, you select **WAP Tutorial** from the list, then it will display the following screen:



WML - ontimer Event

The **ontimer** event is used to trigger an event after a given time period. Let's say you want to display a message after 5 seconds of loading a card, then you can use this event to do so.

Here is the syntax to define an event handler for **ontimer** event:

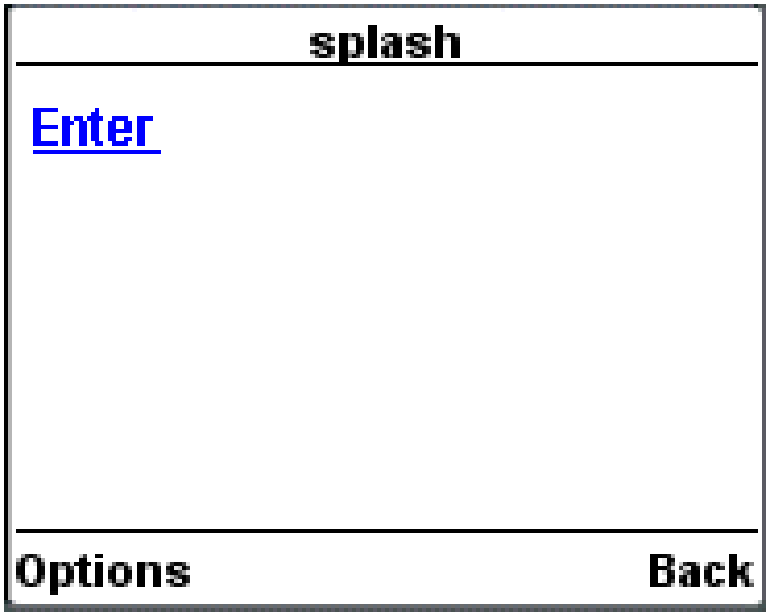
```
<onevent type="ontimer">
   A task to be performed.
</onevent>
<timer value="50"/>
```

Here, it means that a task will be performed after 5 seconds.

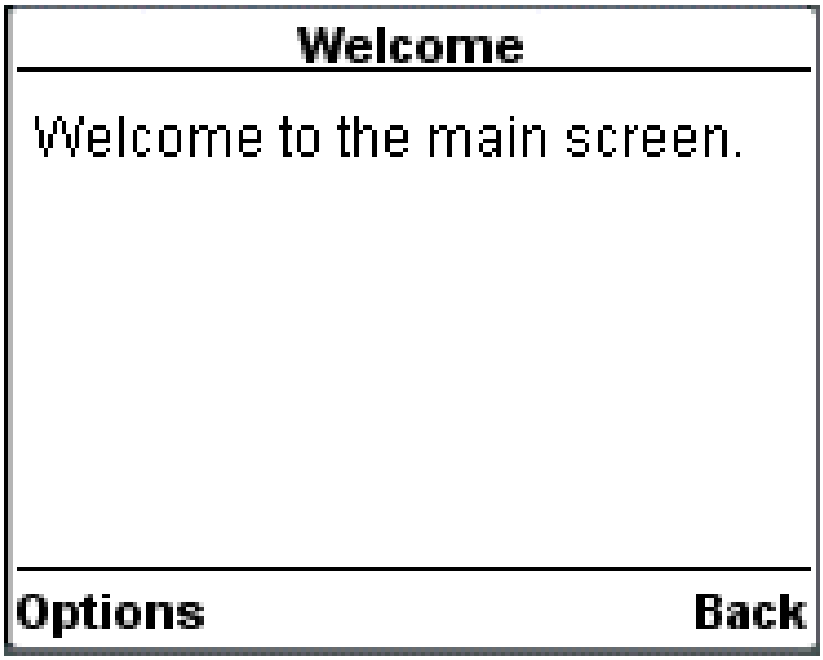
The following example shows the usage of **ontimer** event along with <onevent> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="splash" title="splash">
  <onevent type="ontimer">
    <go href="#welcome"/>
  </onevent>
  <timer value="50"/>
<p>
  <a href="#welcome">Enter</a>
</p>
</card>

<card id="welcome" title="Welcome">
<p>
Welcome to the main screen.
</p>
</card>
</wml>
```

When you load this program, it will display the following screen:



If you do not select the given **Enter** option, then after 5 seconds, you will be directed to **Welcome** page and following screen will be displayed automatically.



WML <onevent> Element

The <onevent>...</onevent> tags are used to create event handlers. Its usage takes the following form:

```
<onevent type="event_type">
```

```
   A task to be performed.
</onevent>
```

You can use either *go, prev,* or *refresh* task inside <onevent>...</onevent> tags against an event.

The <onevent> element supports the following attributes:

Attribute	Value	Description
type	onenterbackward onenterforward onpick ontimer	Defines a type of event occurred.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <onevent> element. In this example, whenever you try to go back from second card to first card, then **onenterbackward** occurs, which moves you to card number three. Copy and paste this program and try to play with it.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
<onevent type="onenterbackward">
  <go href="#card3"/>
</onevent>

<card id="card1" title="Card 1">
<p>
  <anchor>
     <go href="#card2"/>
     Go to card 2
  </anchor>
```

```
</p>
</card>
<card id="card2" title="Card 2">
<p>
   <anchor>
   <prev/>
      Going backwards
   </anchor>
</p>
</card>
<card id="card3" title="Card 3">
<p>
Hello World!
</p>
</card>
</wml>
```

17. WML - TIMER

Previous chapter has described how events are triggered by the users and how do we handle them using event handlers.

Sometime, you may want something to happen without the user explicitly having to activate a control. Yes, WML provides you **ontimer** event to handle this.

The ontimer event is triggered when a card's timer counts down from one to zero, which means that it doesn't occur if the timer is initialized to a timeout of zero.

You can bind a task to this event with the <onevent> element. Here is the syntax:

```
<onevent type="ontimer">
   A task to be performed.
</onevent>
```

Here, a task could be <go>, <prev> or <refresh>.

WML <timer> Element

A timer is declared inside a WML card with the <timer> element. It must follow the <onevent> elements if they are present. (If there is no <onevent> elements, the <timer> must be the first element inside the <card>.) No more than one <timer> may be present in a card.

The <timer> element supports the following attributes:

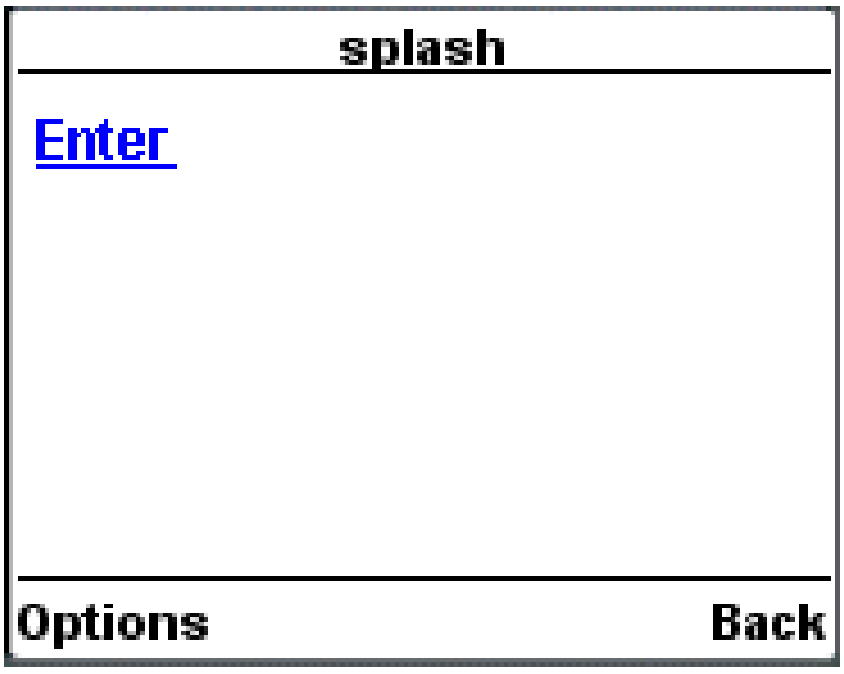
Attribute	Value	Description
name	text	Sets a name for the element.
value	number	Specifies the timer after which timer will be expired. Timeouts are specified in units of a tenth of a second.
class	class_data	Sets a class name for the element.
id	element ID	A unique ID for the element.

The following example shows the usage of <timer> element:

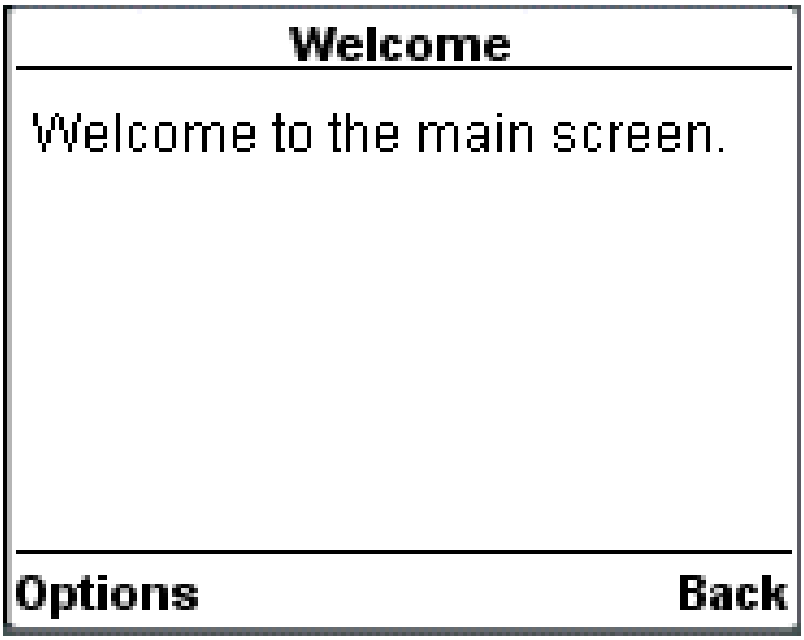
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
```

```
<wml>

<card id="splash" title="splash">
  <onevent type="ontimer">
    <go href="#welcome"/>
  </onevent>
  <timer value="50"/>
<p>
  <a href="#welcome">Enter</a>
</p>
</card>

<card id="welcome" title="Welcome">
<p>
Welcome to the main screen.
</p>
</card>
</wml>
```

When you load this program it shows you following screen:



If you do not select given **Enter** option, then after 5 seconds, you will be directed to **Welcome** page and following screen will be displayed automatically.



18. WML – TEMPLATE

The <template> is used to apply <do> and <onevent> elements to all cards in a deck. This element defines a template for all the cards in a deck and the code in the <template> tag is added to each card in the deck.

You can override a <do> element of a template by defining another <do> element with the same *name* attribute value in a WML card.

The <template> element supports the following attributes:

Attribute	Value	Description
onenterbackward	URL	Occurs when the user navigates into a card using a "prev" task.
onenterforward	URL	Occurs when the user navigates into a card using a "go" task.
ontimer	URL	Occurs when the "timer" expires.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

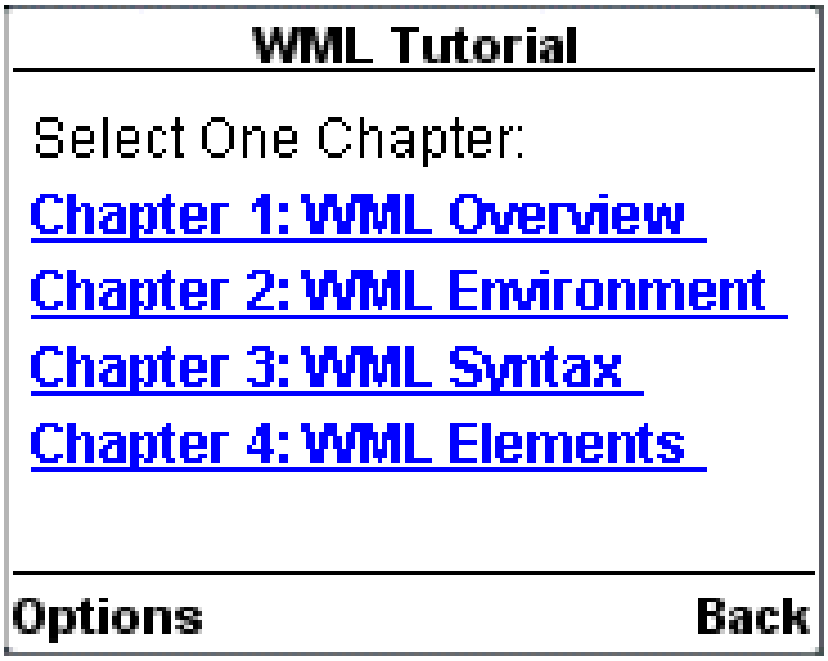
The following example shows the usage of <go> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml13.dtd">

<wml>
  <template>
    <do name="main_menu" type="accept" label="Chapters">
      <go href="chapters"/>
    </do>
    <do name="menu_1" type="accept" label="Chapter 1">
      <go href="#chapter1"/>
    </do>
    <do name="menu_2" type="accept" label="Chapter 2">
```

```
      <go href="#chapter2"/>
    </do>
    <do name="menu_3" type="accept" label="Chapter 3">
      <go href="#chapter3"/>
    </do>
    <do name="menu_4" type="accept" label="Chapter 4">
      <go href="#chapter4"/>
    </do>
  </template>

  <card id="chapters" title="WML Tutorial">
    <p>
      Select One Chapter:<br/>
      <anchor>
        <go href="#chapter1"/>
        Chapter 1: WML Overview
      </anchor><br />

      <anchor>
        <go href="#chapter2"/>
        Chapter 2: WML Environment
      </anchor><br />

      <anchor>
        <go href="#chapter3"/>
        Chapter 3: WML Syntax
      </anchor><br />

      <anchor>
        <go href="#chapter4"/>
        Chapter 4: WML Elements
      </anchor><br />
    </p>
```

```
  </card>

  <card id="chapter1" title="WML Tutorial Ch1">
    <p>
      <em>Chapter 1: WML Introduction</em><br/>
      ...
    </p>
  </card>

  <card id="chapter2" title="WML Tutorial Ch2">
    <p>
      <em>Chapter 2: WML Environment</em><br/>
      ...
    </p>
  </card>

  <card id="chapter3" title="WML Tutorial Ch3">
    <p>
      <em>Chapter 3: WML Syntax</em><br/>
      ...
    </p>
  </card>

  <card id="chapter4" title="WML Tutorial Ch4">
    <p>
      <em>Chapter 4: WML Elements</em><br/>
      ...
    </p>
  </card>
</wml>
```

It will produce the following menu and now you can navigate through all the chapters:



19. THE WML 1.2 – DTD

Here is the complete DTD taken from W3.org. For a latest DTD, please check WML Useful Resources section of this tutorial.

```
<!--
Wireless Markup Language (WML) Document Type Definition.
WML is an XML language. Typical usage:
   <?xml version="1.0"?>
   <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
   "http://www.wapforum.org/DTD/wml12.dtd">
   <wml>
   ...
   </wml>

   Terms and conditions of use are available from the WAP Forum
   Ltd. web site at http://www.wapforum.org/docs/copyright.htm.
-->
<!ENTITY % length  "CDATA">
   <!-- [0-9]+ for pixels or [0-9]+"%" for
   percentage length -->
<!ENTITY % vdata   "CDATA">
   <!-- attribute value possibly containing
   variable references -->
<!ENTITY % HREF    "%vdata;">
   <!-- URI, URL or URN designating a hypertext
   node. May contain variable references -->
<!ENTITY % boolean "(true|false)">
<!ENTITY % number  "NMTOKEN">
   <!-- a number, with format [0-9]+ -->
<!ENTITY % coreattrs   "id     ID     #IMPLIED
                        class  CDATA  #IMPLIED">
<!ENTITY % ContentType "%vdata;">
```

```
<!-- media type. May contain variable references -->

<!ENTITY % emph     "em | strong |b |i |u |big |small">
<!ENTITY % layout   "br">

<!ENTITY % text     "#PCDATA | %emph;">

<!-- flow covers "card-level" elements,
                          such as text and images -->
<!ENTITY % flow     "%text; | %layout; | img | anchor |a |table">

<!-- Task types -->
<!ENTITY % task "go | prev | noop | refresh">

<!-- Navigation and event elements -->
<!ENTITY % navelmts "do | onevent">

<!--================ Decks and Cards ================-->

<!ELEMENT wml ( head?, template?, card+ )>
<!ATTLIST wml
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!-- card intrinsic events -->
<!ENTITY % cardev
  "onenterforward  %HREF;      #IMPLIED
  onenterbackward  %HREF;      #IMPLIED
  ontimer          %HREF;      #IMPLIED"
>

<!-- card field types -->
```

```
<!ENTITY % fields "%flow; | input | select | fieldset">
<!ELEMENT card (onevent*, timer?, (do | p | pre)*)>
<!ATTLIST card
  title            %vdata;     #IMPLIED
  newcontext       %boolean;   "false"
  ordered          %boolean;   "true"
  xml:lang         NMTOKEN     #IMPLIED
  %cardev;
  %coreattrs;
>

<!--================ Event Bindings ================-->

<!ELEMENT do (%task;)>
<!ATTLIST do
  type             CDATA       #REQUIRED
  label            %vdata;     #IMPLIED
  name             NMTOKEN     #IMPLIED
  optional         %boolean;   "false"
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT onevent (%task;)>
<!ATTLIST onevent
  type             CDATA       #REQUIRED
  %coreattrs;
>

<!--================ Deck-level declarations ================-->

<!ELEMENT head ( access | meta )+>
<!ATTLIST head
```

```
  %coreattrs;
>

<!ELEMENT template (%navelmts;)*>
<!ATTLIST template
  %cardev;
  %coreattrs;
>

<!ELEMENT access EMPTY>
<!ATTLIST access
  domain           CDATA       #IMPLIED
  path             CDATA       #IMPLIED
  %coreattrs;
>

<!ELEMENT meta EMPTY>
<!ATTLIST meta
  http-equiv       CDATA       #IMPLIED
  name             CDATA       #IMPLIED
  forua            %boolean;   "false"
  content          CDATA       #REQUIRED
  scheme           CDATA       #IMPLIED
  %coreattrs;
>

<!--================ Tasks ================-->

<!ELEMENT go (postfield | setvar)*>
<!ATTLIST go
  href             %HREF;          #REQUIRED
  sendreferer      %boolean;       "false"
  method           (post|get)      "get"
```

```
  enctype %ContentType; "application/x-www-form-urlencoded"
  accept-charset   CDATA           #IMPLIED
  %coreattrs;
>

<!ELEMENT prev (setvar)*>
<!ATTLIST prev
  %coreattrs;
>

<!ELEMENT refresh (setvar)*>
<!ATTLIST refresh
  %coreattrs;
>

<!ELEMENT noop EMPTY>
<!ATTLIST noop
  %coreattrs;
>

<!--================ postfield ================-->

<!ELEMENT postfield EMPTY>
<!ATTLIST postfield
  name             %vdata;     #REQUIRED
  value            %vdata;     #REQUIRED
  %coreattrs;
>

<!--================ variables ================-->

<!ELEMENT setvar EMPTY>
<!ATTLIST setvar
```

```
  name              %vdata;     #REQUIRED
  value             %vdata;     #REQUIRED
  %coreattrs;
>

<!--================ Card Fields ================-->

<!ELEMENT select (optgroup|option)+>
<!ATTLIST select
  title             %vdata;     #IMPLIED
  name              NMTOKEN     #IMPLIED
  value             %vdata;     #IMPLIED
  iname             NMTOKEN     #IMPLIED
  ivalue            %vdata;     #IMPLIED
  multiple          %boolean;   "false"
  tabindex          %number;    #IMPLIED
  xml:lang          NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT optgroup (optgroup|option)+ >
<!ATTLIST optgroup
  title             %vdata;     #IMPLIED
  xml:lang          NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT option (#PCDATA | onevent)*>
<!ATTLIST option
  value             %vdata;     #IMPLIED
  title             %vdata;     #IMPLIED
  onpick            %HREF;      #IMPLIED
  xml:lang          NMTOKEN     #IMPLIED
```

```
  %coreattrs;
>

<!ELEMENT input EMPTY>
<!ATTLIST input
  name              NMTOKEN         #REQUIRED
  type              (text|password) "text"
  value             %vdata;         #IMPLIED
  format            CDATA           #IMPLIED
  emptyok           %boolean;       "false"
  size              %number;        #IMPLIED
  maxlength         %number;        #IMPLIED
  tabindex          %number;        #IMPLIED
  title             %vdata;         #IMPLIED
  accesskey         %vdata;         #IMPLIED
  xml:lang          NMTOKEN         #IMPLIED
  %coreattrs;
>

<!ELEMENT fieldset (%fields; | do)* >
<!ATTLIST fieldset
  title           %vdata;      #IMPLIED
  xml:lang        NMTOKEN      #IMPLIED
  %coreattrs;
>

<!ELEMENT timer EMPTY>
<!ATTLIST timer
  name              NMTOKEN     #IMPLIED
  value             %vdata;     #REQUIRED
  %coreattrs;
>
```

```
<!--================ Images ================-->

<!ENTITY % IAlign "(top|middle|bottom)" >
<!ELEMENT img EMPTY>
<!ATTLIST img
  alt              %vdata;     #REQUIRED
  src              %HREF;      #REQUIRED
  localsrc         %vdata;     #IMPLIED
  vspace           %length;    "0"
  hspace           %length;    "0"
  align            %IAlign;    "bottom"
  height           %length;    #IMPLIED
  width            %length;    #IMPLIED
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!--================ Anchor ================-->

<!ELEMENT anchor ( #PCDATA | br | img | go | prev | refresh )*>
<!ATTLIST anchor
  title            %vdata;     #IMPLIED
  accesskey        %vdata;     #IMPLIED
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>
<!ELEMENT a ( #PCDATA | br | img )*>
<!ATTLIST a
  href             %HREF;      #REQUIRED
  title            %vdata;     #IMPLIED
  accesskey        %vdata;     #IMPLIED
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
```

```
>

<!--================ Tables ================-->

<!ELEMENT table (tr)+>
<!ATTLIST table
  title            %vdata;     #IMPLIED
  align            CDATA       #IMPLIED
  columns          %number;    #REQUIRED
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT tr (td)+>
<!ATTLIST tr
  %coreattrs;
>
<!ELEMENT td ( %text; | %layout; | img | anchor |a )*>
<!ATTLIST td
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!--============= Text layout and line breaks ==============-->

<!ELEMENT em (%flow;)*>
<!ATTLIST em
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT strong (%flow;)*>
<!ATTLIST strong
```

```
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT b (%flow;)*>
<!ATTLIST b
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT i (%flow;)*>
<!ATTLIST i
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT u (%flow;)*>
<!ATTLIST u
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT big (%flow;)*>
<!ATTLIST big
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT small (%flow;)*>
<!ATTLIST small
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>
```

```
<!ENTITY % TAlign    "(left|right|center)">
<!ENTITY % WrapMode  "(wrap|nowrap)" >

<!ELEMENT p (%fields; | do)*>
<!ATTLIST p
  align            %TAlign;    "left"
  mode             %WrapMode;  #IMPLIED
  xml:lang         NMTOKEN     #IMPLIED
  %coreattrs;
>

<!ELEMENT br EMPTY>
<!ATTLIST br
  %coreattrs;
>

<!ELEMENT pre (#PCDATA | a | br | i | b | em | strong |
                                        input | select )*>
<!ATTLIST pre
  xml:space        CDATA       #FIXED    "preserve"
  %coreattrs;
>

<!ENTITY quot """>       <!-- quotation mark -->
<!ENTITY amp  "&#38;">   <!-- ampersand -->
<!ENTITY apos "'">       <!-- apostrophe -->
<!ENTITY lt   "&#60;">   <!-- less than -->
<!ENTITY gt   ">">       <!-- greater than -->
<!ENTITY nbsp " ">      <!-- non-breaking space -->
<!ENTITY shy  ""> <!-- soft hyphen (discretionary hyphen) -->
```

20. WML 2.0

WML2 is a language, which extends the syntax and semantics of the followings:

- XHTML Basic [XHTMLBasic]
- CSS Mobile Profile [CSSMP]
- Unique semantics of WML1.0 [WML1.0]

WML2 is optimized for specifying presentation and user interaction on limited capability devices such as mobile phones and other wireless mobile terminals.

This tutorial gives detail of the Wireless Markup Language (WML) Version 2. This tutorial refers to version 2.0 of WML as WML2.

The XHTML Basic defined by the W3C is a proper subset of XHTML, which is a reformulation of HTML in XML.

Basic Goals of WML2

There are five major goals for WML2:

- Backward compatibility: WML2 application should be running on old devices as well.

- Convergence with existing and evolving Internet standards: XHTML Basic [XHTMLBasic] and CSS Mobile Profile [CSSMP].

- Optimization of access from small, limited devices: WAP-enabled devices are generally small and battery operated and they have relatively limited memory and CPU power. So WML2 should be optimized enough to run on these devices.

- Allowance for the creation of distinct user interfaces: WAP enables the creation of Man Machine Interfaces (MMIs) with maximum flexibility and ability for a vendor to enhance the user experience.

- Internationalization of the architecture: WAP targets common character codes for international use. This includes international symbols and pictogram sets for the end users, and local-use character encoding for the content developers.

WML2 Vision

The WML2 vision is to create a language that extends the syntax and semantics of XHTML Basic and CSS Mobile profile with the unique semantics of WML1. The user should not be aware of how WML1 compatibility is achieved.

The WML2 Language Structure

WML2 is a new language with the following components:

(1) XHTML Basic

This element group is for the W3C convergence. For some of the elements, WML extension attributes are added in order to achieve WML1 functionality.

(1a) XHTML Basic elements

a abbr acronym address base blockquote br caption cite code dd dfn div dl dt em form h1 h2 h3 h4 h5 h6 head kbd label li link object ol param pre q samp span strong table td th title tr ul var

(1b) XHTML Basic elements with WML extension attributes

body html img input meta option p select style textarea

(2) XHTML Modularization elements

This element group consists of select elements from those modules of XHTML not included in XHTML Basic. Most of the elements are included for the WML1 compatibility. One element is included as an enhancement that fits limited handset capabilities.

(2a) XHTML Modularization for backwards compatibility with WML1

b big i small (from Presentation Module) u (from Legacy Module) fieldset optgroup (from Forms Module)

(2b) XHTML Modularization elements for feature enhancement

hr

(3) WML extensions elements

Some elements are brought from WML1, because the equivalent capabilities are not provided in XHTML Basic or XHTML Modularization. One element is included for enhancement of WML1 capabilities.

(3a) WML extensions elements (for WML1 compatibility)

wml:access wml:anchor wml:card wml:do wml:getvar wml:go wml:noop wml:onevent wml:postfield wml:prev wml:refresh wml:setvar wml:timer

(3b) WML extensions elements (for feature enhancement)

wml:widget

WML Document Structure Modules

The following elements in the Structure Module are used to specify the structure of a WML2 document:

- body
- html
- wml:card
- head
- title

The body Element

The wml:newcontext attribute specifies whether the browser context is initialized to a well-defined state when the document is loaded. If the wml:newcontext attribute value is "true", the browser MUST reinitialize the browser context upon navigation to this card.

The html Element

The xmlns:wml attribute refers to the WML namespace for example : http://www.wapforum.org/2001/wml.

The wml:use-xml-fragments attribute is used to specify how a fragment identifier is interpreted by the user agent.

The wml:card Element

The wml:card element specifies a fragment of the document body. Multiple wml:card elements may appear in a single document. Each wml:card element represents an individual presentation and/or interaction with the user.

If the wml:card element's newcontext attribute value is "true", the browser MUST reinitialize the browser context upon navigation to this card.

The head Element

This element keeps header information of the document like meta element and style sheet etc.

The title Element

This element is used to put a document title.

NOTE: WML developers can use the XHTML document style, that is, body structure, or they can use a collection of cards. When the body structure is used, a document is constructed using a body element. The body element contains the content of the document. When a collection of cards is used, a document is constructed using one or more wml:card elements.

WML2 Tasks

The following tasks are defined in WML2.0. These tasks are very similar to WML1.0

- The go task
- The prev task
- The noop task
- The refresh task

WML2 Events

The following event types are defined in WML2:

- **Intrinsic event:** An event generated by the user agent and includes the following events similar to WML1.0
  - ontimer
  - onenterforward
  - onenterbackward
  - onpick

- **Extrinsic event:** An event sent to the user agent by some external agent. The WML 2 specification does not specify any classes of extrinsic events. One example of a WML extrinsic event class may be WTA events.

WML2 Document Type

WML2 documents are identified by the MIME media type "application/wml+xml". The type "application/xhtml+xml" can be used to identify documents from any of the XHTML-based markup languages, including XHTML Basic.

The DOCTYPE declaration may include the XHTML Basic Formal Public Identifier and may also include the URI of the XHTML Basic DTD as specified below:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"
"http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
```

Style Sheets with WML2

Style sheets can be used to style WML2 documents. Style information can be associated with a document in 3 ways:

External style sheet

An external style sheet can be associated with a document using a special XML processing instruction or the link element. The use of the XML processing instruction can also be used.

In the following example, the XML processing instruction is used to associate the external style sheet "mobile.css".

```
<?xml-stylesheet href="mobile.css"
                    media="handheld" type="text/css" ?>
```

In the following example, the link element is used to associate the external style sheet "mystyle.css":

```
<html>
<head>
<link href="mystyle.css" type="text/css" rel="stylesheet"/>
...
</head>
...
</html>
```

Internal Style Sheets

Style information can be located within the document using the style element. This element, like link, must be located in the document header.

The following shows an example of an internal style sheet:

```
<html>
<head>
<style type="text/css">
p { text-align: center; }
</style>
...
</head>
```

```
...
</html>
```

Inline Style

You can specify style information for a single element using the *style* attribute. This is called **inline style**.

In the following example, inline styling information is applied to a specific paragraph element:

```
<p style="text-align: center">...</p>
```

The WML2 Default Style Sheet

Here is a sample style sheet for WML 2.0:

```
body, card, div, p, center, hr, h1, h2, h3, h4, h5, h6,
address, blockquote, pre, ol, ul, dl, dt, dd,
form, fieldset, object { display: block }
li      { display: list-item }
head    { display: none }
table   { display: table }
tr      { display: table-row }
td, th  { display: table-cell }
caption { display: table-caption }
th      { font-weight: bolder; text-align: center }
caption { text-align: center }
h1, h2, h3, h4, h5, h6, b, strong { font-weight: bolder }
i, cite, em, var,address { font-style: italic }
pre, code, kbd, pre { white-space: pre }
big     { font-size: larger}
small   { font-size: smaller}
hr      { border: 1px inset }
ol      { list-style-type: decimal }
u       { text-decoration: underline }
```

The WML2 Elements

Here is the link to a complete list of all the WML2 elements: WML2 Tags Reference

Most of the elements are available in XHTML specification except a few elements starting with *WML:* These elements are specific to WML.

All the elements have the same meaning here as in XHTML specification.

21. WML – ENTITIES

WML entities represent symbols that either can't easily be typed in or that have a special meaning in WML.

For example, if you put a < character into your text normally, the browser thinks it's the start of a tag; the browser then complains when it can't find the matching > character to end the tag.

The following table displays the three forms of entities in WML. Named entities are something you may be familiar with from HTML: they look like & or <, and they represent a single named character via a mnemonic name. Entities can also be entered in one of two numeric forms (decimal or hexadecimal), allowing you to enter any Unicode character into your WML.

Named Entity	Decimal Entity	Hexa Entity	Character
"	"	"	Double quote (")
&	&	&	Ampersand (&)
'	'	'	Apostrophe (')
<	<	<	Less than (<)
>	>	>	Greater than (>)
			Nonbreaking space
			Soft hyphen

Note that all entities start with an ampersand (&) and end with a semicolon (;). This semicolon is very important: some web pages forget this and cause problems for browsers that want correct HTML. WAP browsers also are likely to be stricter about errors like these.

22. WML – TAGS REFERENCE

The following table lists all the valid WML elements. Click over the links to know more detail of that element.

Deck & Card Elements

WML Elements	Purpose
<!-->	Defines a WML comment.
<wml>	Defines a WML deck (WML root).
<head>	Defines head information.
<meta>	Defines meta information.
<card>	Defines a card in a deck.
<access>	Defines information about the access control of a deck.
<template>	Defines a code template for all the cards in a deck.

WML <!–...–> Tag

The WML <!--...> tag is used to comment out a portion of WML code.

A comment starts with the four characters <!-- and ends with the three characters --> . Everything that appears between these two markers, including tags, body text, entities, and line breaks, is ignored.

If for some reason, you want the sequence <!-- in your body text, write it with an entity (which you would have to do for the < anyway):

```
&lt;
```

WML does not allow nesting of comments. This means you cannot keep comment inside a comment. The following doesn't work:

```
<!-- A simple <!-- EMBEDDED COMMENT, NOT! --> comment. -->
```

Note that the WAP gateway removes all comments as part of its processing, so the browser doesn't even know they exist and there is no effect on traffic due to your comments.

Attributes

There is no attribute related to <!--...--> tag.

Example

The following example shows the usage of this element:

```
<!-- This will be assumed as a comment -->
```

A multiline comment can be given as follows:

```
<!-- This is a multi-line
     comment -->
```

WML <wml> Tag

The WML <wml> tag is used to define a WML deck and contains cards and other elements of the document.

The <wml> element serves a purpose much like the <html> element does for the HTML pages.

Attributes

The <wml> element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="one" title="First Card">
```

```
<p>
This is the first card in the deck
</p>
</card>

<card id="two" title="Second Card">
<p>
Ths is the second card in the deck
</p>
</card>

</wml>
```

WML <head> Tag

The <head> element in WML is similar to the <head> element in HTML.

It marks a place for meta-information about the document to be stored. Meta-information is information about the document itself, rather than its content.

If present, this element must be the first thing inside the <wml> element.

Attributes

The <head> element supports the following attributes:

Attribute	Value	Description
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<head>
```

```
   <access domain="www.tutorialspoint.com"/>
   <meta name="keyword" content="WML"/>
</head>

<wml>

<card id="one" title="First Card">
<p>
This is the first card in the deck
</p>
</card>

<card id="two" title="Second Card">
<p>
Ths is the second card in the deck
</p>
</card>

</wml>
```

WML <meta> Tag

The <meta> element places an item of arbitrary meta-information in a WML deck. This item is structured as a property name and its value.

You can put any number of <meta> elements into the <head> element. This can add keywords for indexing purposes, store hints about the content of the deck, and store any other information.

Attributes

The <meta> element supports the following attributes:

Attribute	Value	Description
name	string	Gives the name of this property. Meta-information with this attribute is intended for server-side applications, so it may be removed before it gets to the browser. Could be "keywords", "author", etc.

http-equiv	string	An alternative for the name attribute.
forua	true false	If present and set to true, indicates that the property is intended for the use of the browser.
content	string	Should specify a description of the name attribute.
scheme	string	Can specify a format or structure that some properties may need to interpret their values. This attribute is used by a few properties.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<head>
   <meta name="keyword" content="WML"/>
   <meta http-equiv="Cache-control" content="no-cache"/>
</head>

<wml>

<card id="one" title="First Card">
<p>
This is the first card in the deck
</p>
</card>

<card id="two" title="Second Card">
<p>
```

```
Ths is the second card in the deck
</p>
</card>


</wml>
```

WML <card> Tag

The <card> element encloses a WML card within a deck. In addition, text and graphics enclosed within <p> elements, it may also contain a number of event bindings.

Attributes

The <card> element supports the following attributes:

Attribute	Value	Description
title	cdata	Gives a title to this card. This title is displayed in some way by the browser when the card is visible.
newcontext	true false	Specifies that when this card is entered, the browser context should be cleared.
ordered	true false	Provides a hint to the browser about how the card is organized. Set it to true if the card consists of a number of separate fields that should be dealt with in the order they appear in the card. Set it to false if the card contains optional fields or may be filled in out of order.
onenterforward	URL	Occurs when the user navigates into a card using a "go" task.
onenterbackward	URL	Occurs when the user navigates into a card using a "prev" task.
ontimer	URL	Occurs when a "timer" expires.
xml:lang	language_code	Sets the language used in the element.
class	cdata	Sets a class name for the element.
id	element_ID	A unique ID for the element.

Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>

<card id="one" title="First Card">
<p>
This is the first card in the deck
</p>
</card>

<card id="two" title="Second Card">
<p>
Ths is the second card in the deck
</p>
</card>

</wml>
```

WML<access>Tag

The <access> element provides a simple form of access control for a deck. This allows a deck to specify that only certain other decks may link to it (these decks are known as referring URLs).

There may be no more than one <access> element in a deck, and it must be the first thing inside the <head> element.

Attributes

The <access> element supports the following attributes:

Attribute	Value	Description
domain	cdata	Specifies the domain (effectively, the range of servers) from which the referring deck must come.

path	cdata	Specifies the path within the referring URL that must match. (The path is the part of the URL that specifies a file or directory on the server.)
class	cdata	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element:

```
Access is limited to other decks in www.tutorialspoint.com
<head>
   <access domain="www.tutorialspoint.com"/>
</head>

Access is limited to other decks in www.tutorialspoint.com/wml
<head>
   <access domain="www.tutorialspoint.com" path="/wml"/>
</head>
```

WML <template> Tag

The <template> is used to apply <do> and <onevent> elements to all cards in a deck. This element defines a template for all the cards in a deck and the code in the <template> tag is added to each card in the deck.

You can override a <do> element of a template by defining another <do> element with the same *name* attribute value in a WML card.

Attributes

The <template> element supports the following attributes:

Attribute	Value	Description
onenterbackward	URL	Occurs when the user navigates into a card using a "prev" task.
onenterforward	URL	Occurs when the user navigates into a card using a "go" task.
ontimer	URL	Occurs when the "timer" expires.

class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

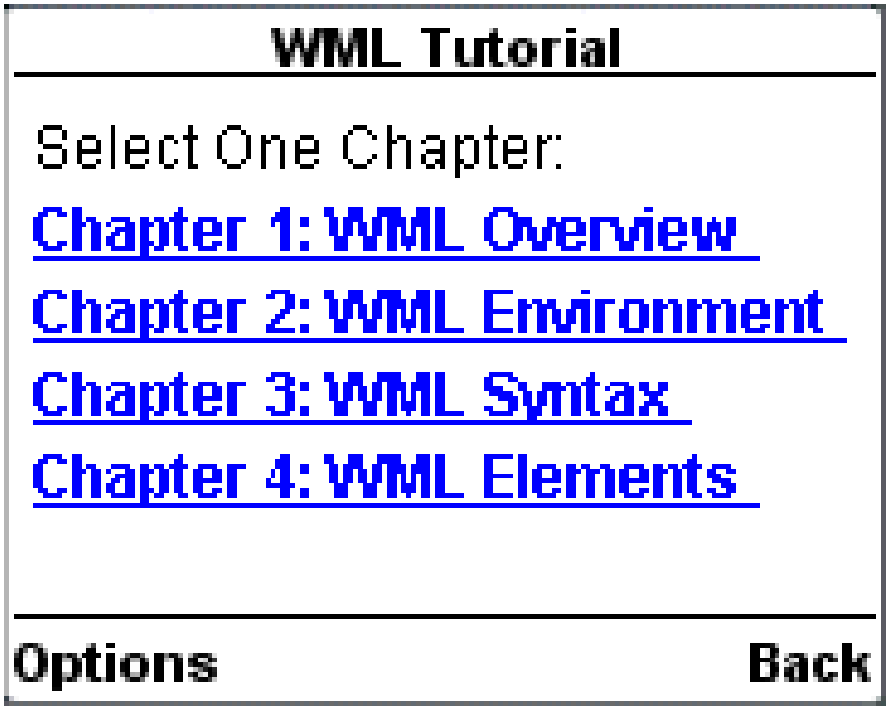
The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
"http://www.wapforum.org/DTD/wml13.dtd">

<wml>
  <template>
    <do name="main_menu" type="accept" label="Chapters">
      <go href="chapters"/>
    </do>
    <do name="menu_1" type="accept" label="Chapter 1">

      <go href="#chapter1"/>
    </do>
    <do name="menu_2" type="accept" label="Chapter 2">
      <go href="#chapter2"/>
    </do>
    <do name="menu_3" type="accept" label="Chapter 3">

      <go href="#chapter3"/>
    </do>
    <do name="menu_4" type="accept" label="Chapter 4">
      <go href="#chapter4"/>
    </do>
  </template>

  <card id="chapters" title="WML Tutorial">
    <p>
```

```
      Select One Chapter:<br/>
      <anchor>
        <go href="#chapter1"/>

        Chapter 1: WML Overview
      </anchor><br />

      <anchor>
        <go href="#chapter2"/>
        Chapter 2: WML Environment
      </anchor><br />

      <anchor>
        <go href="#chapter3"/>
        Chapter 3: WML Syntax
      </anchor><br />

      <anchor>
        <go href="#chapter4"/>

        Chapter 4: WML Elements
      </anchor><br />
    </p>
  </card>

  <card id="chapter1" title="WML Tutorial Ch1">
    <p>

      <em>Chapter 1: WML Introduction</em><br/>
      ...
    </p>
  </card>
```

```
  <card id="chapter2" title="WML Tutorial Ch2">

    <p>
      <em>Chapter 2: WML Environment</em><br/>
      ...
    </p>
  </card>

  <card id="chapter3" title="WML Tutorial Ch3">
    <p>
      <em>Chapter 3: WML Syntax</em><br/>
      ...
    </p>
  </card>

  <card id="chapter4" title="WML Tutorial Ch4">
    <p>
      <em>Chapter 4: WML Elements</em><br/>
      ...
    </p>

  </card>
</wml>
```

It will produce the following menu and now you can navigate through all the chapters:



Text Elements

WML Elements	Purpose
** **	Defines a line break
<p>	Defines a paragraph
<table>	Defines a table
<td>	Defines a table cell (table data)
<tr>	Defines a table row
<pre>	Defines preformatted text

WML
 Tag

The <br /> element defines a line break and almost all WAP browsers support a line break tag.

Attributes

The <br /> element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.

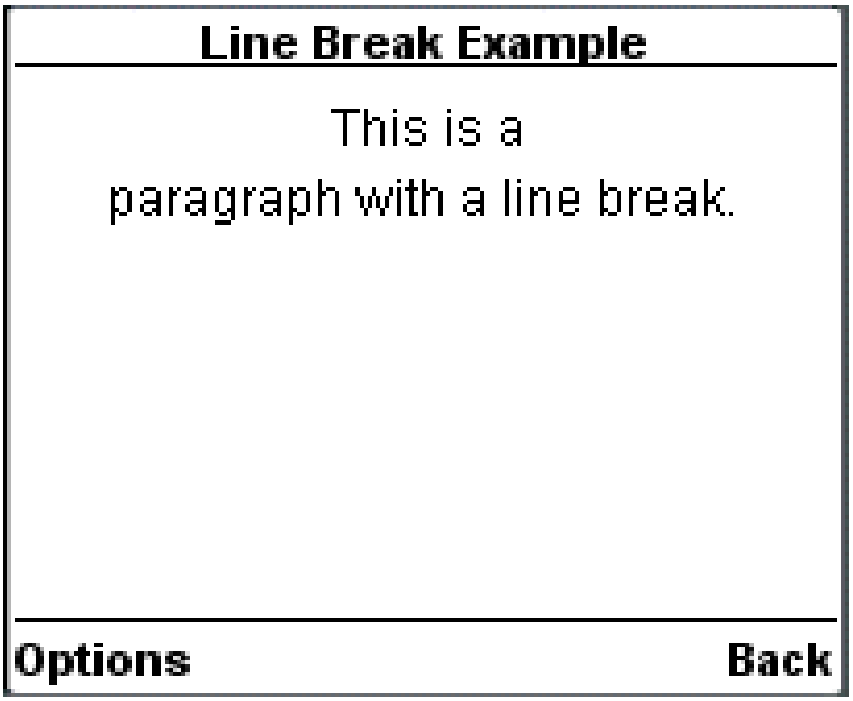
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of <br /> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Line Break Example">
<p align="center">
This is a <br /> paragraph with a line break.
</p>
</card>

</wml>
```

It will produce the following result:



WML <p> Tag

The <p> element defines a paragraph of text and WAP browsers always render a paragraph in a new line.

A <p> element is required to define any text, image, or a table in WML.

Attributes

The <p> element supports the following attributes:

Attribute	Value	Description
align	left right center	This is used to change the horizontal alignment of a paragraph.
mode	wrap nowrap	Sets whether a paragraph should wrap lines or not.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

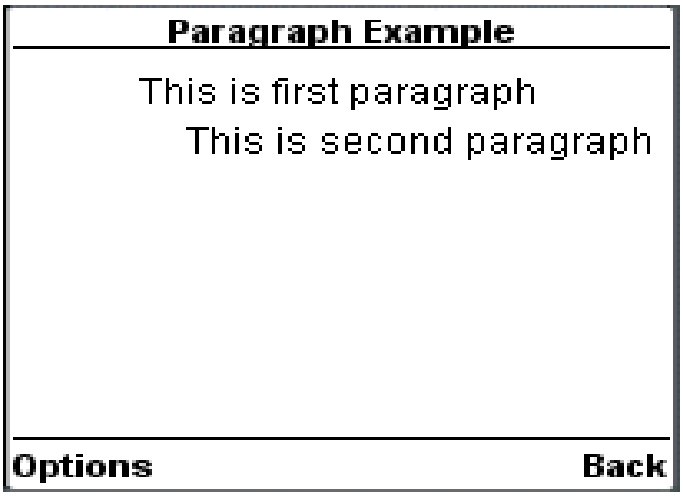
Example

The following example shows the usage of <p> element.

```
<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Paragraph Example">
<p align="center">
This is first  paragraph
</p>
<p align="right">
This is second  paragraph
```

```
</p>
</card>


</wml>
```

It will produce the following result:



WML <table> Tag

The <table> element along with <tr> and <td> is used to create a table in WML. WML does not allow the nesting of tables.

A <table> element should be put with-in ... elements.

Attributes

The <table /> element supports the following attributes:

Attribute	Value	Description
columns	number	Sets the number of columns in the table.
align	L C R	To specify the horizontal text alignment of the columns, you need to assign three letters to the align attribute. Each letter represents the horizontal text alignment of a column. The letter can be L, C, or R. For example, if you want the following settings to be applied to your table: First table column -- Left-aligned Second table column -- Center-aligned Third table column -- Right-aligned

		Then you should set the value of the align attribute to LCR.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

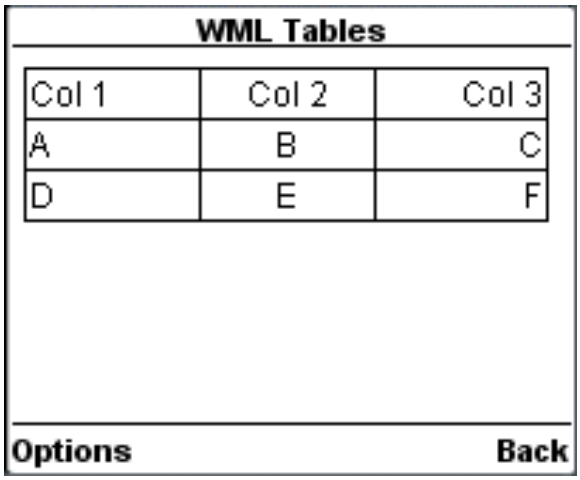
Example

The following example shows the usage of <table> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="WML Tables">
<p>
<table columns="3" align="LCR">
     <tr>
       <td>Col 1</td>
       <td>Col 2</td>
       <td>Col 3</td>
     </tr>

     <tr>
       <td>A</td>
       <td>B</td>
       <td>C</td>
     </tr>

     <tr>
       <td>D</td>
       <td>E</td>
```

```
      <td>F</td>
    </tr>
</table>
</p>
</card>

</wml>
```

It will produce the following result:



WML <td> Tag

The <td> element encloses a single cell within a table.

It may appear only inside a <tr> element. It takes no attributes and may contain the following: flow text; images using the <img> element; text style changes using the , , <b>, <i>, <u>, <big>, and <small> elements; and anchored text using the <a> or <anchor> elements.

An empty <td> element, or one containing only whitespace, is legal and significant.

Attributes

This element supports the following attributes:

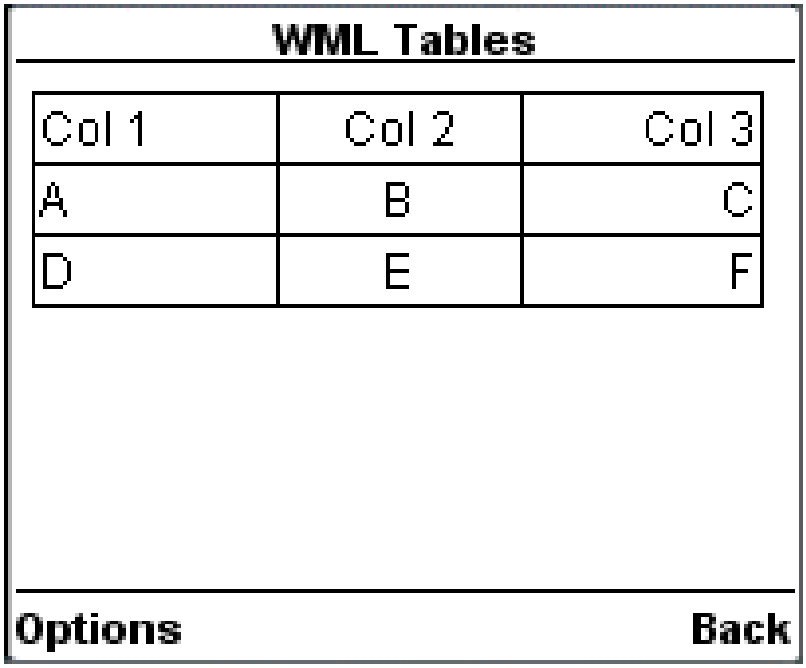
Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="WML Tables">
<p>
<table columns="3" align="LCR">
     <tr>
       <td>Col 1</td>
       <td>Col 2</td>
       <td>Col 3</td>
     </tr>

     <tr>
       <td>A</td>
       <td>B</td>
       <td>C</td>
     </tr>

     <tr>
       <td>D</td>
       <td>E</td>
       <td>F</td>
     </tr>
</table>
</p>
</card>

</wml>
```

It will produce the following result:



WML <tr> Tag

The <td> element defines a table row.

This element can contain only <td> elements, giving the cells within the row. It's legal for a <tr> element to contain no <td> elements or only empty ones. This indicates an empty row in the table.

Attributes

This element supports the following attributes:

Attribute	Value	Description
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

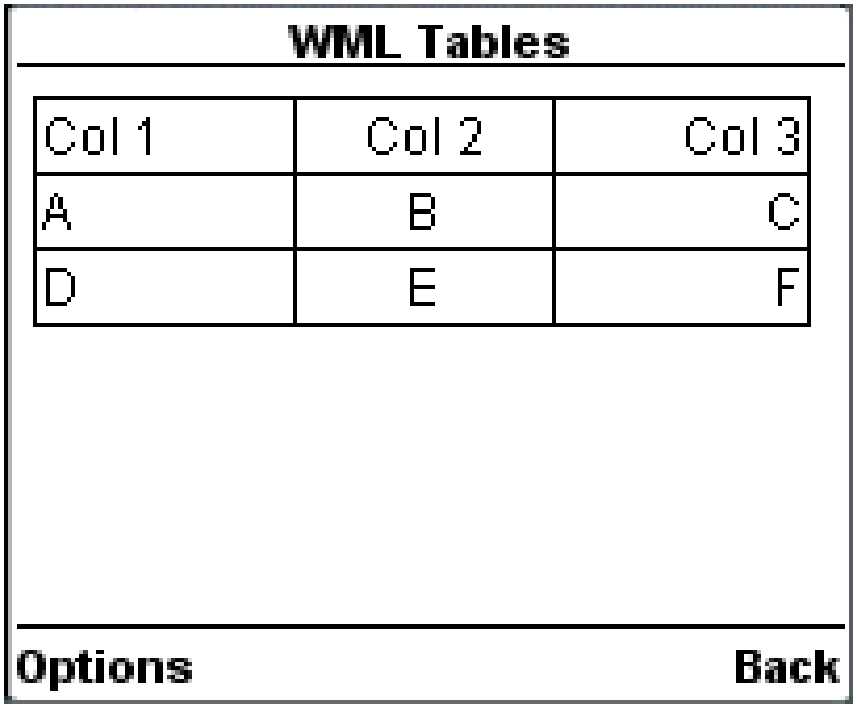
Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
```

```
<card title="WML Tables">
<p>
<table columns="3" align="LCR">
     <tr>
       <td>Col 1</td>
       <td>Col 2</td>
       <td>Col 3</td>
     </tr>

     <tr>
       <td>A</td>
       <td>B</td>
       <td>C</td>
     </tr>

     <tr>
       <td>D</td>
       <td>E</td>
       <td>F</td>
     </tr>
</table>
</p>
</card>

</wml>
```

It will produce the following result:



WML <pre> Tag

The <pre> element is used to specify preformatted text in WML. Preformatted text is text of which the format follows the way it is typed in the WML document.

This tag preserves all the white spaces enclosed inside this tag. Make sure you are not putting this tag inside ...

Attributes

The <pre> element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

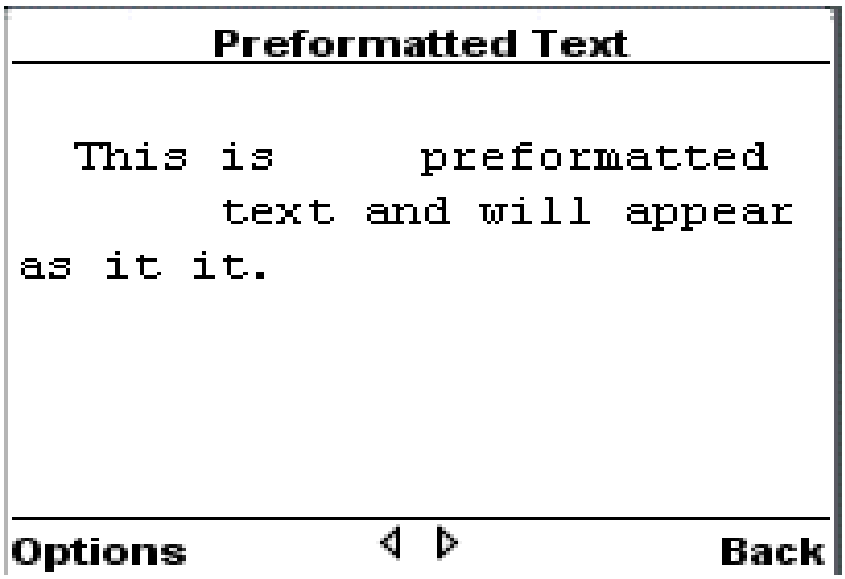
Example

The following example shows the usage of <pre> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
```

```
<wml>

<card title="Preformatted Text">
<pre>
   This is     preformatted
        text and will appear
 as it it.
</pre>
</card>

</wml>
```

It will produce the following result:



Text Formatting Tags

WML Elements	Purpose
<b>	Defines bold text.
<big>	Defines big text.
****	Defines emphasized text.
<i>	Defines italic text.
<small>	Defines small text.
****	Defines strong text.

<u>	Defines underlined text.

WML <b> Tag

The <b> element defines a bold text.

Attributes

This element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

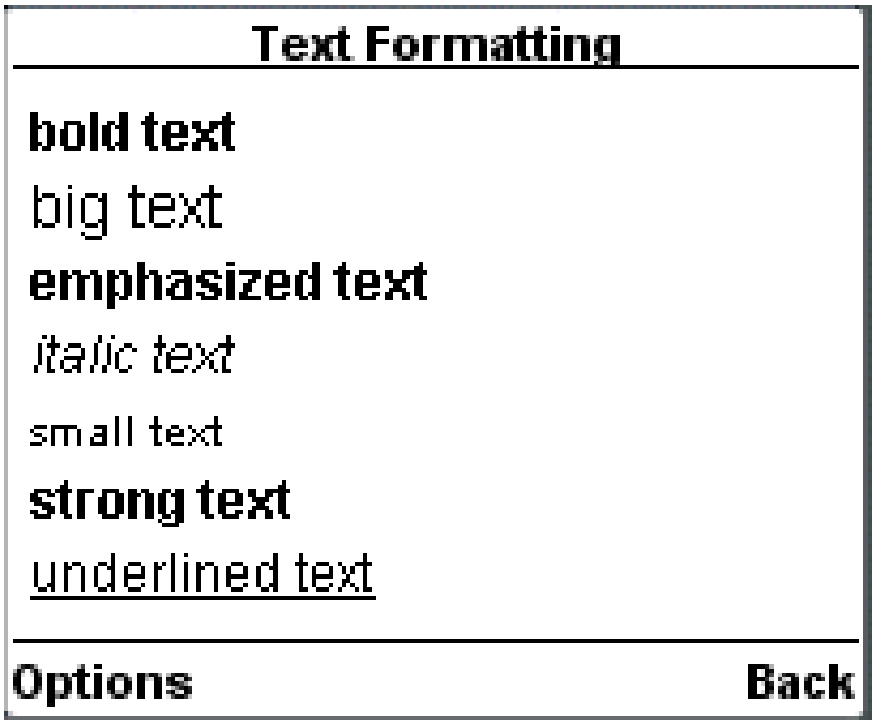
Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Text Formatting">
<p>
  <b>bold text</b><br/>
  <big>big text</big><br/>
  <em>emphasized text</em><br/>
  <i>italic text</i><br/>
  <small>small text</small><br/>
  <strong>strong text</strong><br/>
  <u>underlined text</u>
</p>
```

```
</card>


</wml>
```

It will produce the following result:



WML <big> Tag

The <big> element defines a big text.

Attributes

This element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

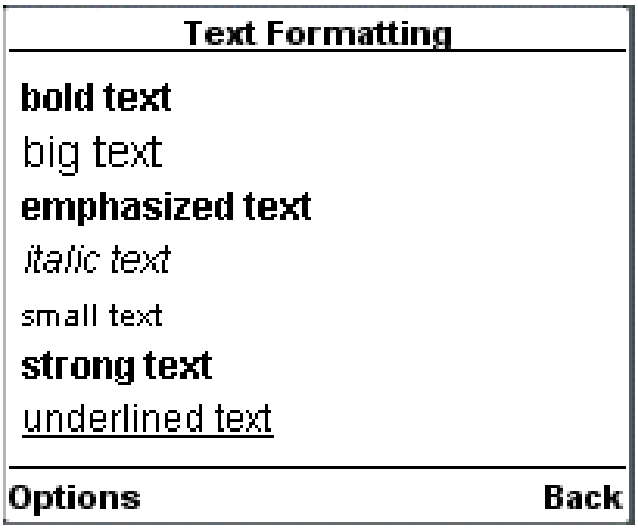
Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
```

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Text Formatting">
<p>
  <b>bold text</b><br/>
  <big>big text</big><br/>
  <em>emphasized text</em><br/>
  <i>italic text</i><br/>
  <small>small text</small><br/>
  <strong>strong text</strong><br/>
  <u>underlined text</u>
</p>
</card>

</wml>
```

It will produce the following result:



WML Tag

The element defines an emphasized text.

Attributes

This element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

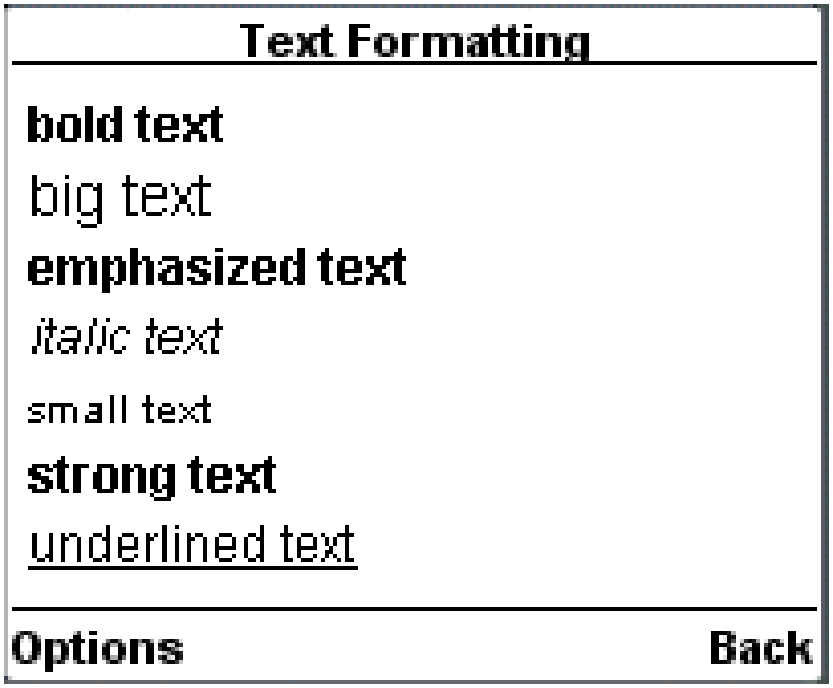
Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Text Formatting">
<p>
  <b>bold text</b><br/>
  <big>big text</big><br/>
  <em>emphasized text</em><br/>
  <i>italic text</i><br/>
  <small>small text</small><br/>
  <strong>strong text</strong><br/>
  <u>underlined text</u>
</p>
</card>
```

```
</wml>
```

It will produce the following result:



WML <i> Tag

The <i> element defines an italic text.

Attributes

This element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

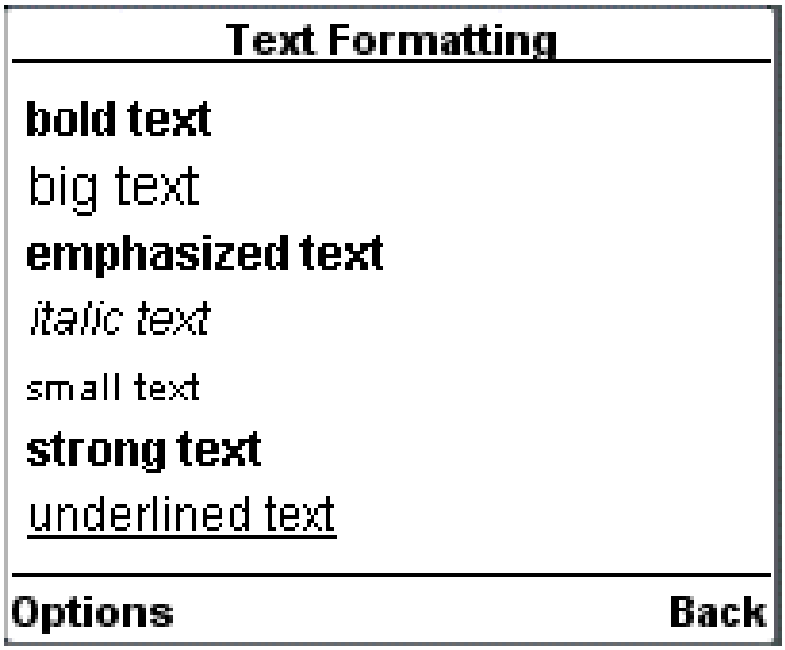
Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
```

```
<wml>

<card title="Text Formatting">
<p>
  <b>bold text</b><br/>
  <big>big text</big><br/>
  <em>emphasized text</em><br/>
  <i>italic text</i><br/>
  <small>small text</small><br/>
  <strong>strong text</strong><br/>
  <u>underlined text</u>
</p>
</card>

</wml>
```

It will produce the following result:



WML <small> Tag

The <small> element defines a small text.

Attributes

This element supports the following attributes:

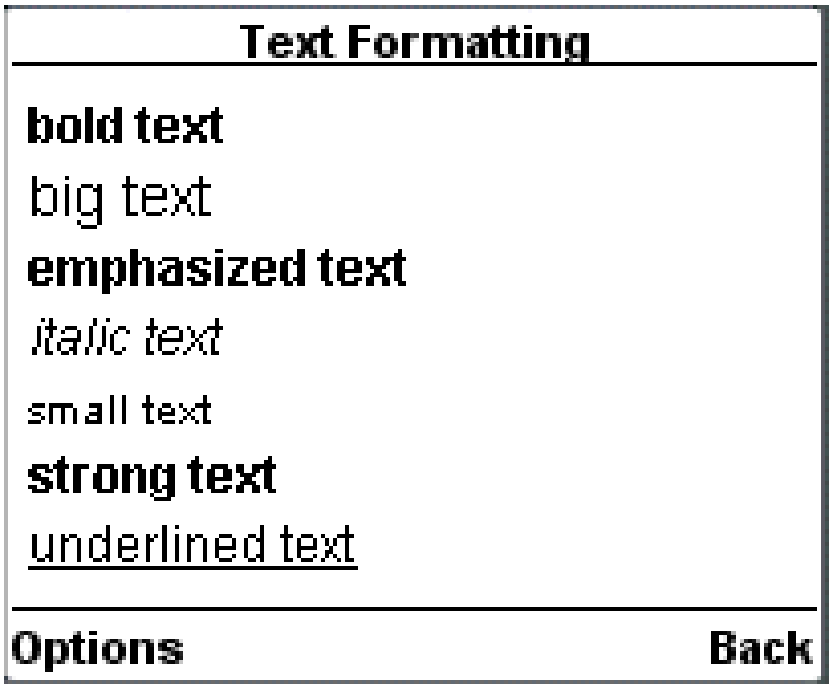
Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Text Formatting">
<p>
  <b>bold text</b><br/>
  <big>big text</big><br/>
  <em>emphasized text</em><br/>
  <i>italic text</i><br/>
  <small>small text</small><br/>
  <strong>strong text</strong><br/>
  <u>underlined text</u>
</p>
</card>

</wml>
```

It will produce the following result:



WML Tag

The element defines a strong text.

Attributes

This element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

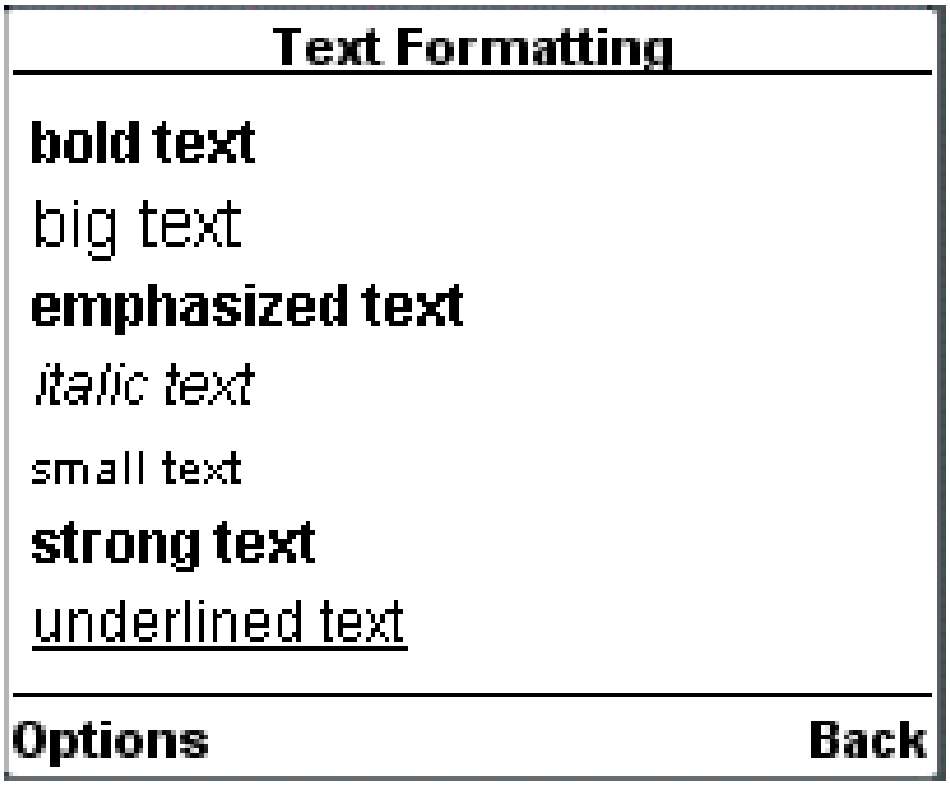
Example

The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
```

```
<wml>

<card title="Text Formatting">
<p>
  <b>bold text</b><br/>
  <big>big text</big><br/>
  <em>emphasized text</em><br/>
  <i>italic text</i><br/>
  <small>small text</small><br/>
  <strong>strong text</strong><br/>
  <u>underlined text</u>
</p>
</card>

</wml>
```

It will produce the following result:



WML <u> Tag

The <u> element defines an underlined text.

Attributes

This element supports the following attributes:

Attribute	Value	Description
xml:lang	language_code	Sets the language used in the element.
class	string	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Text Formatting">
<p>
  <b>bold text</b><br/>
  <big>big text</big><br/>
  <em>emphasized text</em><br/>
  <i>italic text</i><br/>
  <small>small text</small><br/>
  <strong>strong text</strong><br/>
  <u>underlined text</u>
</p>
</card>
```

```
</wml>
```

It will produce the following result:

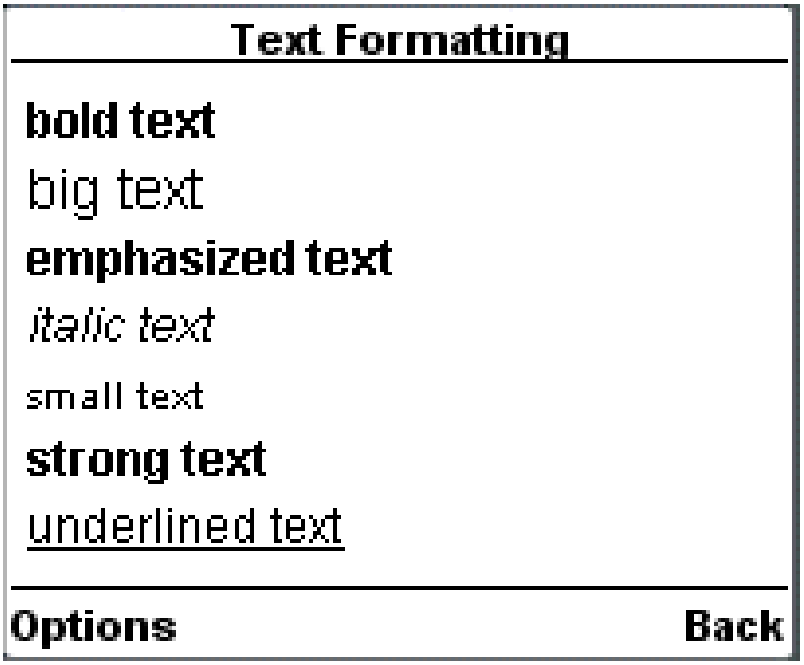


Image Elements

WML Elements	Purpose
<img>	Defines an image.

WML <img> Tag

The <img> element is used to include an image in a WAP card. WAP-enabled wireless devices only supported the Wireless Bitmap (WBMP) image format.

WBMP images can only contain two colors: black and white. The file extension of WBMP is ".wbmp" and the MIME type of WBMP is "image/vnd.wap.wbmp".

Attributes

This element supports the following attributes:

The <img> element supports the following attributes:

Attribute	Value	Description
align	top middle	Alignment of the image.

	bottom	
alt	alternative text	Sets an alternate text to be displayed if the image is not displayed.
height	px %	Height of the image in pixels or percentage. If you specify the value in pixels, the syntax is "140", instead of "140px".
hspace	px %	Sets white space to the left and right of the image. If you specify the value in pixels, the syntax is "140", instead of "140px".
localsrc	cdata	Sets an alternate representation for the image. If this attribute is set, the browser will use it instead of the "src" attribute.
src	image url	A path to wbmp image.
vspace	px %	Sets white space above and below the image. If you specify the value in pixels, the syntax is "140", instead of "140px".
width	px %	Sets the width of the image. If you specify the value in pixels, the syntax is "140", instead of "140px".
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

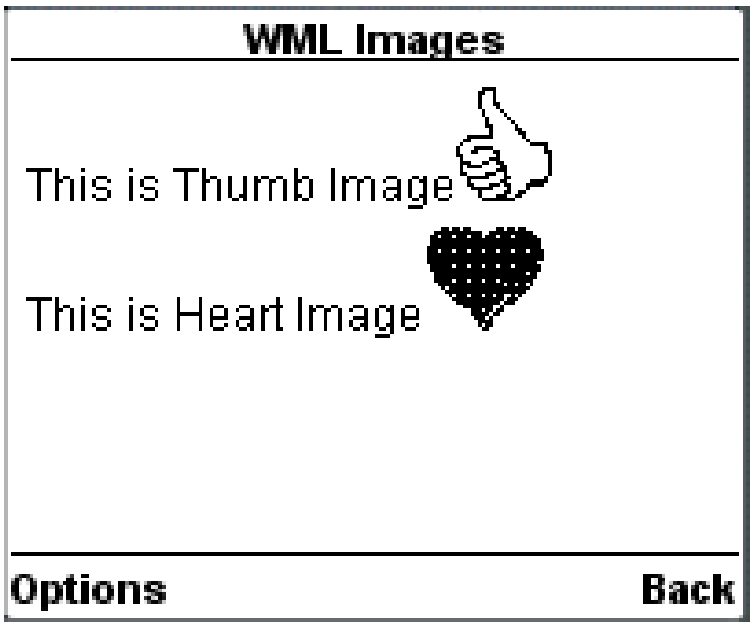
Example

The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="WML Images">
<p>
This is Thumb image
<img src="/images/thumb.wbmp" alt="Thumb Image"/>
```

```
</p>

<p>
This is Heart image
<img src="/images/heart.wbmp" alt="Heart Image"/>
</p>

</card>

</wml>
```

It will produce the following result:



Anchor Elements

WML Elements	Purpose
<a>	Defines an anchor.
<anchor>	Defines an anchor.

WML <a>Tag

The <a>... tag pair can also be used to create an anchor link and always a preferred way of creating links.

You can enclose Text or image inside <a>... tags.

Attributes

This element supports the following attributes:

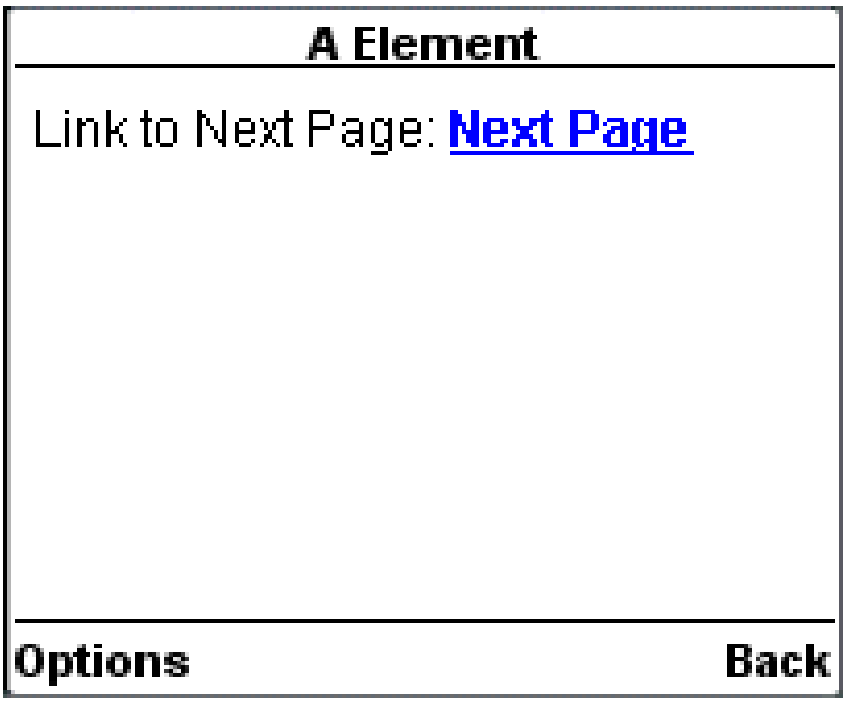
Attribute	Value	Description
href	URL	Defines URL of the liked page.
title	cdata	Defines a text identifying the link.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="A Element">
<p> Link to Next Page:
   <a href="nextpage.wml">Next Page</a>
</p>
</card>

</wml>
```

It will produce the following result:



WML <anchor> Tag

The <anchor>...</anchor> tag pair is used to create an anchor link. It is used together with other WML elements called <go/>, <refresh> or <prev/>. These elements are called task elements and tell WAP browsers what to do when a user selects the anchor link.

You can enclose Text or Image along with a task tag inside <anchor>...</anchor> tag pair.

Attributes

This element supports the following attributes:

Attribute	Value	Description
title	cdata	Defines a text identifying the link.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

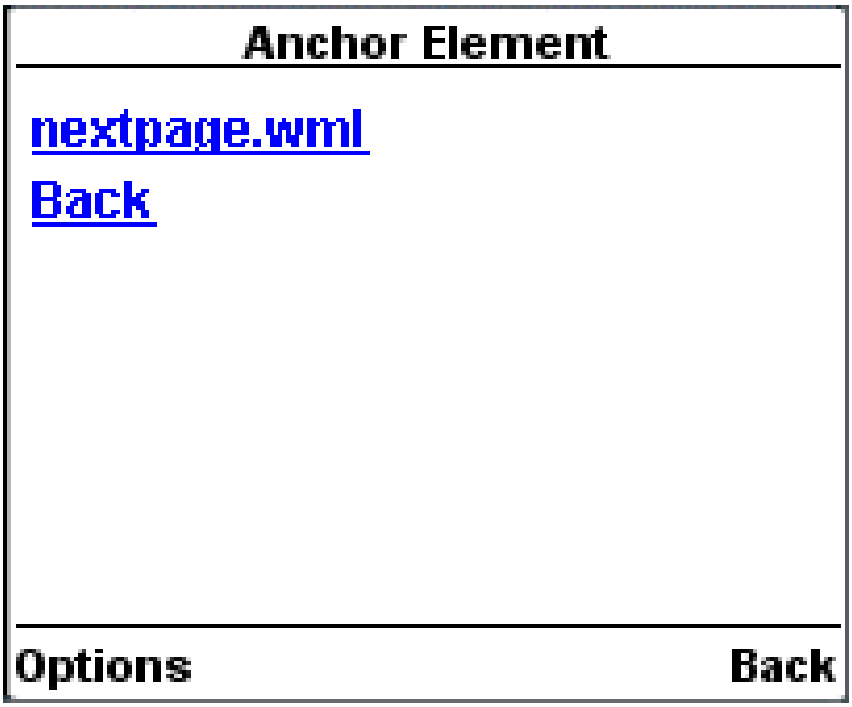
Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
```

```
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Anchor Element">
<p>
   <anchor>
      <go href="nextpage.wml"/>
   </anchor>
</p>
<p>
   <anchor>
      <prev/>
   </anchor>
</p>
</card>

</wml>
```

It will produce the following result:



Event Elements

WML Elements	Purpose
<do>	Defines a do event handler.
<onevent>	Defines an onevent event handler.
<postfield>	Defines a postfield event handler.
<ontimer>	Defines an ontimer event handler.
<onenterforward>	Defines an onenterforward handler.
<onenterbackward>	Defines an onenterbackward handler.
<onpick>	Defines an onpick event handler.

WML <do> Tag

The <do> tag can be used to activate a task when the user clicks on a word/phrase on the screen.

Attributes

This element supports the following attributes:

Attribute	Value	Description
name	text	Sets a name for the <do> element.
label	string	Sets a label for the <do> element.
type	accept prev help reset options delete unknown x-*	Defines the type of the <do> element.

	vnd.*	
value	number	Specifies the timer after which timer will be expired. Timeouts are specified in units of a tenth of a second.
class	class_data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

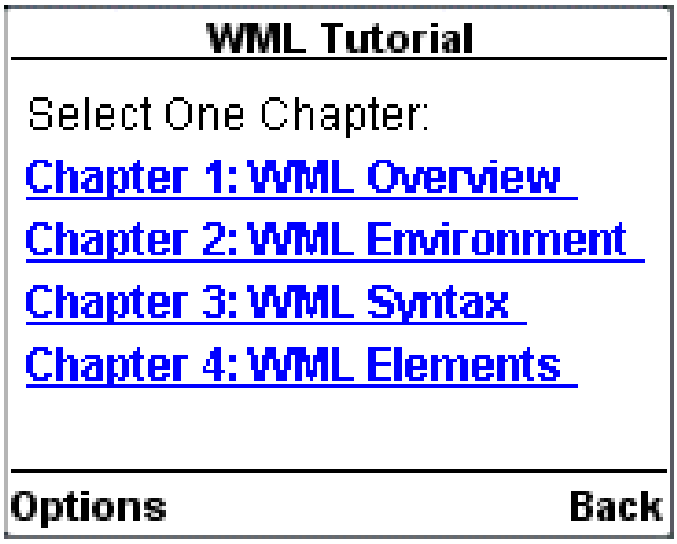
The following example shows the usage of <do> element along with <go> element:

```
<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"

"http://www.wapforum.org/DTD/wml13.dtd">


<wml>
  <template>
    <do name="main_menu" type="accept" label="Chapters">
      <go href="chapters"/>
    </do>
    <do name="menu_1" type="accept" label="Chapter 1">

      <go href="#chapter1"/>
    </do>
    <do name="menu_2" type="accept" label="Chapter 2">
      <go href="#chapter2"/>
    </do>
    <do name="menu_3" type="accept" label="Chapter 3">

      <go href="#chapter3"/>
    </do>
    <do name="menu_4" type="accept" label="Chapter 4">
      <go href="#chapter4"/>
    </do>
```

```
</template>

<card id="chapters" title="WML Tutorial">
  <p>
    Select One Chapter:<br/>
    <anchor>
      <go href="#chapter1"/>

      Chapter 1: WML Overview
    </anchor><br />

    <anchor>
      <go href="#chapter2"/>
      Chapter 2: WML Environment
    </anchor><br />

    <anchor>
      <go href="#chapter3"/>
      Chapter 3: WML Syntax
    </anchor><br />

    <anchor>
      <go href="#chapter4"/>

      Chapter 4: WML Elements
    </anchor><br />
  </p>
</card>

<card id="chapter1" title="WML Tutorial Ch1">
  <p>

    <em>Chapter 1: WML Introduction</em><br/>
```

```
      ...
    </p>
  </card>

  <card id="chapter2" title="WML Tutorial Ch2">

    <p>
      <em>Chapter 2: WML Environment</em><br/>
      ...
    </p>
  </card>

  <card id="chapter3" title="WML Tutorial Ch3">
    <p>
      <em>Chapter 3: WML Syntax</em><br/>
      ...
    </p>
  </card>

  <card id="chapter4" title="WML Tutorial Ch4">
    <p>
      <em>Chapter 4: WML Elements</em><br/>
      ...
    </p>

  </card>
</wml>
```

It will produce the following menu and now you can navigate through all the chapters:



WML <onevent> Tag

The <onevent>...</onevent> tags are used to create event handlers.

Attributes

This element supports the following attributes:

Attribute	Value	Description
type	onenterbackward onenterforward onpick ontimer	Defines a type of event occurred.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of <onevent> element. In this example, whenever you try to go back from second card to first card then **onenterbackward** occurs, which moves you to card number three. Copy and paste this program and try to play with it.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
```

```
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
<onevent type="onenterbackward">
  <go href="#card3"/>
</onevent>

<card id="card1" title="Card 1">
<p>
  <anchor>
     <go href="#card2"/>
     Go to card 2
  </anchor>
</p>

</card>
<card id="card2" title="Card 2">
<p>
   <anchor>
   <prev/>
      Going backwards
   </anchor>
</p>
</card>

<card id="card3" title="Card 3">
<p>
Hello World!
</p>
</card>
</wml>
```

WML <postfield> Tag

The <postfield> tag is used to post variables values to the server.

Attributes

This element supports the following attributes:

Attribute	Value	Description
name	string	Sets the name of the variable.
value	string	Sets the value of the variable.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

Following example shows how to submit three fields *name, age,* and *sex* to the server.

```
<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="card1" title="WML Form">
<p>
   Name: <input name="name" size="12"/>

   Sex : <select name="sex">
      <option value="male">Male</option>
      <option value="female">Female</option>
      </select>

   Age :  <input name="age" size="12" format="*N"/>
   <anchor>
      <go method="get" href="process.php">
```

```
          <postfield name="name" value="$(name)"/>
          <postfield name="age" value="$(age)"/>
          <postfield name="sex" value="$(sex)"/>
      </go>
      Submit Data
    </anchor>
</p>
</card>

</wml>
```

When you download the above code on your WAP device, it will provide you option to enter three fields *name, age,* and *sex* and one link *Submit Data*. You will enter three fields and then finally you will select *Submit Data* link to send entered data to the server.

WML <ontimer> Tag

The **ontimer** event is used to trigger an event after a given time period. Let's say, you want to display a message after 5 seconds of loading a card, then you can use this event to do so.

Here is the syntax to define an event handler for **ontimer** event:

```
<onevent type="ontimer">
   A task to be performed.
</onevent>
<timer value="50"/>
```

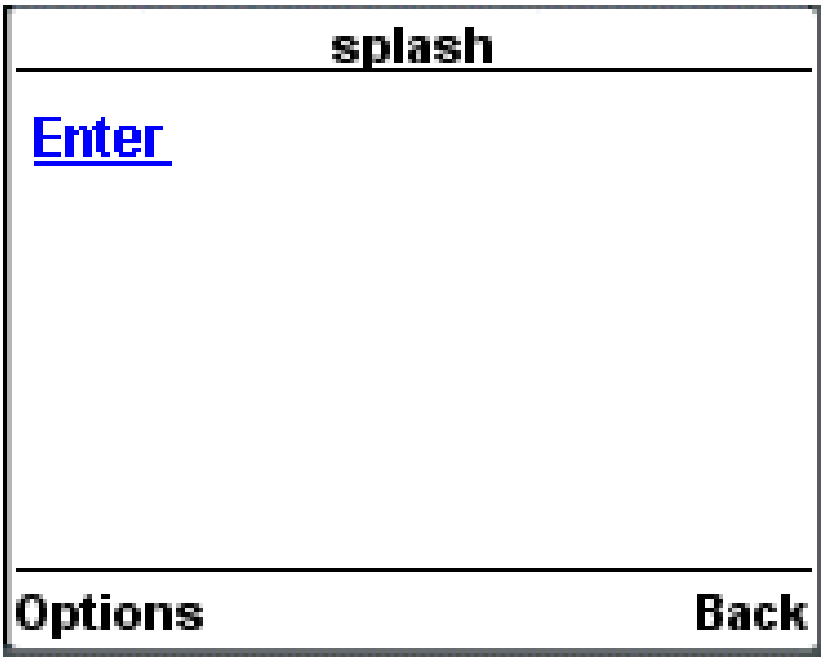
Example

The following example shows the usage of **ontimer** event along with <onevent> element:

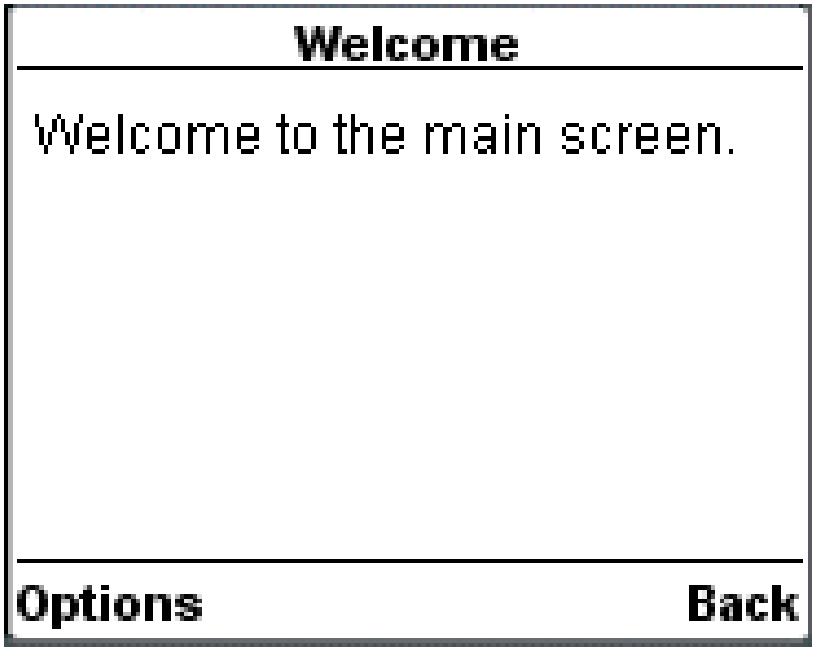
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
```

```
<card id="splash" title="splash">
  <onevent type="ontimer">
    <go href="#welcome"/>
  </onevent>
  <timer value="50"/>
<p>
  <a href="#welcome">Enter</a>
</p>
</card>

<card id="welcome" title="Welcome">
<p>
Welcome to the main screen.
</p>
</card>
</wml>
```

When you load this program, it shows you the following screen:



If you do not select given **Enter** option, then after 5 seconds you will be directed to **Welcome** page and following screen will be displayed automatically.



WML <onenterforward> Tag

The onenterforward event is triggered when a user goes to a card in the forward direction. For example, if you go to a card by entering the URL directly or by following an anchor link of which the action is <go>, the onenterforward event will be triggered and the WML code associated with the event will be executed.

The **onenterforward** event will be useful to you if you want to do something before a card is displayed. For example, you need the onenterforward event if you want to assign a value to a variable before a card is displayed.

Here is the syntax to define an event handler for **onenterforward** event:

```
<onevent type="onenterforward">
   A task to be performed.
</onevent>
```

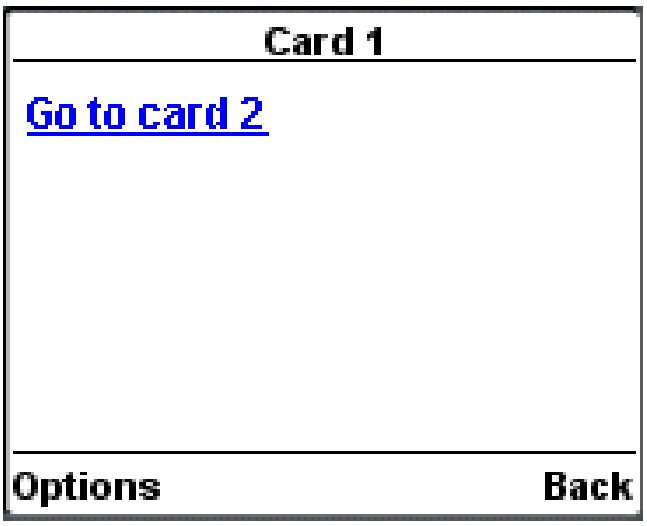
Example

Following is the example showing how **onenterbackward** event occurs whenever you try to go on second card from the first card and defined event handler takes you to card number three instead of card number 2. Copy and paste this program and try to play with it to understand **onenterforward** event type.

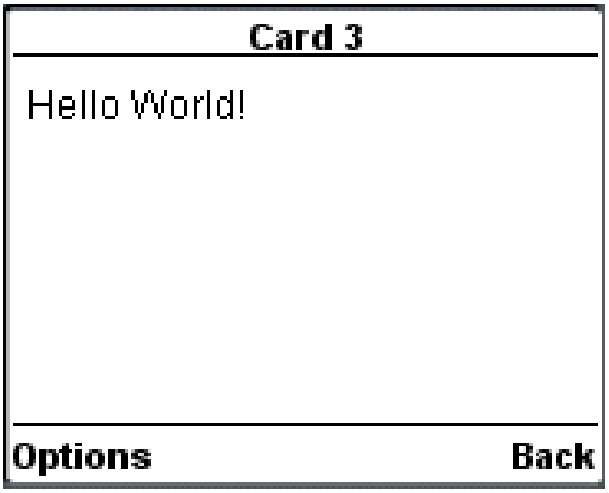
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
```

```
<wml>

<card id="card1" title="Card 1">
<p>
  <anchor>
     <go href="#card2"/>
     Go to card 2
  </anchor>
</p>
</card>

<card id="card2" title="Card 2">
<onevent type="onenterforward">
  <go href="#card3"/>

</onevent>
<p>
  This is card 2
</p>
</card>

<card id="card3" title="Card 3">
<p>
Hello World!
</p>

</card>
</wml>
```

When you load this program, you will get the following screen:



Now, press option **Go to card 2** to go to the second card, because of this **onenterforward,** event will occur and it will take on card number 3 instead of card number 2 and you will see the following screen:



WML <onenterbackward> Tag

This event occurs when the user hits a card by normal backward navigational means. That is, user presses the Back key on a later card and arrives back at this card in the history stack.

Here is the syntax to define an event handler for **onenterbackward** event:

```
<onevent type="onenterbackward">
   A task to be performed.
</onevent>
```

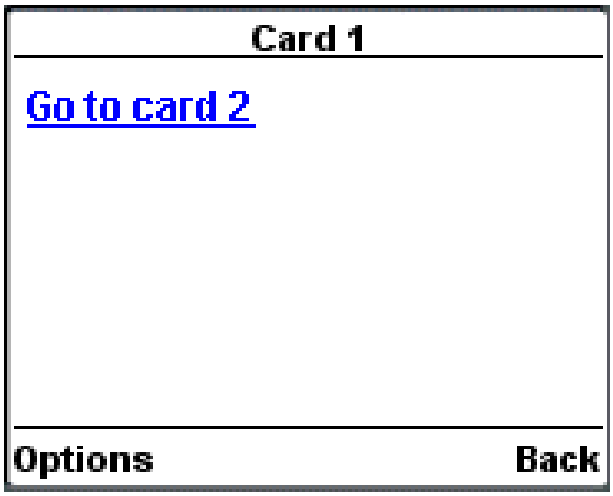
Example

Following is the example showing how **onenterbackward** event occurs whenever you try to go back from second card to first card and defined event handler takes you to card number three instead of card number 1. Copy and paste this program and try to play with it to understand **onenterbackward** event type.

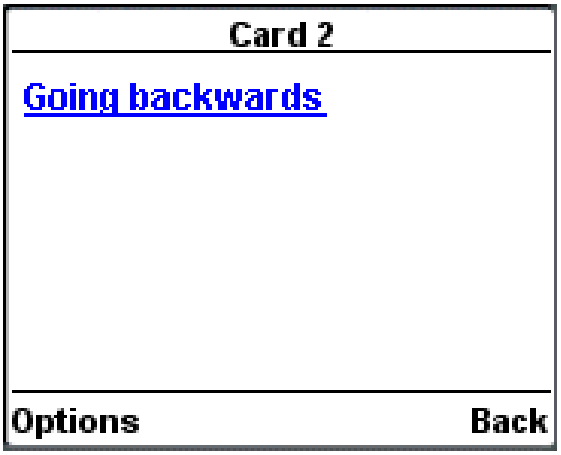
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
<card id="card1" title="Card 1">
<onevent type="onenterbackward">
  <go href="#card3"/>

</onevent>
<p>
  <anchor>
     <go href="#card2"/>
     Go to card 2
  </anchor>
</p>
</card>

<card id="card2" title="Card 2">
<p>
   <anchor>
   <prev/>
      Going backwards
   </anchor>
</p>
</card>

<card id="card3" title="Card 3">
<p>
```

```
Hello World!
</p>
</card>
</wml>
```

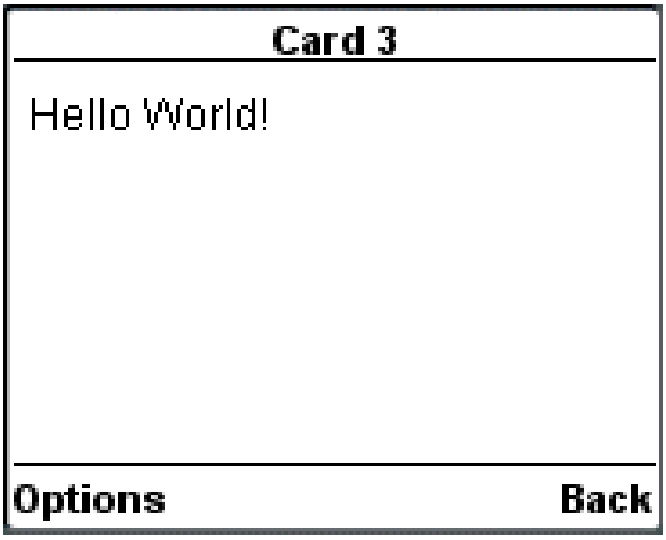
When you load this program you, will get the following screen:



Now, press option **Go to card 2** to go to the second card this will take you to the following screen:



Now, you are on second page. Now, when you try to go back on card number **onenterbackward** event is executed and it takes you on card number 3 instead of card number 2 and you see the following screen.



WML <onpick> Tag

The **onpick** attribute is a great shortcut if you are using a select menu. Instead of writing a lot of codes that allow the user to go to another card if an option is selected, you can simply place the destination into the **onpick** attribute. Here is a code fragment without the onpick attribute:

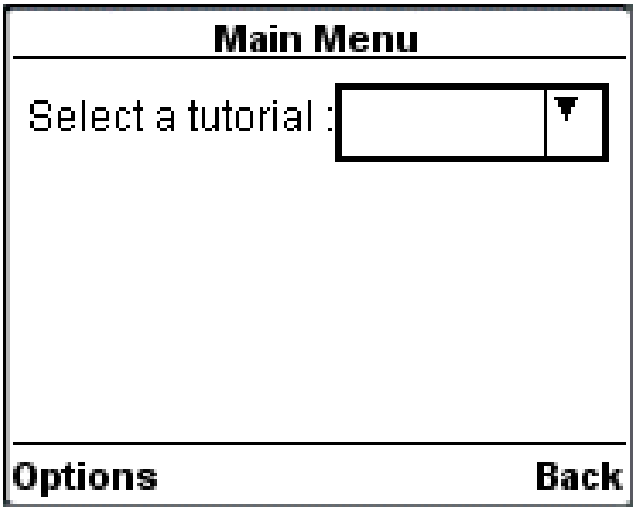
Example

The following example shows the usage of **onpick** attribute along with <option> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="tutorials" title="Main Menu">
<p>
  Select a tutorial :
  <select title="tutorials" name="selection_list">
    <option onpick="#xhtml">XHTML Tutorial</option>
    <option onpick="#wap">WAP Tutorial</option>
  </select>
</p>
```

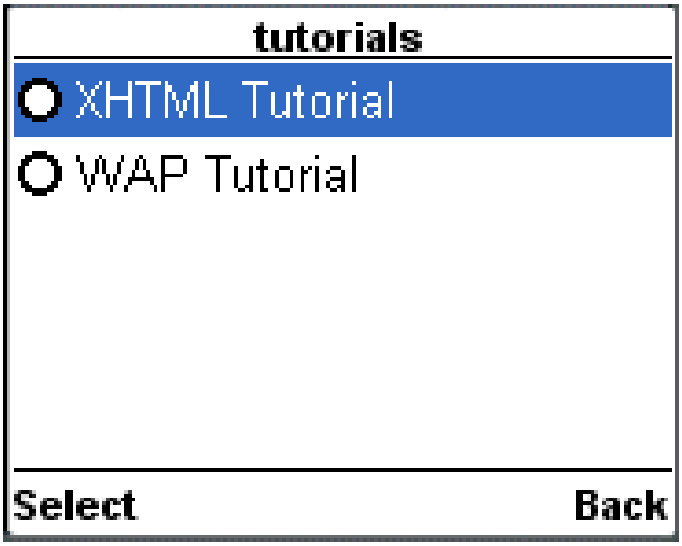
```
</card>
<card id="xhtml" title="XHTML Tutorial">
<p>
Go through our XHTML tutorial
</p>
</card>

<card id="wap" title="WAP Tutorial">
<p>
Go through our WAP tutorial
</p>
</card>
</wml>
```

When you load this program, it shows you the following screen:

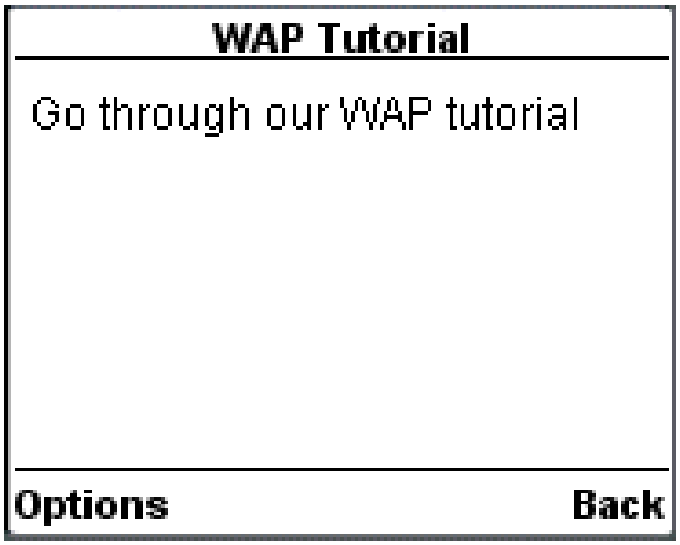


Now, highlight the dropdown box and select it. It will give you two options as follows:

When you load this program, it shows you the following screen:



Now, assume you select **WAP Tutorial** from the list, then it will display the following screen:



Task Elements

WML Elements	Purpose
<go>	Represents the action of switching to a new card.
<noop>	Says that nothing should be done.
<prev>	Represents the action of going back to the previous card.
<refresh>	Refreshes some specified card variables.

WML <go> Tag

The <go/> element represents a <go> task represents the action of going to a new card.

Attributes

This element supports the following attributes:

Attribute	Value	Description
href	URL	Gives the URL of the new card. Relative URLs are resolved relative to the current card.
method	get post	Specifies the method that should be used to fetch the deck. This must be one of the values get or post, corresponding to the GET and POST methods of HTTP. When using method="get", the data is sent as a request with? Data appended to the url. The method has a disadvantage that it can be used only for a limited amount of data, and if you send sensitive information it will be displayed on the screen and saved in the web server's logs. So do not use this method if you are sending password etc. With method="post", the data is sent as a request with the data sent in the body of the request. This method has no limit, and sensitive information is not visible in the URL.
sendreferer	true false	If set to true, the browser sends the URL of the current deck along with the request. This URL is sent as a relative URL if possible. The purpose of this is to allow servers to perform simple access control on decks, based on which decks are linking to them. For example, using HTTP, this attribute is sent in the HTTP Referer header.
accept-charset	charset_list	Specifies a comma- or space-separated list of character sets that can encode data sent to the server in a POST request. The default value is "unknown".
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of <go> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
   <anchor>
        Chapter 2 : <go href="chapter2.wml"/>
   </anchor>
</p>
</card>
</wml>
```

Another example showing how to upload data using Get Method:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
   <anchor>
      Using Get Method
      <go href="chapter2.wml?x=17&y=42" method="get"/>
   </anchor>
</p>
</card>
```

```
</wml>
```

Another example showing how to upload data using <setvar> element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="GO Element">
<p>
   <anchor>

      Using setvar:
       <go href="chapter2.wml">
           <setvar name="x" value="17"/>
           <setvar name="y" value="42"/>
       </go>
   </anchor>
</p>
</card>

</wml>
```

Another example showing how to upload data using <postfiled> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
```

```
<card title="GO Element">
<p>
   <anchor>
      Using setvar:
       <go href="chapter2.wml" method="get">
              <postfield name="x" value="17"/>
              <postfield name="y" value="42"/>

       </go>
   </anchor>
</p>
</card>

</wml>
```

WML <noop> Tag

The purpose of the <noop> task is to do nothing (no operation).

The only real use for this task is in connection with the templates.

Attributes

This element supports the following attributes:

Attribute	Value	Description
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Noop Element">
```

```
<p>
  <do type="prev" label="Back">
      <noop/>
  </do>
</p>
</card>

</wml>
```

WML <prev> Tag

The <prev> task represents the action of returning to the previously visited card on the history stack. When this action is performed, the top entry is removed from the history stack, and that card is displayed again, after any <setvar> variable assignments in the <prev> task have taken effect.

Attributes

This element supports the following attributes:

Attribute	Value	Description
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of <prev> element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Prev Element">
<p>
   <anchor>
        Previous Page :<prev/>
```

```
   </anchor>
</p>
</card>
</wml>
```

One situation where it can be useful to include variables in a <prev> task is a login page, which prompts for a username and password. In some situations, you may want to clear out the password field when returning to the login card, forcing the user to reenter it. This can be done with a construct such as:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Prev Element">
<p>
   <anchor>
        <prev>
           <setvar name="password" value=""/>
        </prev>
   </anchor>
</p>
</card>
</wml>
```

WML <refresh> Tag

The <refresh> task is the simplest task that actually does something. Its effect is simply to perform the variable assignments specified by its <setvar> elements, then redisplay the current card with the new values. The <go> and <prev> tasks perform the same action just before displaying the new card.

The <refresh> task is more often used to perform some sort of "reset" action on the card.

Attributes

This element supports the following attributes:

Attribute	Value	Description
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Referesh Element">
<p>
   <anchor>
         Refresh this page:
          <go href="test.wml"/>
          <refresh>
            <setvar name="x" value="100"/>
          </refresh>
   </anchor>
</p>
</card>

</wml>
```

Input Elements

WML Elements	Purpose
<input>	Defines an input field.

<select>	Defines a select group.
<option>	Defines an option in a selectable list.
<fieldset>	Defines a set of input fields.
<optgroup>	Defines an option group in a selectable list.

WML <input> Tag

The <input/> element is used to create input fields and input fields are used to obtain alphanumeric data from the users.

Attributes

This element supports the following attributes:

Attribute	Value	Description
name	text	The name of the variable that is set with the result of the user's input.
maxlength	number	Sets the maximum number of characters the user can enter in the field.
emptyok	true false	Sets whether the user can leave the input field blank or not. Default is "false."
format	A a N X x M m *f *nf*	Sets the data format for the input field. Default is "*M". A = uppercase alphabetic or punctuation characters. a = lowercase alphabetic or punctuation characters. N = numeric characters. X = uppercase characters. x = lowercase characters. M = all characters. m = all characters. *f* = Any number of characters. Replace the *f* with one of the letters above to specify what characters the user can enter. *nf* = Replace the *n* with a number from 1 to 9 to specify the number of characters the user can enter. Replace the *f* with one of the letters above to specify what characters the user can enter.
size	number	Sets the width of the input field.

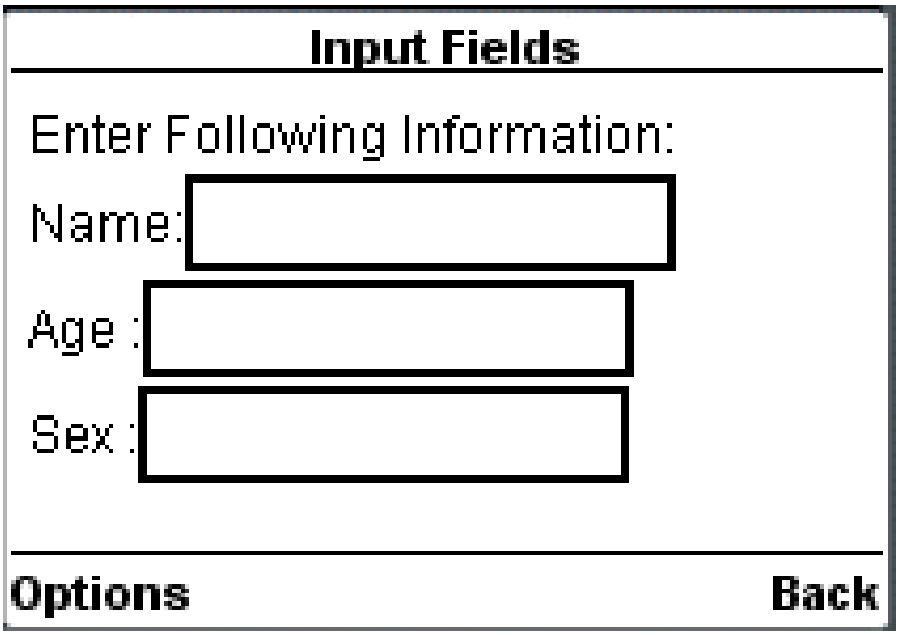
tabindex	number	Sets the tabbing position for the select element.
title	text	Sets a title for the list.
type	text password	Indicates the type of the input field. The default value is "text". Password field is used to take password for authentication purpose.
value	text	Sets the default value of the variable in the "name" attribute.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Input Fields">
<p> Enter Following Information:<br/>
 Name: <input name="name" size="12"/>
 Age :  <input name="age" size="12" format="*N"/>
 Sex :  <input name="sex" size="12"/>
</p>
</card>

</wml>
```

It will produce the following screen to enter the required information:



WML <select> Tag

The <select>...</select> WML elements are used to define a selection list and the <option>...</option> tags are used to define an item in a selection list. Items are presented as radio buttons in some WAP browsers. The <option>...</option> tag pair should be enclosed within the <select>...</select> tags.

Attributes

This element supports the following attributes:

Attribute	Value	Description
iname	text	Names the variable that is set with the index result of the selection.
ivalue	text	Sets the pre-selected option element.
multiple	true false	Sets whether multiple items can be selected. Default is "false."
name	text	Names the variable that is set with the result of the selection.
tabindex	number	Sets the tabbing position for the select element.

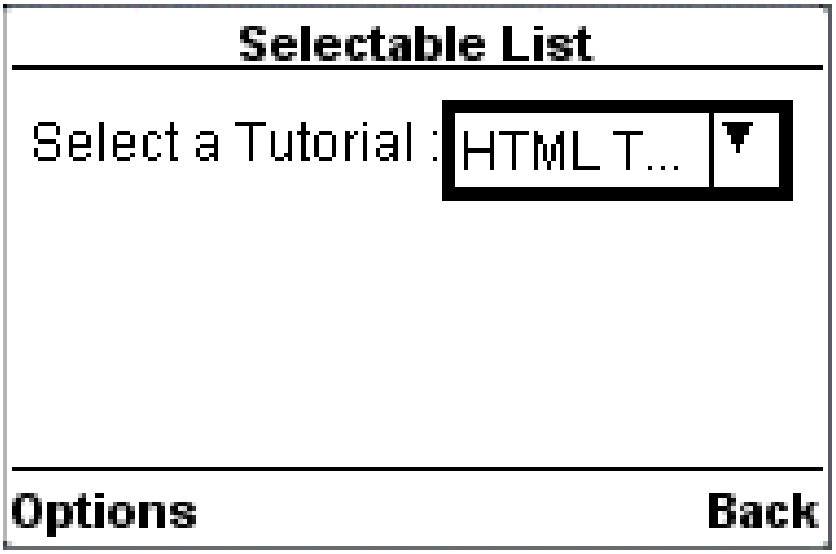
title	text	Sets a title for the list.
value	text	Sets the default value of the variable in the "name" attribute.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

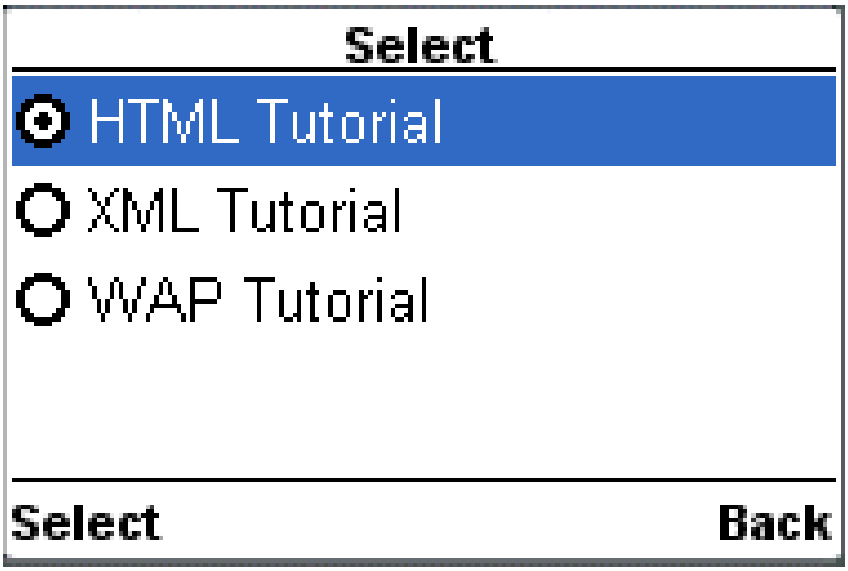
The following example shows the usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Selectable List">
<p> Select a Tutorial :
 <select>
  <option value="htm">HTML Tutorial</option>
  <option value="xml">XML Tutorial</option>
  <option value="wap">WAP Tutorial</option>
 </select>
</p>
</card>

</wml>
```

When you will load this program, it will show you the following screen:



Once you highlight and enter on the options, it will display the following screen:

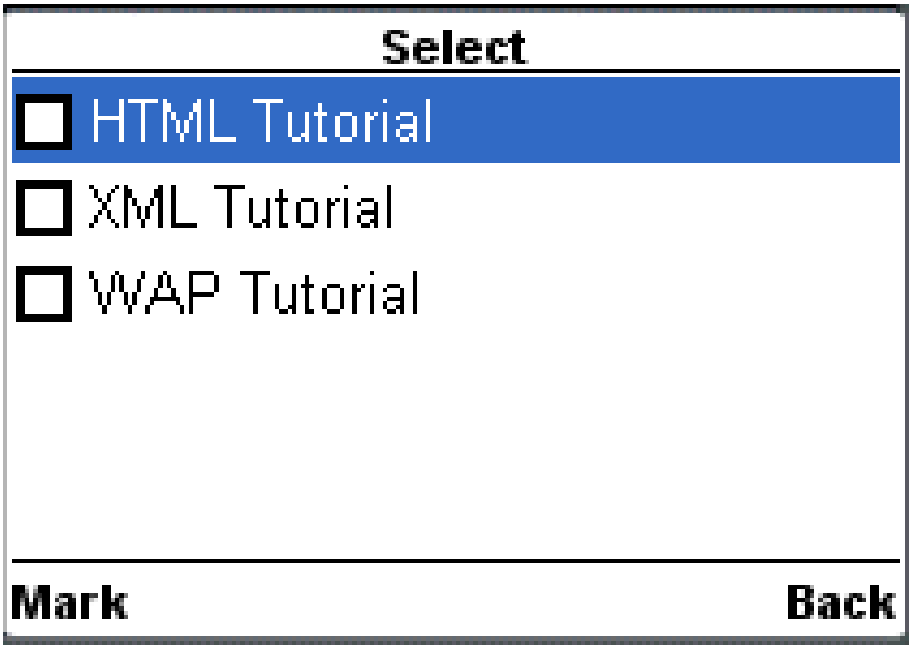


You want to provide option to select multiple options, then set *multiple* attribute to *true* as follows:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Selectable List">
```

```
<p> Select a Tutorial :
 <select multiple="true">
  <option value="htm">HTML Tutorial</option>
  <option value="xml">XML Tutorial</option>
  <option value="wap">WAP Tutorial</option>
 </select>
</p>
</card>

</wml>
```

This will give you a screen to select multiple options as follows:



WML <option> Tag

The <option>...</option> tags are used to define an item in a selection list. Items are presented as radio buttons in some WAP browsers. The <option>...</option> tag pair should be enclosed within the <select>...</select> tags.

Attributes

This element supports the following attributes:

Attribute	Value	Description

onpick	url	Sets what is going to happen when a user selects an item.
title	text	Sets a title for the option.
value	text	Sets the default value of the variable in the "name" attribute.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

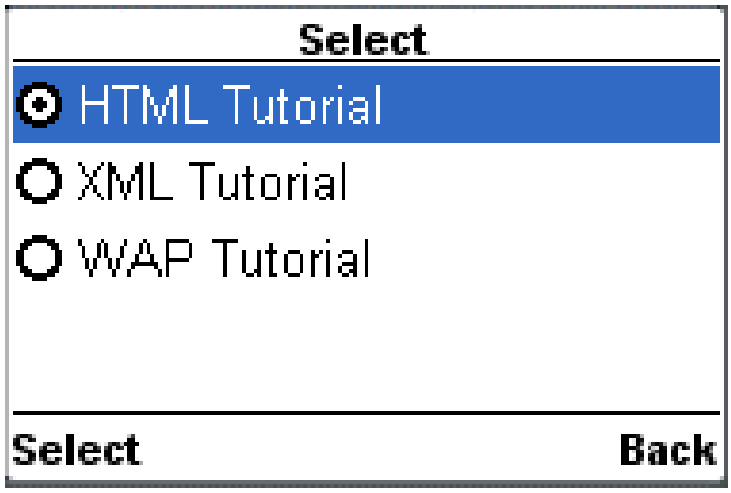
Example

The following example shows the usage of this element:

```
<?xml version="1.0"?>

<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"

"http://www.wapforum.org/DTD/wml12.dtd">


<wml>


<card title="Selectable List">
<p> Select a Tutorial :
 <select>
  <option value="htm">HTML Tutorial</option>
  <option value="xml">XML Tutorial</option>
  <option value="wap">WAP Tutorial</option>
 </select>
</p>
</card>


</wml>
```

When you will load this program, it will show you the following screen:

Once you highlight and enter on the options, it will display the following screen:



WML <fieldset> Tag

The <fieldset/> element is used to group various input fields or selectable lists.

Attributes

This element supports the following attributes:

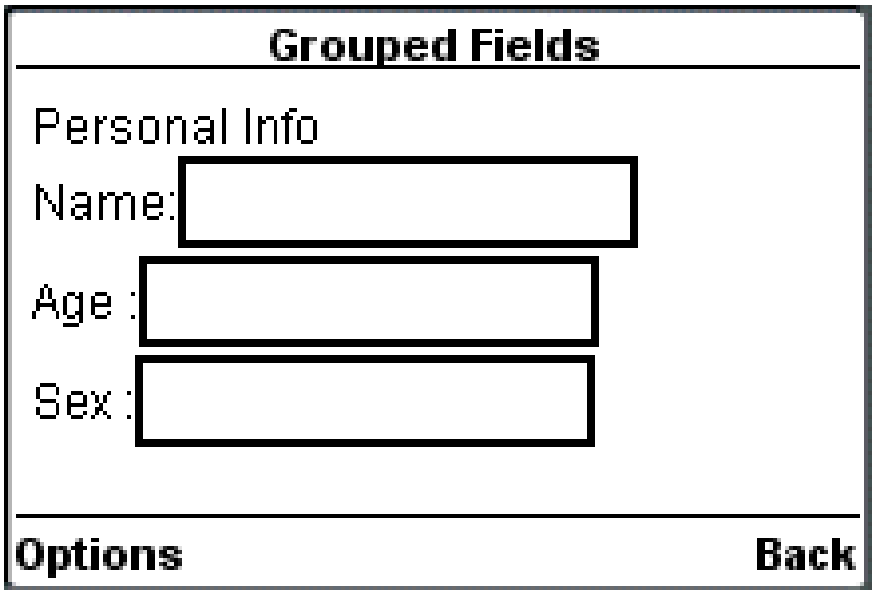
Attribute	Value	Description
title	text	Sets a title for the list.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following example shows the usage of this element.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Grouped Fields">
<p>
<fieldset title="Personal Info">
 Name: <input name="name" size="12"/>
 Age :  <input name="age" size="12" format="*N"/>
 Sex :  <input name="sex" size="12"/>
</fieldset>
</p>
</card>

</wml>
```

It will produce the following screen to enter the required information. The result may differ from browser to browser.



WML <optgroup> Tag

The <optgroup/> element is used to group various options together inside a selectable list.

Attributes

This element supports the following attributes:

Attribute	Value	Description
title	text	Sets a title for the list.
xml:lang	language_code	Sets the language used in the element.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

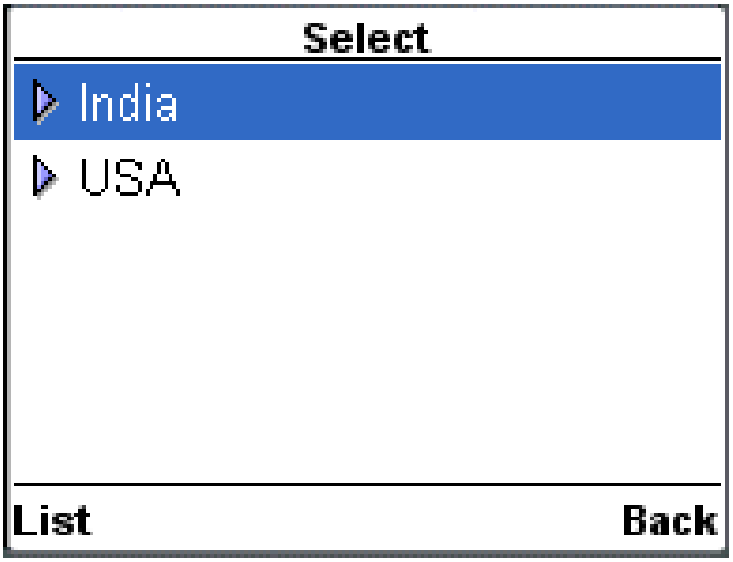
Example

The following example shows the usage of this element:

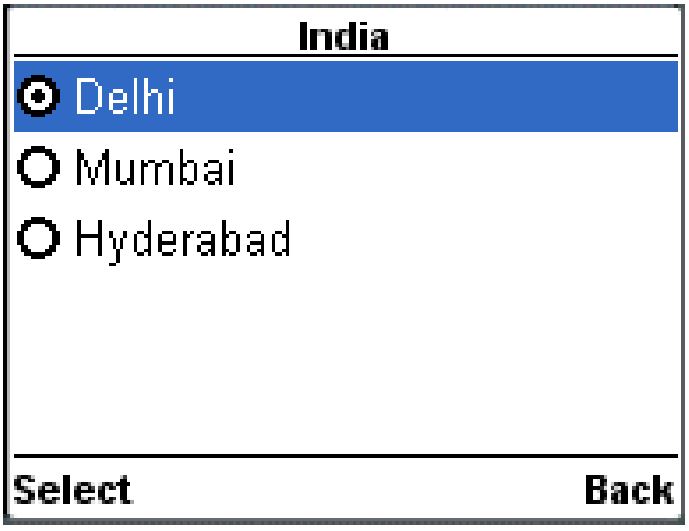
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card title="Selectable List">
<p>
 <select>
   <optgroup title="India">
    <option value="delhi">Delhi</option>
    <option value="mumbai">Mumbai</option>
    <option value="hyderabad">Hyderabad</option>
   </optgroup>

   <optgroup title="USA">
    <option value="ohio">Ohio</option>
    <option value="maryland">Maryland</option>
    <option value="washington">Washingtone</option>
```

```
   </optgroup>
 </select>
</p>
</card>

</wml>
```

When a user loads the above code, it will produce two options to be selected:



When a user selects any of the options, it will show the final options to be selected. If a user selects India, then it will show you the following options:



Variable Elements

WML Elements	Purpose

<setvar>	Defines and sets a variable.
<timer>	Defines a timer.

WML <setvar> Tag

The <setvar> element is used as a result of the user executing some task. The >setvar> element can be used to set a variable's state within the following elements: <go>, <prev>, and <refresh>.

Attributes

This element supports the following attributes:

Attribute	Value	Description
name	string	Sets the name of the variable.
value	string	Sets the value of the variable.
class	class data	Sets a class name for the element.
id	element ID	A unique ID for the element.

Example

The following element would create a variable named '*a*' with a value of 1000:

```
<setvar name="a" value="1000"/>
```

Following is another example showing usage of this element:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>
<card>
<p>
 <anchor>
  Go to next chapter
  <go href="#chapter2">
   <setvar name="x" value="30"/>
  </go>
```

```
 </anchor>
</p>
</card>


</wml>
```

WML <timer> Tag

A timer is declared inside a WML card with the <timer> element. It must follow the <onevent> elements if they are present. (If there are no <onevent> elements, the <timer> must be the first element inside the <card>.) No more than one <timer> may be present in a card.

Attributes

This element supports the following attributes:

Attribute	Value	Description
name	text	Sets a name for the element.
value	number	Specifies the timer after which timer will be expired. Timeouts are specified in units of a tenth of a second.
class	class_data	Sets a class name for the element.
id	element ID	A unique ID for the element.

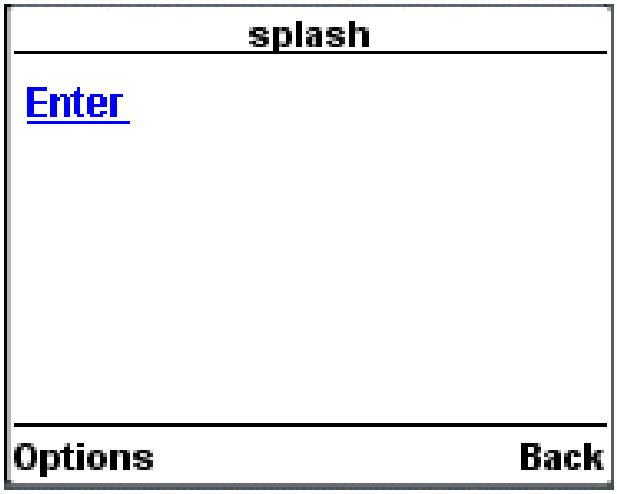
Example

The following example shows the usage of <timer> element:

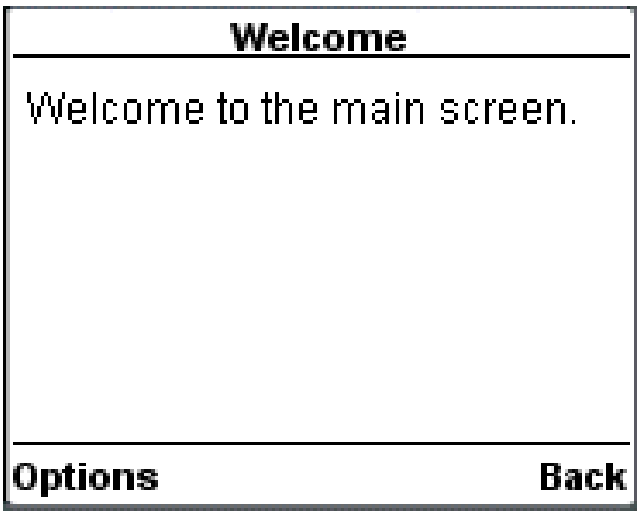
```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">

<wml>

<card id="splash" title="splash">
  <onevent type="ontimer">
    <go href="#welcome"/>
```

```
  </onevent>
  <timer value="50"/>
<p>
  <a href="#welcome">Enter</a>
</p>
</card>

<card id="welcome" title="Welcome">
<p>
Welcome to the main screen.
</p>
</card>

</wml>
```

When you load this program, it will display the following screen:



If you do not select given **Enter** option, then after 5 seconds you will be directed to the **Welcome** page and the following screen will be displayed automatically:



23. WML – WAP EMULATORS

Instead of installing an entire WAP SDK, you can install a WML emulator. An emulator simply lets you view the contents of your WML files as they would be seen on the screen of a WAP-enabled device.

While the emulators do a great job, they are not perfect. Try a few different ones, and you will quickly decide, which you like the most. When the time will come to develop a real (commercial) WAP site, you need to do a lot more testing, first with other SDKs/emulators and then with all the WAP-enabled devices that you plan to support.

The following lists some of the WAP emulators that are freely available:

- **Klondike WAP Browser**: This is produced by Apache Software. Klondike looks a lot like a Web browser and is therefore very easy to use for beginners. You can access local WML files easily. It also supports drag-and drop, making local file use very easy.

- **Yospace**: This is produced by Yospace. WAP developers can use the desktop edition of the emulator to preview WAP applications from their desktop, safe with the knowledge that the emulator provides a reasonably faithful reproduction of the actual handset products.

- **Ericsson R380 Emulator**: This is produced by Ericsson. The R380 WAP emulator is intended to be used to test WML applications developed for the WAP browser in the Ericsson R380. The emulator contains the WAP browser and WAP settings functionality that can be found in the R380.

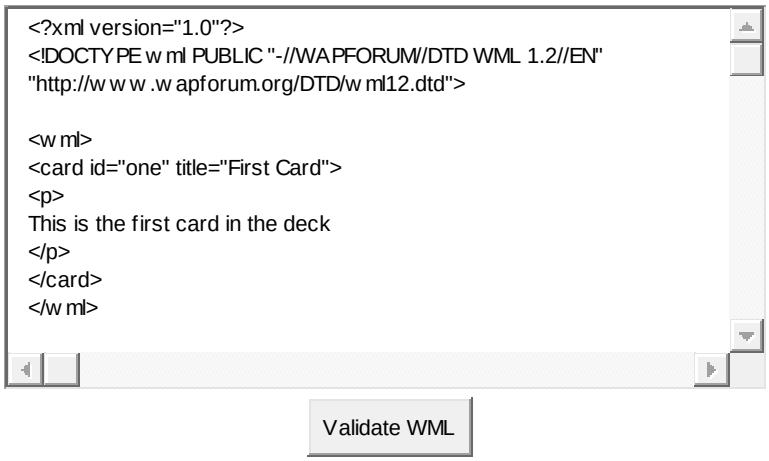
- **WinWAP**: This is produced by Slob-Trot Software. WinWAP is a WML browser that works on any computer with 32-bit Windows installed. You can browse WML files locally from your hard drive or the Internet with HTTP (as with your normal Web browser).

- **Nokia WAP simulator**: This is produced by Nokia and fully loaded with almost all functionalities. Try this one.

24. WML – VALIDATOR

Validate WML Content

Copy and paste WML content in the following box and then click *Validate WML* to see the result at the bottom of this page:



Validate WML File

Type your WML page URL and then click *Validate WML* to see the result at the bottom of this page:

File Name: