Simple Entity EJB - xDoclet, MyEclipse, Jboss and PostgreSql, MySql

Creation and testing of a first Entity Bean using MyEcplise, Jboss and xDoclet.

General

Author:
Sebastian Hennebrüder
http://www.laliluna.de/tutorial.html – Tutorials for Struts, EJB, xdoclet and eclipse.

Date:
Revised
January, 7th 2005
Initial Version
November, 1st 2004

Development Tools
Eclipse 3.x
MyEclipse plugin 3.8
(A cheap and quite powerful Extension to Eclipse to develop Web Applications and EJB (J2EE) Applications. I think that there is a test version availalable at MyEclipse.)

Application Server
Jboss 3.2.5
PDF-Version des Tutorials:
http://www.laliluna.de/assets/tutorials/xDoclet_jboss_first_EJB.pdf

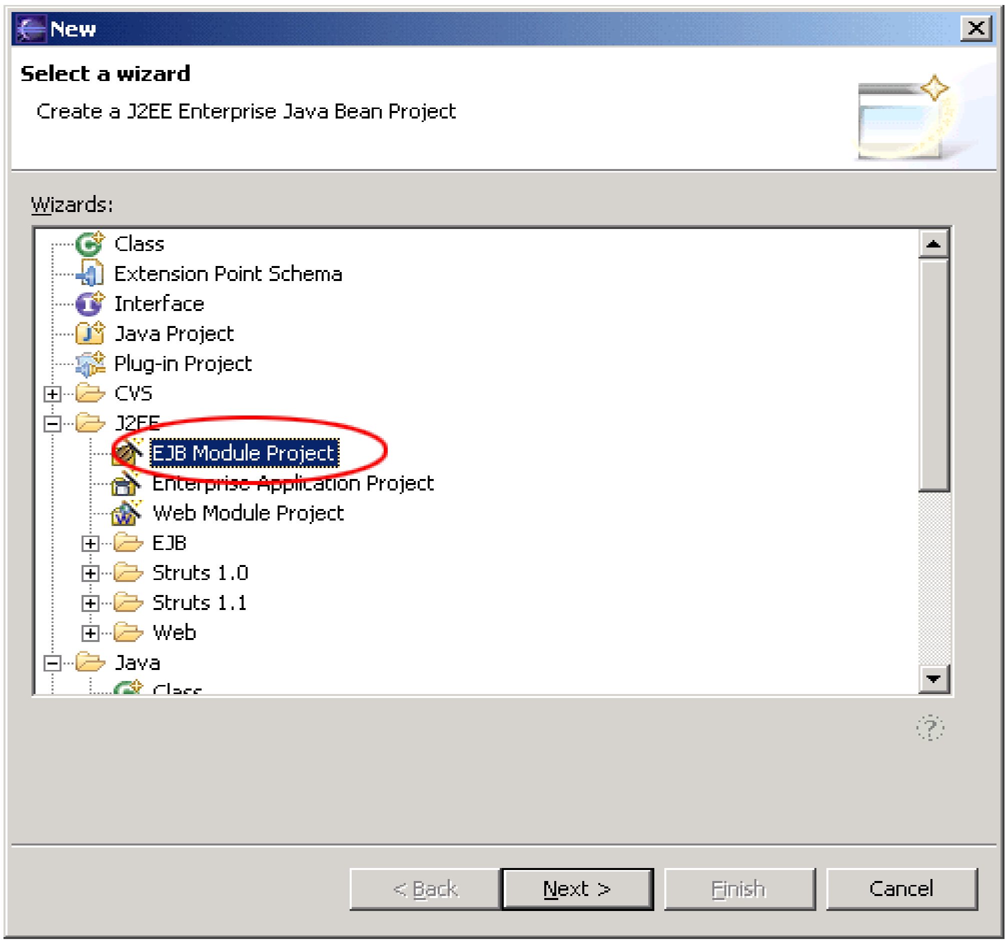
Introduction

If you do not have experinces with development of enterprice java beans with Eclipse, Jboss and xDoclet, this tutorial will help you to start.

Table of Contents

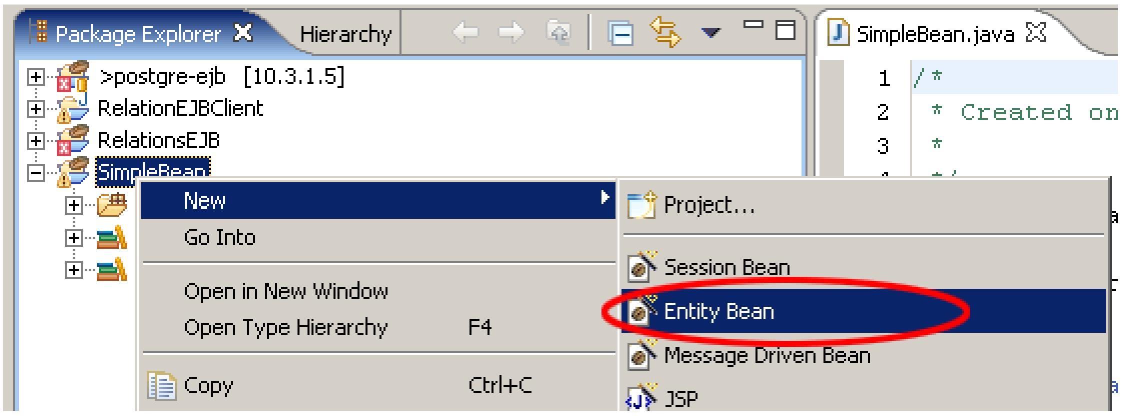
Create an EJB Module Projects

Create a new EJB Module Project. Right click on the package explorer or with shortcut „Strg + n“.



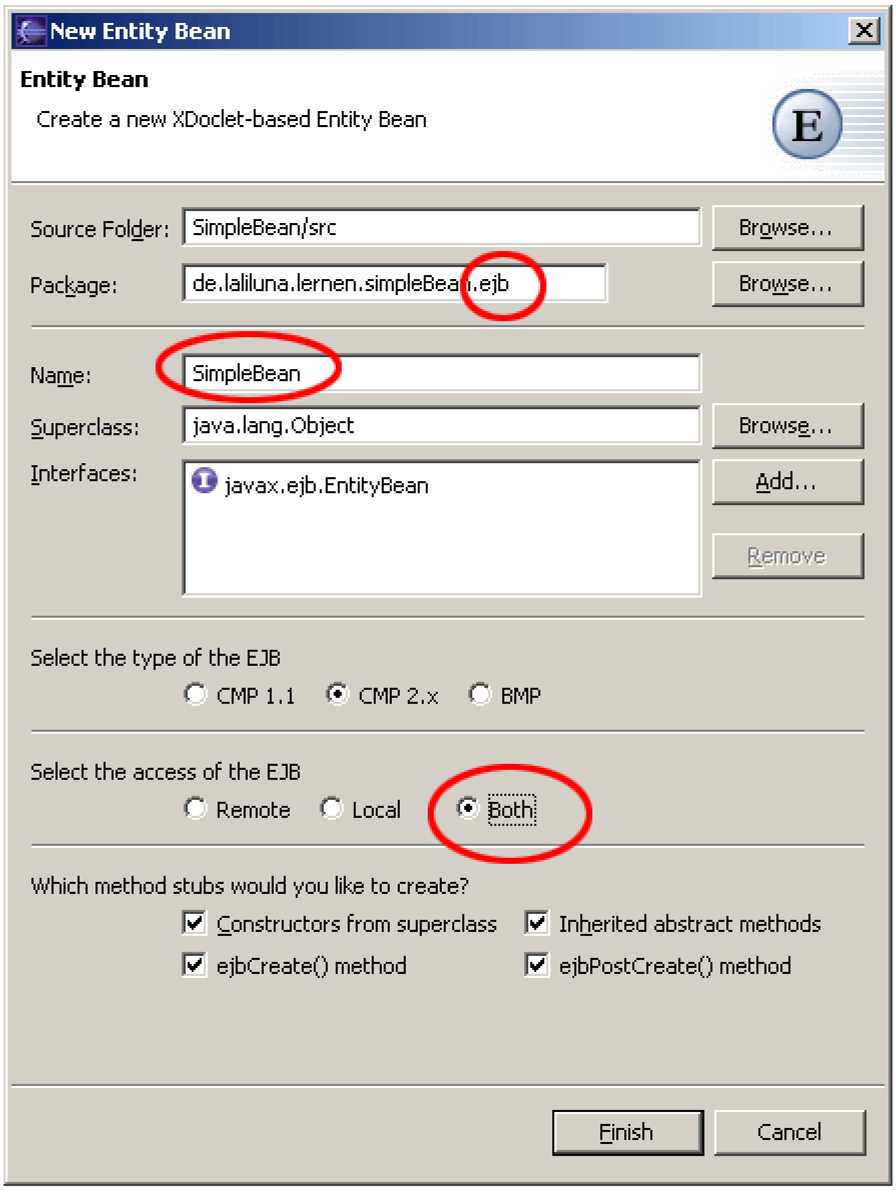
Create an EJB Class

Within the project create a new EJB.



- Make sure that the package ends with ".ejb" else xDoclet will not work!

Now you should see the generated source code.


New Entity Bean

Entity Bean

Create a new XDoclet-based Entity Bean

Source Folder: SimpleBean/src

Package: de.laliluna.lernen.simpleBean.ejb

Name: SimpleBean

Superclass: java.lang.Object

Interfaces: javax.ejb.EntityBean

Select the type of the EJB

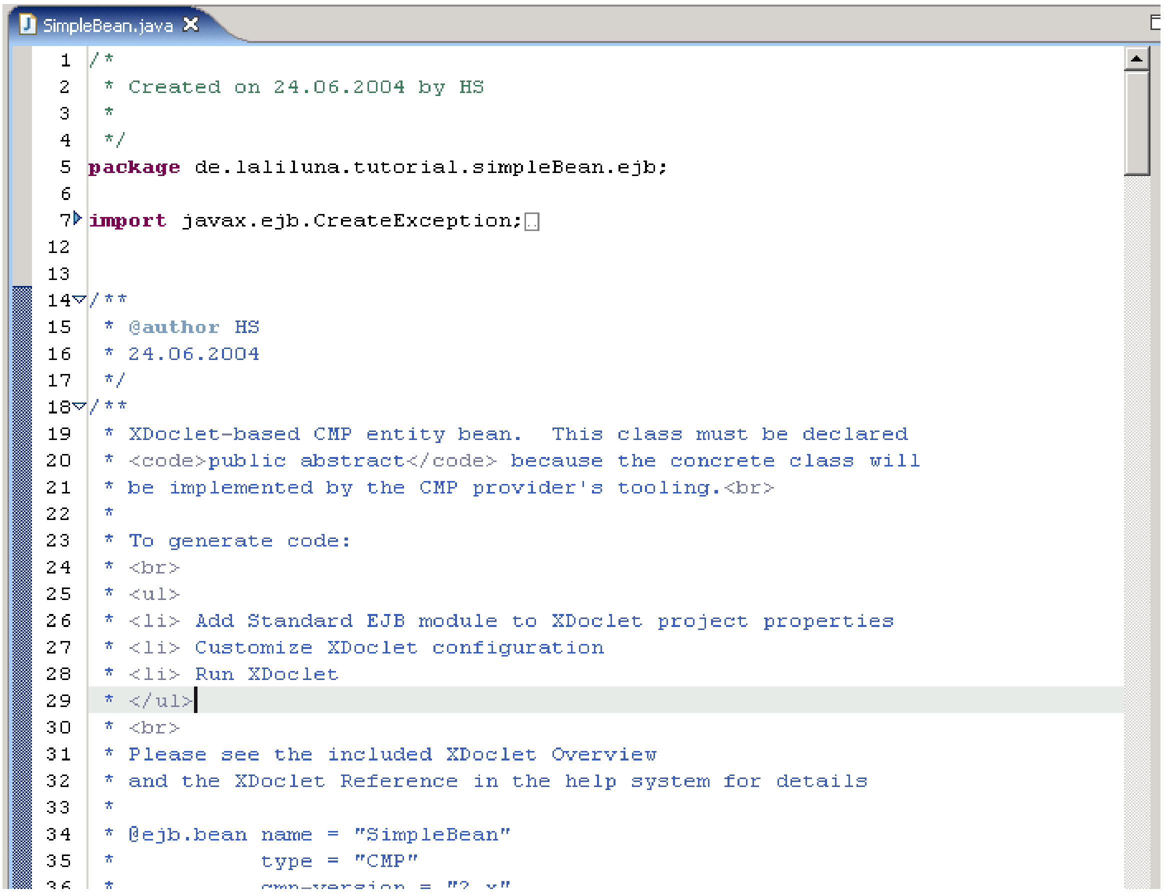
CMP 1.1 [✓ CMP 2.x] BMP

Select the access of the EJB

Remote Local [✓ Both]

Which method stubs would you like to create?

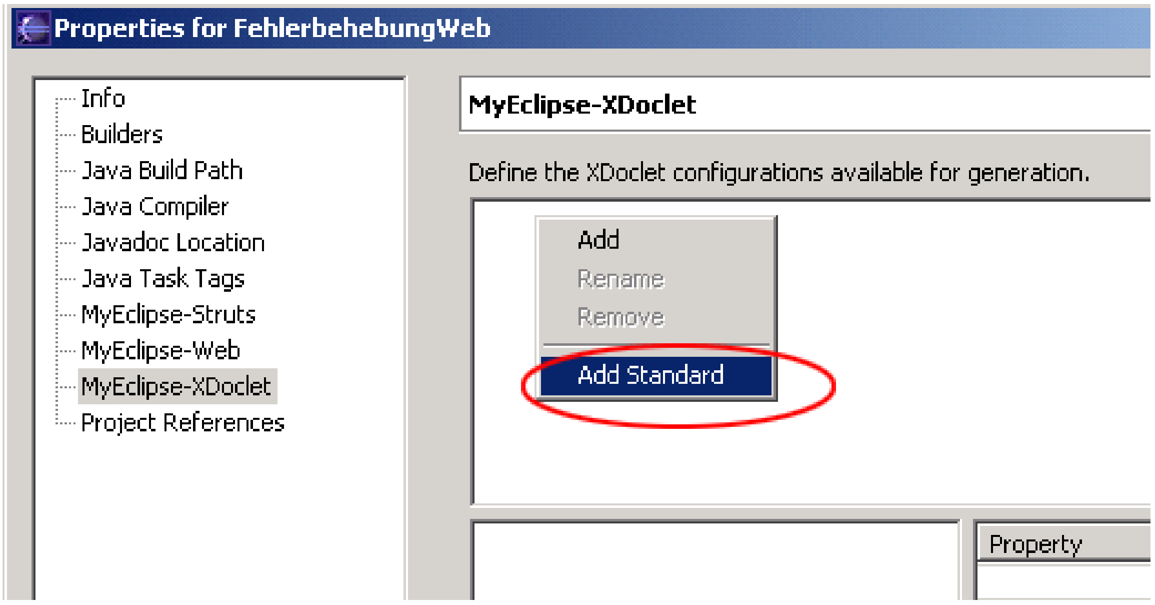
- [✓] Constructors from superclass
- [✓] Inherited abstract methods
- [✓] ejbCreate() method
- [✓] ejbPostCreate() method



```
/*
 * Created on 24.06.2004 by HS
 *
 */
package de.laliluna.tutorial.simpleBean.ejb;

import javax.ejb.CreateException;


/**
 * @author HS
 * 24.06.2004
 */
/**
 * XDoclet-based CMP entity bean.  This class must be declared
 * <code>public abstract</code> because the concrete class will
 * be implemented by the CMP provider's tooling.<br>
 *
 * To generate code:
 * <br>
 * <ul>
 * <li> Add Standard EJB module to XDoclet project properties
 * <li> Customize XDoclet configuration
 * <li> Run XDoclet
 * </ul>
 * <br>
 * Please see the included XDoclet Overview
 * and the XDoclet Reference in the help system for details
 *
 * @ejb.bean name = "SimpleBean"
 *           type = "CMP"
```

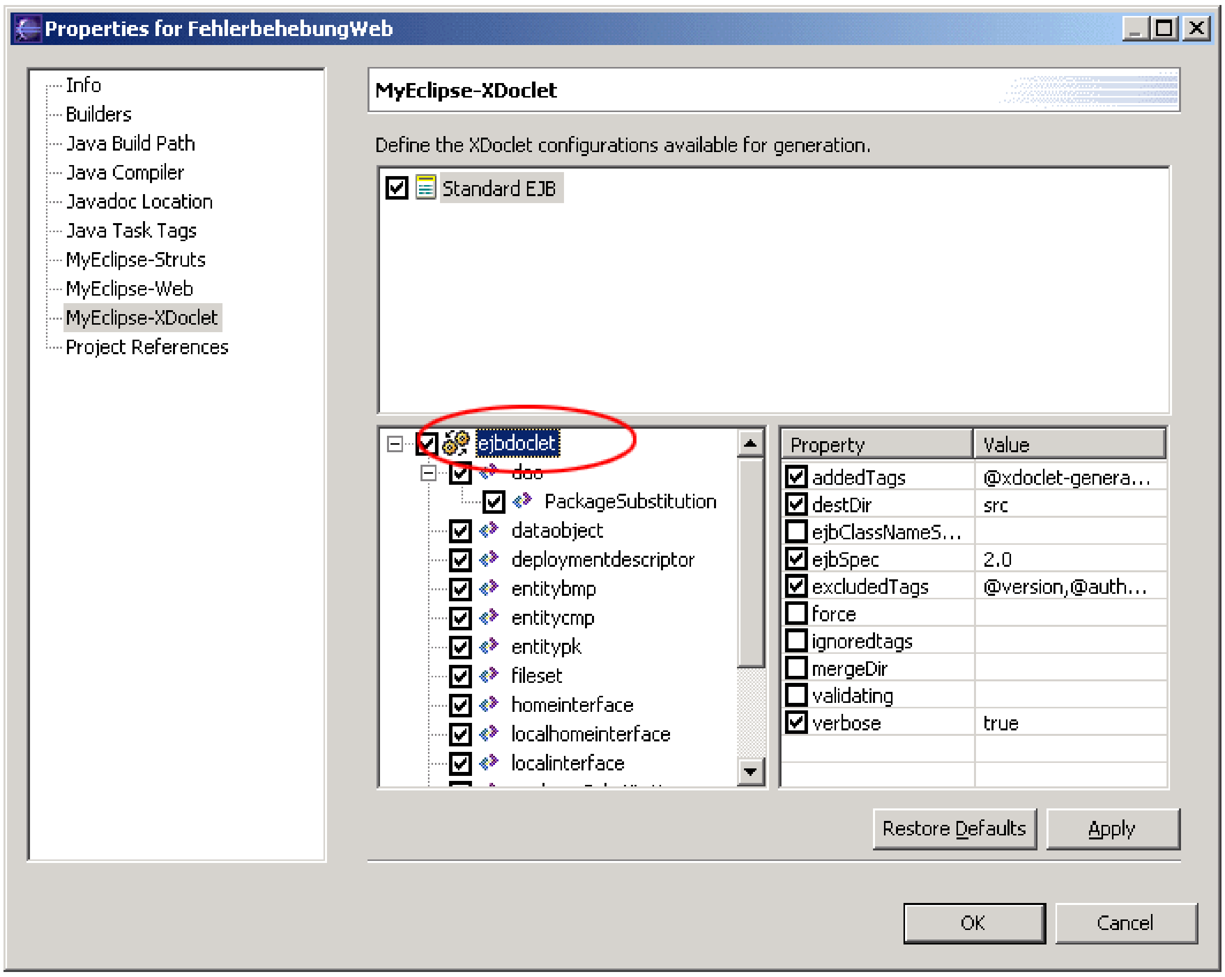
Add xDoclet functionality

- Open the project properties
- Choose "MyEclipse-xDoclet"
- Right click in the right upper window and choose „Add Standard“.
- Selet Standard EJB and OK



Define the XDoclet configurations available for generation.

Click on "Standard EJB" in the right upper window.


Properties for FehlerbehebungWeb

- Info
- Builders
- Java Build Path
- Java Compiler
- Javadoc Location
- Java Task Tags
- MyEclipse-Struts
- MyEclipse-Web
- MyEclipse-XDoclet
- Project References

MyEclipse-XDoclet

Define the XDoclet configurations available for generation.

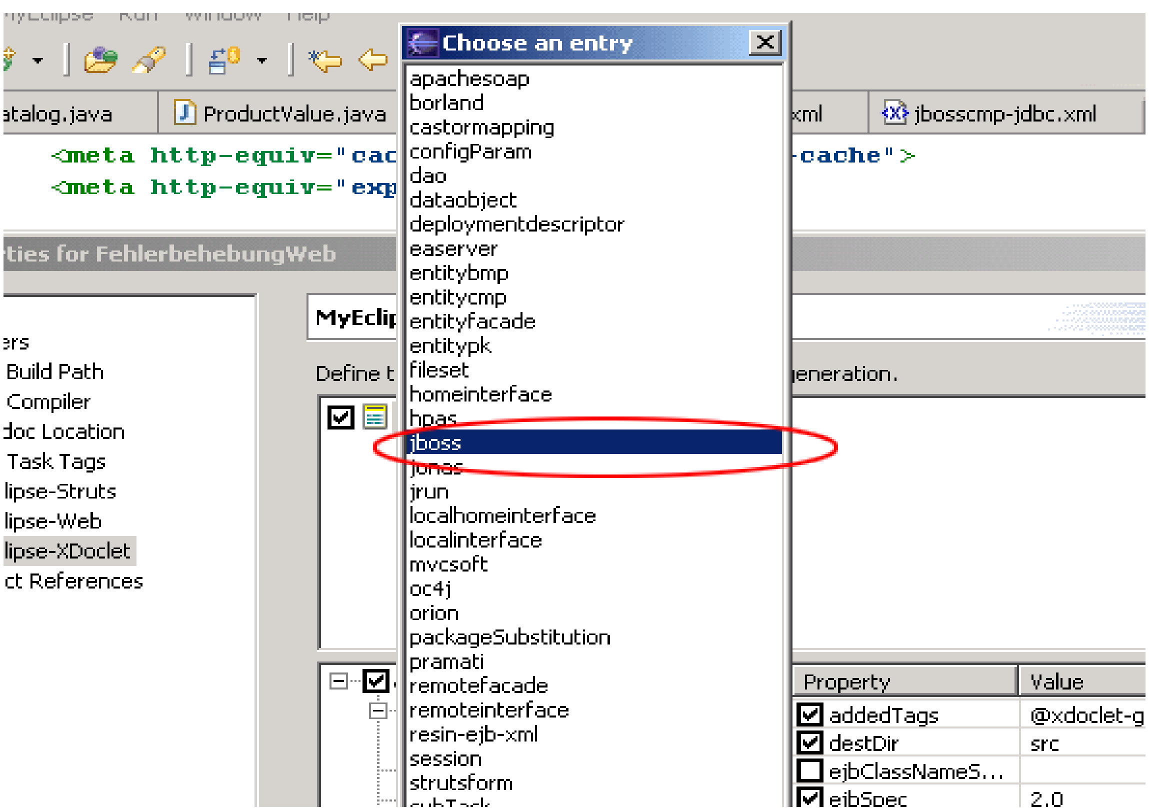
☑ Standard EJB

- ☑ ejbdoclet
  - ☑ dao
    - ☑ PackageSubstitution
  - ☑ dataobject
  - ☑ deploymentdescriptor
  - ☑ entitybmp
  - ☑ entitycmp
  - ☑ entitypk
  - ☑ fileset
  - ☑ homeinterface
  - ☑ localhomeinterface
  - ☑ localinterface

Property	Value
☑ addedTags	@xdoclet-genera...
☑ destDir	src
☐ ejbClassNameS...	
☑ ejbSpec	2.0
☑ excludedTags	@version,@auth...
☐ force	
☐ ignoredtags	
☐ mergeDir	
☐ validating	
☑ verbose	true

Restore Defaults | Apply | OK | Cancel

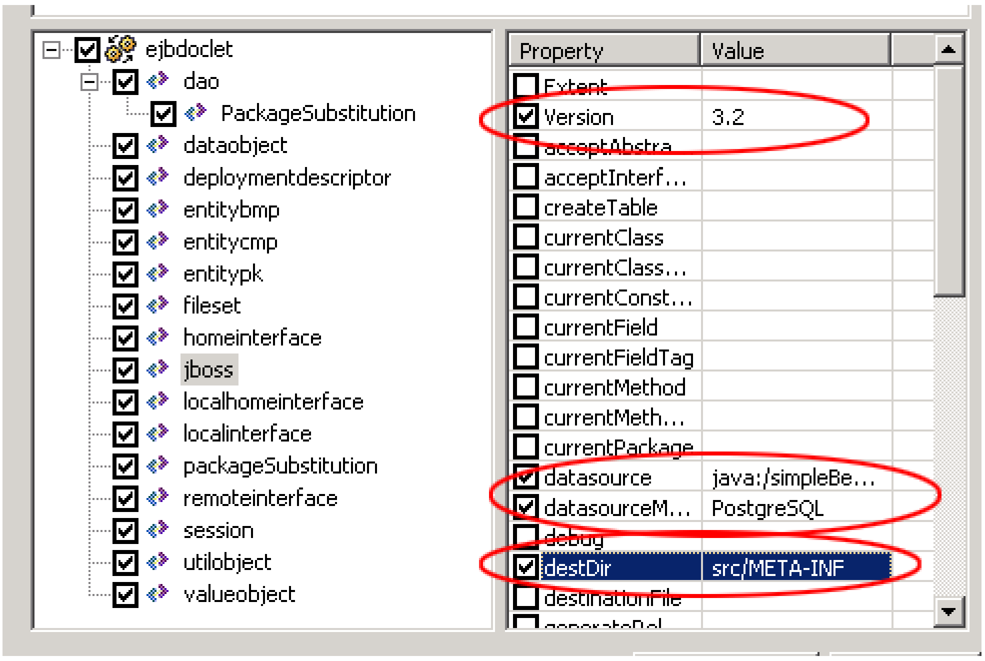
Right click on ejbdoclet and choose jboss from the list.


Choose an entry

apachesoap
borland
castormapping
configParam
dao
dataobject
deploymentdescriptor
easerver
entitybmp
entitycmp
entityfacade
entitypk
fileset
homeinterface
hpas
jboss
jonas
jrun
localhomeinterface
localinterface
mvcsoft
oc4j
orion
packageSubstitution
pramati
remotefacade
remoteinterface
resin-ejb-xml
session
strutsform

The following settings must be add.

- Jboss Version, it is 3.2 for all 3.2.x
- the destDir (where the the jboss.xml and jbosscmp-jdbc.xml will create)
- You need a datasource, we will prepare this later. Take a name here like "java:/tutorialDS"
- Datasource Mapping. Tells the Application Server what kind of field is used in the DB for a jdbc-type. Find out the name yourself by looking into {jboss_home}\server\default\conf\standardjbosscmp-jdbc.xml. mysql is mySQL for example. But we are using PostgreSQL!



- ☑ ejbdoclet
  - ☑ dao
    - ☑ PackageSubstitution
  - ☑ dataobject
  - ☑ deploymentdescriptor
  - ☑ entitybmp
  - ☑ entitycmp
  - ☑ entitypk
  - ☑ fileset
  - ☑ homeinterface
  - ☑ jboss
  - ☑ localhomeinterface
  - ☑ localinterface
  - ☑ packageSubstitution
  - ☑ remoteinterface
  - ☑ session
  - ☑ utilobject
  - ☑ valueobject

Property	Value
☐ Extent	
☑ Version	3.2
☐ acceptAbstra	
☐ acceptInterf…	
☐ createTable	
☐ currentClass	
☐ currentClass…	
☐ currentConst…	
☐ currentField	
☐ currentFieldTag	
☐ currentMethod	
☐ currentMeth…	
☐ currentPackage	
☑ datasource	java:/simpleBe…
☑ datasourceM…	PostgreSQL
☐ debug	
☑ destDir	src/META-INF
☐ destinationFile	
☐ generateRel	

Create Datasource Mapping

- Copy the driver to (you will find it on http://www.postgresql.org) to \jboss-3.2.4\server\default\lib
- Create the database in Postgre.
- Create the tables and two fields fid and fname with type text.
- You can find examples configuration files for all supported DBs in \jboss-3.2.4\docs\examples\jca\
- Copy the file postgres-ds.xml to \jboss-3.2.4\server\default\deploy

change the content of the file to:

```
<datasources>
  <local-tx-datasource>
    <jndi-name>MyDS</jndi-name>
    <connection-url>
    jdbc:postgresql://localhost:5432/database-name
    </connection-url>
    <driver-class>org.postgresql.Driver</driver-class>
    <user-name>username</user-name>
    <password>password</password>
  </local-tx-datasource>
</datasources>
```

Edit source code

Find the following position in the source code.

```
 * @ejb.bean name = "SimpleBean"
 *           type = "CMP"
 *           cmp-version = "2.x"
 *           display-name = "SimpleBean"
 *           description = "SimpleBean EJB"
 *           view-type = "both"
 *           jndi-name = "ejb/SimpleBeanHome"
 *           local-jndi-name = "ejb/SimpleBeanLocalHome"
 *
 * @ejb:util
 *      generate="physical"
 */
public abstract class SimpleBean implements EntityBean {
```

Add the following:

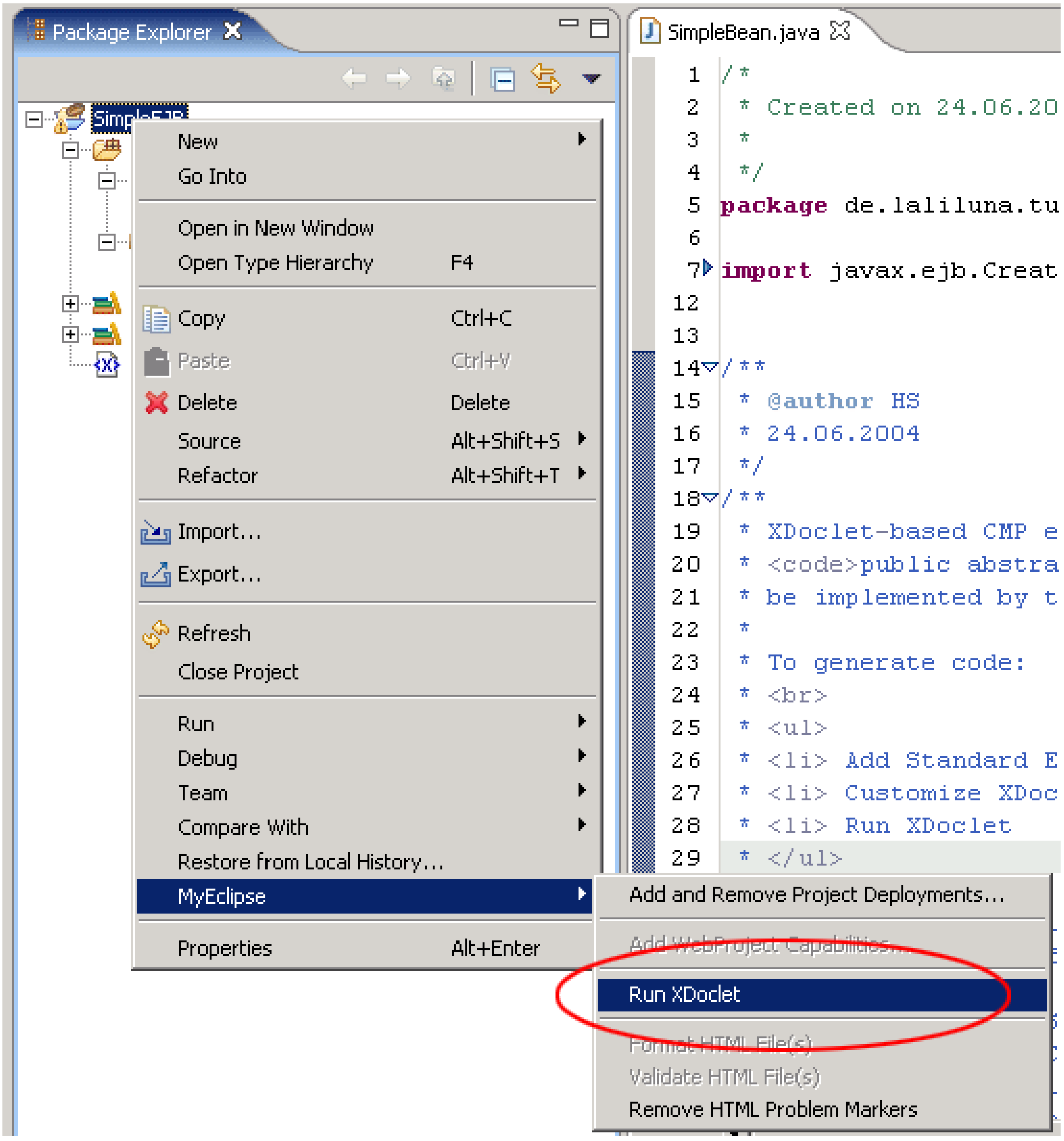
```
 * @ejb.bean name = "SimpleBean"
 *           type = "CMP"
 *           cmp-version = "2.x"
 *           display-name = "SimpleBean"
 *           description = "SimpleBean EJB"
 *           view-type = "both"
 *           jndi-name = "ejb/SimpleBeanHome"
 *           local-jndi-name = "ejb/SimpleBeanLocalHome"
 *      primkey-field = "id"
 * @ejb.persistence table-name = "tsimplebean"
 * @jboss.persistence table-name = "tsimplebean"
 * @ejb:util
 *      generate="physical"
```

Now we will add the Primary key field id and a second field name

```
public abstract class SimpleBean implements EntityBean {

        /** The EntityContext */
        private EntityContext   context;

        /**
         * @ejb.interface-method view-type = "both"
         * @ejb.persistence column-name = "fid"
         * @ejb.pk-field
         *
         * @return
         */
        public abstract String getId();

        /**
         * @ejb.interface-method view-type = "both"
         *
         * @param name
         */
        public abstract void setId(String id);

        /**
         * @ejb.interface-method view-type = "both"
         * @ejb.persistence column-name = "fname"
         *
         * @return
         */
        public abstract String getName();
        /**
         * @ejb.interface-method view-type = "both"
         *
         * @param name
         */
        public abstract void setName(String name);

}
```

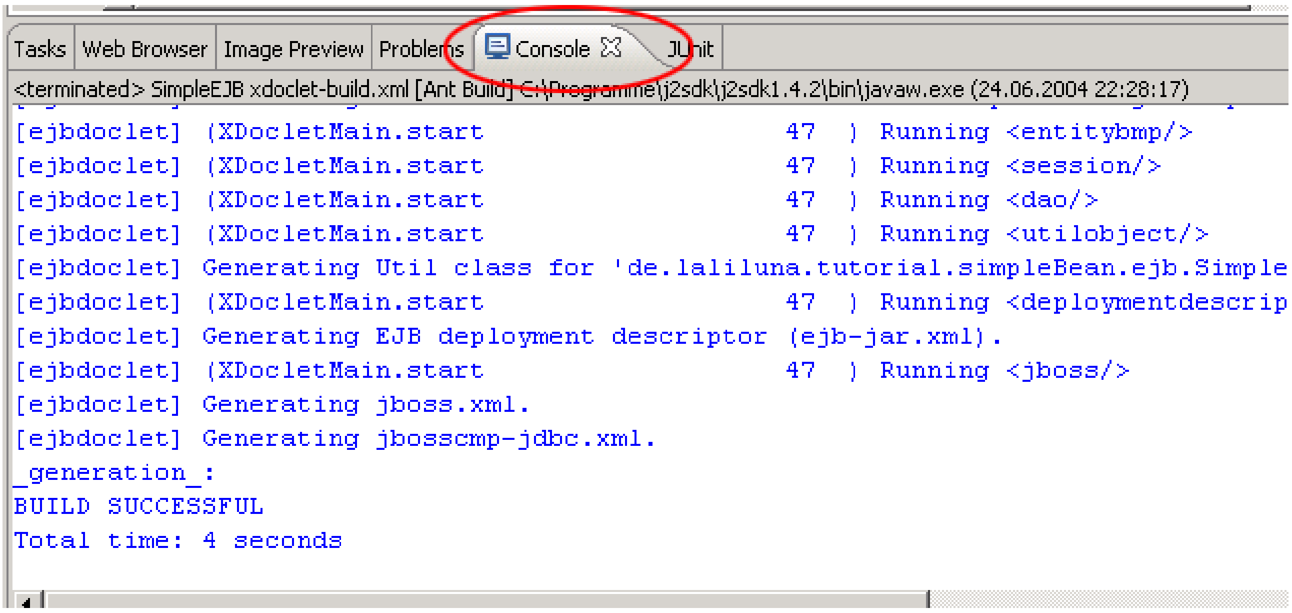
Run xDoclet

We have not completely finished the source code, but it is time for a first generation with xdoclet. Right click on the project and choose run xdoclet.



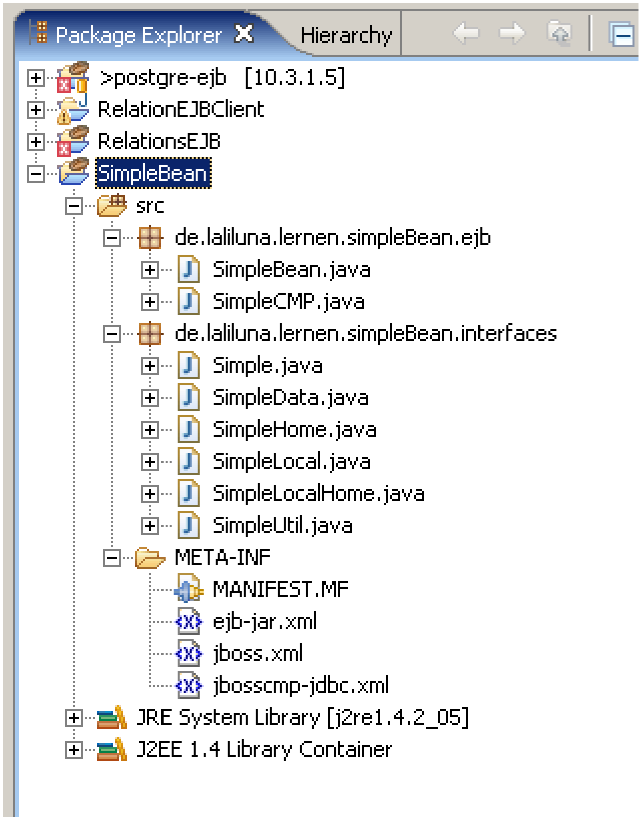
```
/*
 * Created on 24.06.20
 *
 */
package de.laliluna.tu

import javax.ejb.Creat


/**
 * @author HS
 * 24.06.2004
 */
/**
 * XDoclet-based CMP e
 * <code>public abstra
 * be implemented by t
 *
 * To generate code:
 * <br>
 * <ul>
 * <li> Add Standard E
 * <li> Customize XDoc
 * <li> Run XDoclet
 * </ul>
```

Open the console and look if everything is OK.



```
<terminated> SimpleEJB xdoclet-build.xml [Ant Build] C:\Programme\j2sdk\j2sdk1.4.2\bin\javaw.exe (24.06.2004 22:28:17)
[ejbdoclet] (XDocletMain.start                   47  ) Running <entitybmp/>
[ejbdoclet] (XDocletMain.start                   47  ) Running <session/>
[ejbdoclet] (XDocletMain.start                   47  ) Running <dao/>
[ejbdoclet] (XDocletMain.start                   47  ) Running <utilobject/>
[ejbdoclet] Generating Util class for 'de.laliluna.tutorial.simpleBean.ejb.Simple
[ejbdoclet] (XDocletMain.start                   47  ) Running <deploymentdescrip
[ejbdoclet] Generating EJB deployment descriptor (ejb-jar.xml).
[ejbdoclet] (XDocletMain.start                   47  ) Running <jboss/>
[ejbdoclet] Generating jboss.xml.
[ejbdoclet] Generating jbosscmp-jdbc.xml.
_generation_:
BUILD SUCCESSFUL
Total time: 4 seconds
```

Your package should have an interface package now, with the generated home and local interfaces. You should have a jboss.xml and a jbosscmp-jdbc.xml in src/META-INF.



```
>postgre-ejb  [10.3.1.5]
RelationEJBClient
RelationsEJB
SimpleBean
  src
    de.laliluna.lernen.simpleBean.ejb
      SimpleBean.java
      SimpleCMP.java
    de.laliluna.lernen.simpleBean.interfaces
      Simple.java
      SimpleData.java
      SimpleHome.java
      SimpleLocal.java
      SimpleLocalHome.java
      SimpleUtil.java
    META-INF
      MANIFEST.MF
      ejb-jar.xml
      jboss.xml
      jbosscmp-jdbc.xml
  JRE System Library [j2re1.4.2_05]
  J2EE 1.4 Library Container
```

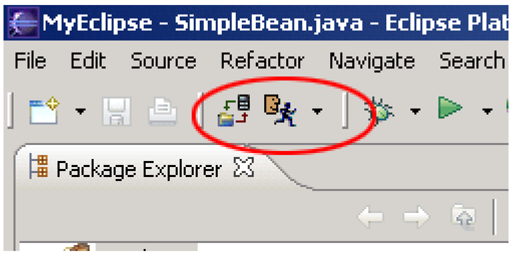
Have a look in the file SimpleBeanUtil. You will find some useful functions.
We will use one to finish our bean. Change the ejbCreate method to the following:

```
   * @ejb.create-method
   */
  public String ejbCreate() throws CreateException
  {
    this.setId(SimpleUtil.generateGUID(this));
    return null;
  }
```

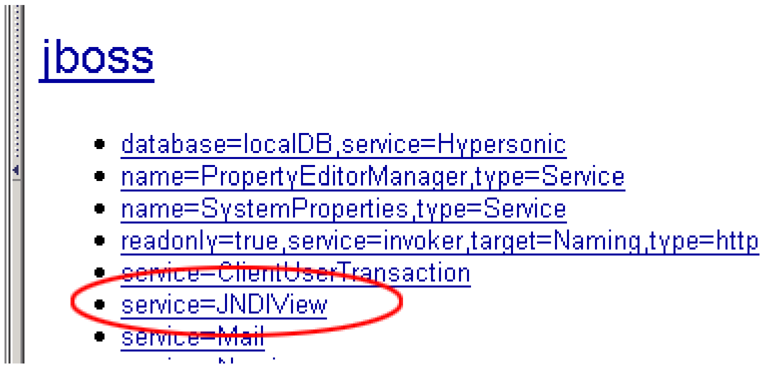
This will generate a random ID. That's it.

Run the jboss server

Start the jboss server and deploy the project.



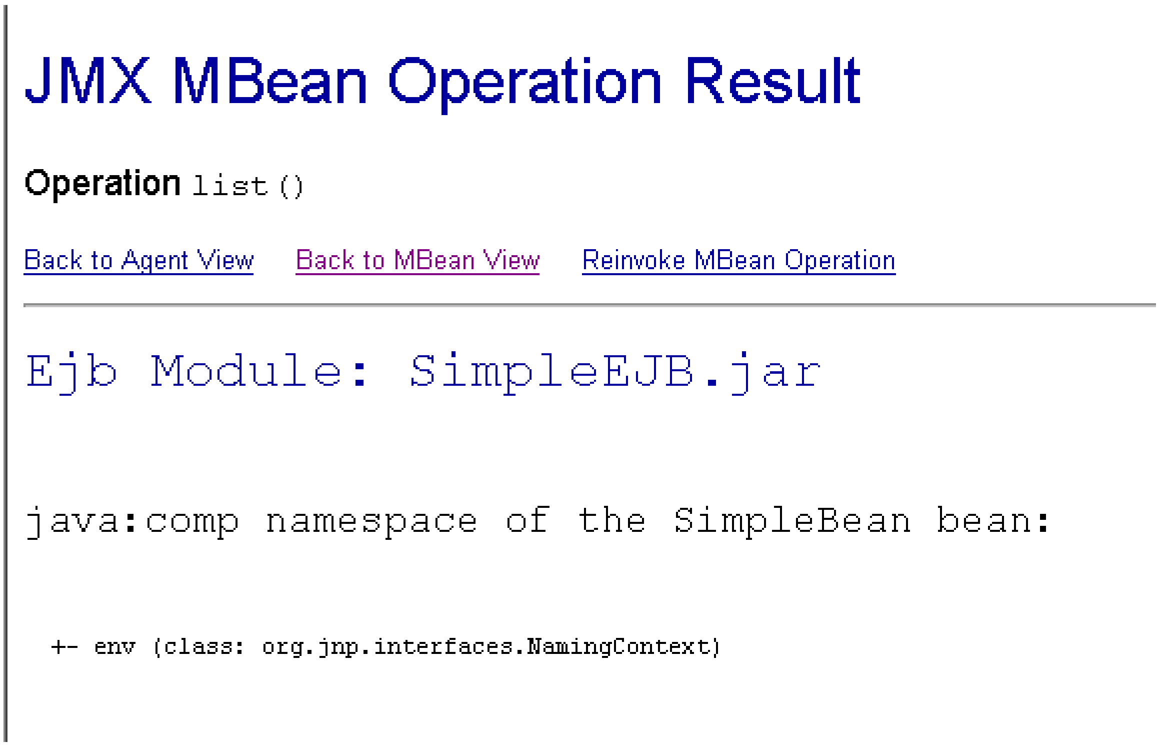
Now open http://localhost:8080/jmx-console in your browser and select service JNDI-View, than select operation „list“



jboss

- database=localDB,service=Hypersonic
- name=PropertyEditorManager,type=Service
- name=SystemProperties,type=Service
- readonly=true,service=invoker,target=Naming,type=http
- service=ClientUserTransaction
- service=JNDIView
- service=Mail

You can see your bean module now and it should also occur in the global JNDI namespace.

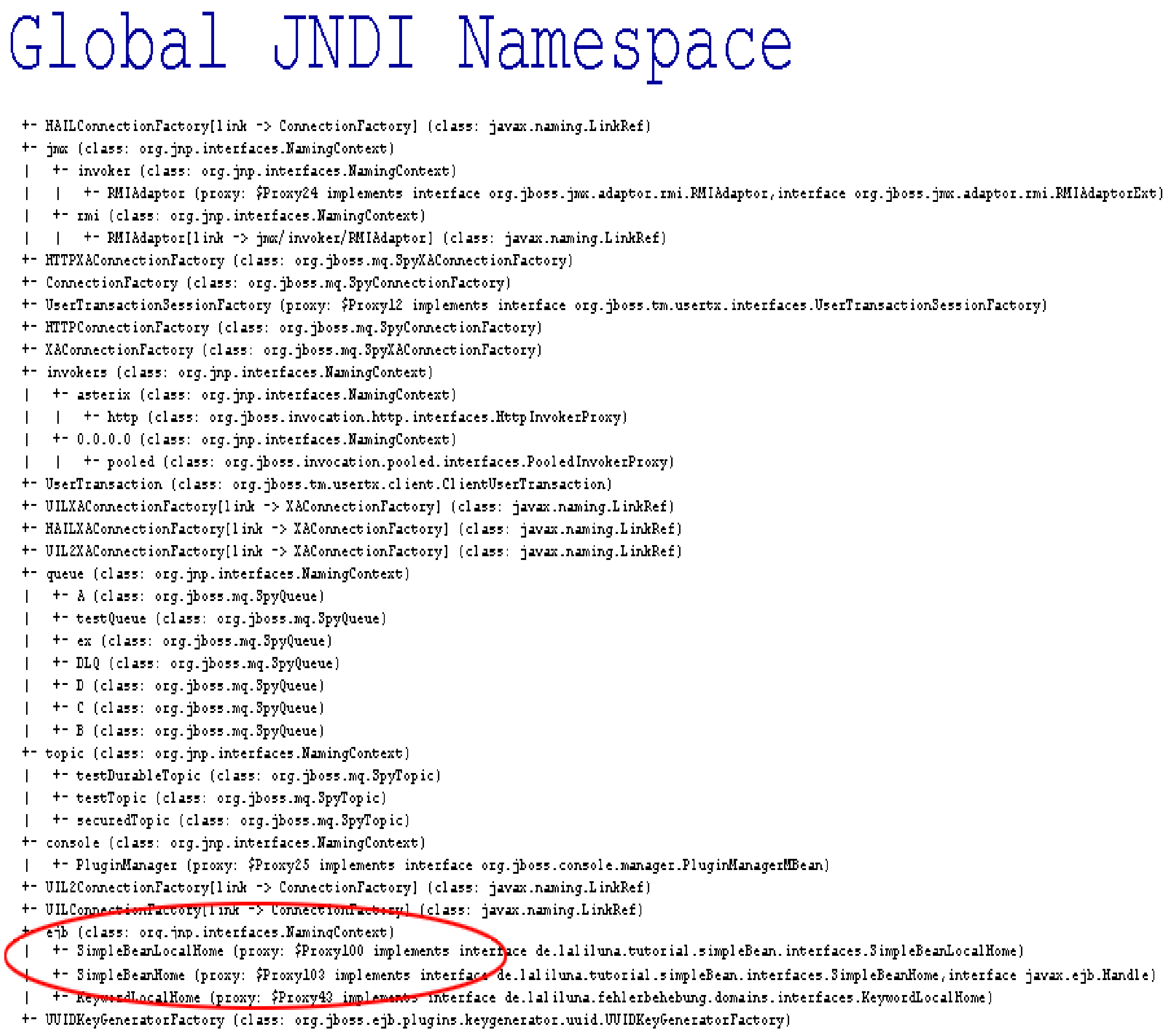


JMX MBean Operation Result

Operation list()

Back to Agent View   Back to MBean View   Reinvoke MBean Operation

```
Ejb Module: SimpleEJB.jar

java:comp namespace of the SimpleBean bean:

  +- env (class: org.jnp.interfaces.NamingContext)
```

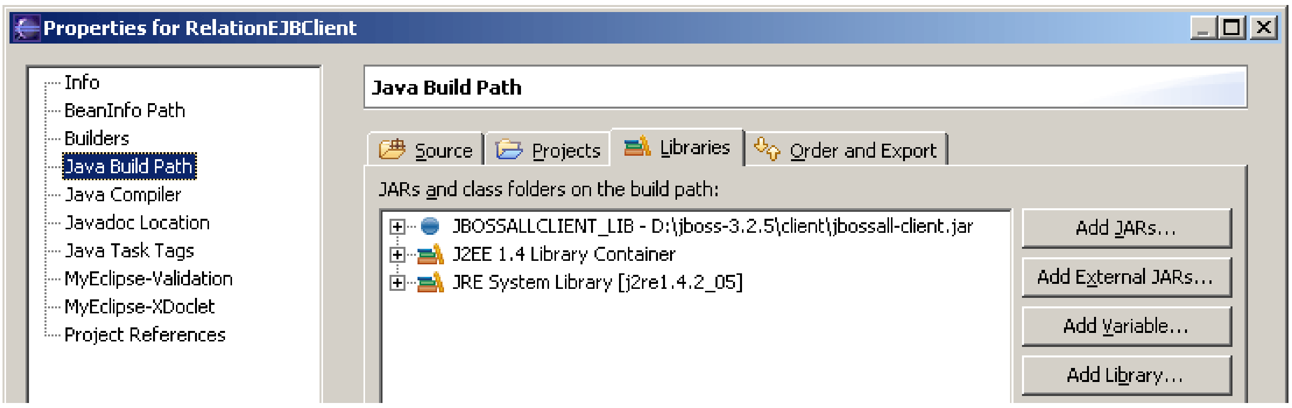


Global JNDI Namespace

```
+- HAILConnectionFactory[link -> ConnectionFactory] (class: javax.naming.LinkRef)
+- jmx (class: org.jnp.interfaces.NamingContext)
|   +- invoker (class: org.jnp.interfaces.NamingContext)
|   |   +- RMIAdaptor (proxy: $Proxy24 implements interface org.jboss.jmx.adaptor.rmi.RMIAdaptor,interface org.jboss.jmx.adaptor.rmi.RMIAdaptorExt)
|   +- rmi (class: org.jnp.interfaces.NamingContext)
|   |   +- RMIAdaptor[link -> jmx/invoker/RMIAdaptor] (class: javax.naming.LinkRef)
+- HTTPXAConnectionFactory (class: org.jboss.mq.SpyXAConnectionFactory)
+- ConnectionFactory (class: org.jboss.mq.SpyConnectionFactory)
+- UserTransactionSessionFactory (proxy: $Proxy12 implements interface org.jboss.tm.usertx.interfaces.UserTransactionSessionFactory)
+- HTTPConnectionFactory (class: org.jboss.mq.SpyConnectionFactory)
+- XAConnectionFactory (class: org.jboss.mq.SpyXAConnectionFactory)
+- invokers (class: org.jnp.interfaces.NamingContext)
|   +- asterix (class: org.jnp.interfaces.NamingContext)
|   |   +- http (class: org.jboss.invocation.http.interfaces.HttpInvokerProxy)
|   +- 0.0.0.0 (class: org.jnp.interfaces.NamingContext)
|   |   +- pooled (class: org.jboss.invocation.pooled.interfaces.PooledInvokerProxy)
+- UserTransaction (class: org.jboss.tm.usertx.client.ClientUserTransaction)
+- UILXAConnectionFactory[link -> XAConnectionFactory] (class: javax.naming.LinkRef)
+- HAILXAConnectionFactory[link -> XAConnectionFactory] (class: javax.naming.LinkRef)
+- UIL2XAConnectionFactory[link -> XAConnectionFactory] (class: javax.naming.LinkRef)
+- queue (class: org.jnp.interfaces.NamingContext)
|   +- A (class: org.jboss.mq.SpyQueue)
|   +- testQueue (class: org.jboss.mq.SpyQueue)
|   +- ex (class: org.jboss.mq.SpyQueue)
|   +- DLQ (class: org.jboss.mq.SpyQueue)
|   +- D (class: org.jboss.mq.SpyQueue)
|   +- C (class: org.jboss.mq.SpyQueue)
|   +- B (class: org.jboss.mq.SpyQueue)
+- topic (class: org.jnp.interfaces.NamingContext)
|   +- testDurableTopic (class: org.jboss.mq.SpyTopic)
|   +- testTopic (class: org.jboss.mq.SpyTopic)
|   +- securedTopic (class: org.jboss.mq.SpyTopic)
+- console (class: org.jnp.interfaces.NamingContext)
|   +- PluginManager (proxy: $Proxy25 implements interface org.jboss.console.manager.PluginManagerMBean)
+- UIL2ConnectionFactory[link -> ConnectionFactory] (class: javax.naming.LinkRef)
+- UILConnectionFactory[link -> ConnectionFactory] (class: javax.naming.LinkRef)
+- ejb (class: org.jnp.interfaces.NamingContext)
|   +- SimpleBeanLocalHome (proxy: $Proxy100 implements interface de.laliluna.tutorial.simpleBean.interfaces.SimpleBeanLocalHome)
|   +- SimpleBeanHome (proxy: $Proxy103 implements interface de.laliluna.tutorial.simpleBean.interfaces.SimpleBeanHome,interface javax.ejb.Handle)
|   +- KeywordLocalHome (proxy: $Proxy43 implements interface de.laliluna.fehlerbehebung.domains.interfaces.KeywordLocalHome)
+- UUIDKeyGeneratorFactory (class: org.jboss.ejb.plugins.keygenerator.uuid.UUIDKeyGeneratorFactory)
```

Test the Bean

Create a Java project. Open the project properties and add the library j2ee and the jar jbossall-client.



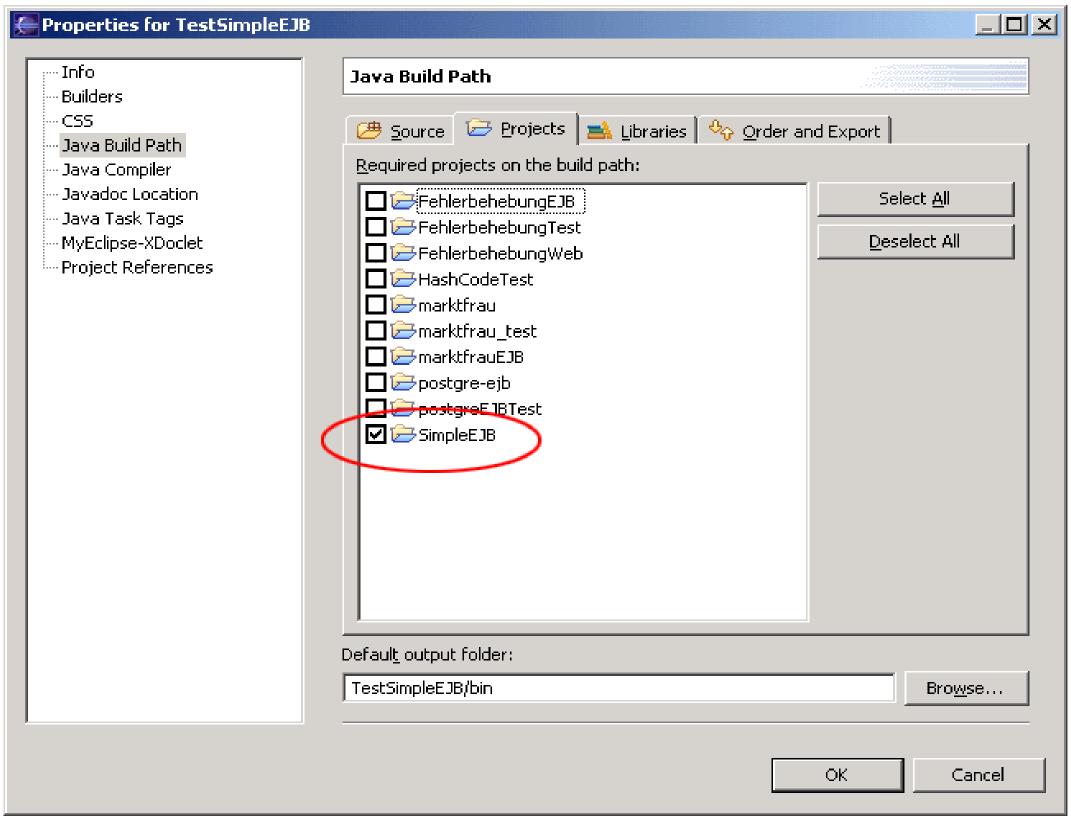
Properties for RelationEJBClient

Java Build Path

JARs and class folders on the build path:

- JBOSSALLCLIENT_LIB - D:\jboss-3.2.5\client\jbossall-client.jar
- J2EE 1.4 Library Container
- JRE System Library [j2re1.4.2_05]

Include your EJB project.



Properties for TestSimpleEJB

Java Build Path

Required projects on the build path:

- [ ] FehlerbehebungEJB
- [ ] FehlerbehebungTest
- [ ] FehlerbehebungWeb
- [ ] HashCodeTest
- [ ] marktfrau
- [ ] marktfrau_test
- [ ] marktfrauEJB
- [ ] postgre-ejb
- [ ] postgreEJBTest
- [✓] SimpleEJB

Default output folder: TestSimpleEJB/bin

Create a new Class like the following and run it as java application.

```
package de.laliluna.tutorial.simpleBean;

import java.rmi.RemoteException;
import java.util.Properties;

import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.rmi.PortableRemoteObject;

import de.laliluna.tutorial.simpleBean.interfaces.Simple;
import de.laliluna.tutorial.simpleBean.interfaces.SimpleHome;

/**
 * @author HS
 *
 *
 */
public class SimpleBeanClient {

  Properties properties;

  public SimpleBeanClient() {
    properties = new Properties();
    properties.put("java.naming.factory.initial",
        "org.jnp.interfaces.NamingContextFactory");
    properties.put("java.naming.factory.url.pkgs",
        "org.jboss.naming:org.jnp.interfaces");
    properties.put("java.naming.provider.url", "jnp://localhost:1099");
    properties.put("jnp.disableDiscovery", "true");
  }

  public static void main(String[] args) {
    SimpleBeanClient beanClient = new SimpleBeanClient();
    beanClient.createBean();
  }

  public void createBean() throws EJBException {
    try {
      // [laliluna] create a context to look up the beans in the JNDI
      InitialContext context = new InitialContext(properties);
      /*
       * [laliluna]
       * we have to look up the remote interaces as we are not in the same
environment as the EJB.
       * Therefore we will have to use the PortableRemote class to convert our
object
       */
      Object object = context.lookup(SimpleHome.JNDI_NAME);
      SimpleHome simpleHome = (SimpleHome) PortableRemoteObject.narrow(object,
          SimpleHome.class);

      Simple simple = simpleHome.create();
      simple.setName("Gunter");
      System.out.println(simple.getId());
      System.out.println(simple.getName());

    } catch (NamingException e) {
      throw new EJBException(e);
    } catch (RemoteException e) {
      throw new EJBException(e);
    } catch (CreateException e) {
      throw new EJBException(e);
    }

  }
}
```

Congratulations that it!