Edition

1

Date: 2003-08-09, 0:08:07 y8/P8

LAURENT ETIEMBLE

The JBoss Project

JBoss IDE 1.2.0 Tutorial

LAURENT ETIEMBLE, AND THE JBOSS PROJECT

JBoss IDE 1.2.0 : Tutorial

Table of Content

Preface

i

Preface

Forward

The JBoss-IDE started with an XDoclet plug-in for eclipse in the middle of 2002. Then Hans Dockter met Marc as he participated at a JBoss training in Mallorca and they talked about the possibility of developing a JBoss-IDE.

About the Authors

Hans Dockter, is the lead architect of the JBoss-IDE. Hans works as an independent consultant and lives in Berlin, Germany.

Laurent Etiemble, is working as a developper on JBoss-IDE. Laurent works as a consultant and lives in Paris, France.

JBoss Project, headed by Marc Fleury, is composed of over 100 developers worldwide who are working to deliver a full range of J2EE tools, making JBoss the premier Enterprise Java application server for the Java 2 Enterprise Edition platform.

JBoss is an Open Source, standards-compliant, J2EE application server implemented in 100% Pure Java. The JBoss/Server and complement of products are delivered under a public license. With a huge amount of downloads per month, JBoss is the most downloaded J2EE based server in the industry.

Acknowledgments

I'd like to thank SG Enterprise Systems and Chris Bedford, author of a nice tutorial for JBoss IDE. This one is inspired from it.

Chapter 1

1. Introduction

JBoss-IDE offers you:

- A very comfortable and sophisticated support for XDoclet.
- The debugging and monitoring of JBoss servers and the controlling of there life cycles.
- An easy way to configure the packaging layout of archives (packed or exploded)
- A simple way to deploy the packaged and/or exploded archive to a JBoss server

Some part of the J2EE development process is not yet covered. That is J2EE specific project management (templates and wiazrds). This has still to be done. Anyway, the main focus for the next version is on these topics.

The following tutorial is an overview of the capabilities of JBoss-IDE. It aims to demonstrate how to use it to develop and debug J2EE applications.

Chapter

2

2. The Tutorial

Introduction

The goals of this tutorial are to demonstrate how it is simple to developp J2EE application with Jboss-IDE/Eclipse. The sample application that will be built is a J2EE application with one session EJB and one Servlet, which computes the Fibonacci suite.

The tutorial is split into several parts:

- The project: this part shows how the project is prepared (source and build path)
- The EJB: this part shows how to write an EJB class with its XDoclet javadoc tags.
- Generation of EJB files: this part shows how to configure the XDoclet generation configuration to generate all the EJB related files
- The Servlet and the Web-App: this part shows how to write a Servlet class with its XDoclet javadoc tags.
- Generation of Servlet files: this part shows how to configure the XDoclet generation configuration to generate all the Web related files
- The J2EE application : this part shows how to create the missing files.
- Packaging: this part shows how to package the J2EE application
- JBoss configuration : this part shows how to define debug configuration to launch JBoss inside Eclipse.
- Deployment : this part shows how to deploy by copy the J2EE application
- Debugging: this part shows how to set up breakpoints to debug the deployed application.

Requirements

For this tutorial you need :

- Java Development Kit 1.3.0 or higher (a JDK is needed to launch JBoss 3.x)
- Eclipse 2.1.0 (from http://www.eclipse.org)

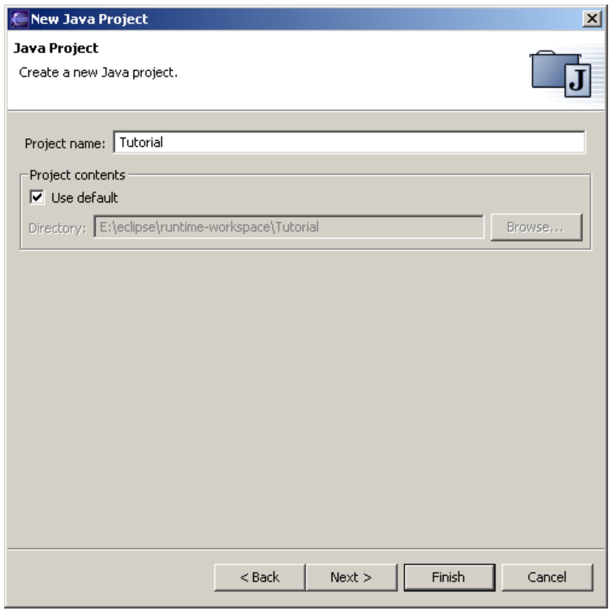
- JBoss Server 3.0.x or 3.2.x

You also need to have some Eclipse basis about development and debugging. Refer to Eclipse website for further informations.

The Project

This part shows how the project is prepared. We will create source folder, import libraries and make the build path.

Create a new Java Project. Enter "Tutorial" as the name of the new project.



Create a new Java project.

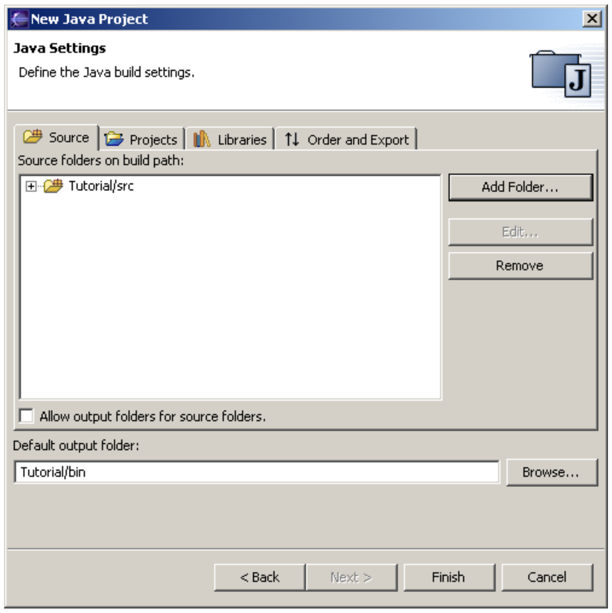
Project name: Tutorial

Project contents

☑ Use default

Directory: E:\eclipse\runtime-workspace\Tutorial

Create a source folder named “src”. The default output folder will be “bin”.


New Java Project

Java Settings

Define the Java build settings.

Source folders on build path:

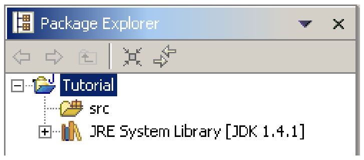
Tutorial/src

Allow output folders for source folders.

Default output folder:

Tutorial/bin

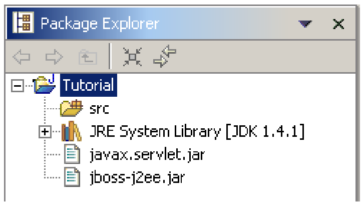
In the package explorer, the new project should look like this.


Package Explorer

Tutorial
src
JRE System Library [JDK 1.4.1]

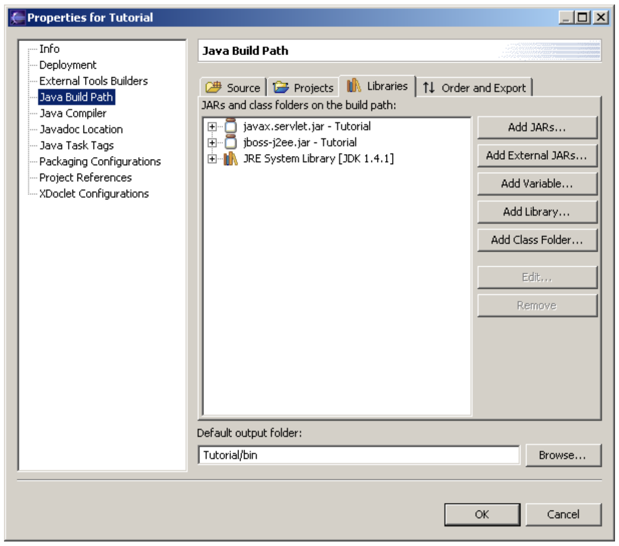
Copy the two following files from your JBoss distribution to the root of the project:

- javax.servlet.jar (taken from the server/default/lib of the JBoss distribution)
- jboss-j2ee.jar (taken from the client folder of the JBoss distribution)


Package Explorer

Tutorial
src
JRE System Library [JDK 1.4.1]
javax.servlet.jar
jboss-j2ee.jar

We want to put the two jars in the project build path. Edit the project properties by right clicking on the project and select "Properties". In the property page, select "Java Build Path". Click on "Add jars..." and select the two libraries (javax.servlet.jar and jboss-j2ee.jar).

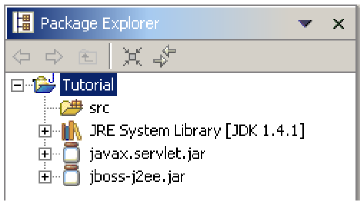


JARs and class folders on the build path:

javax.servlet.jar - Tutorial
jboss-j2ee.jar - Tutorial
JRE System Library [JDK 1.4.1]

Default output folder: Tutorial/bin

Save the changes and you should see something as above. Now, you have a project with all you need to develop a nice J2EE application!

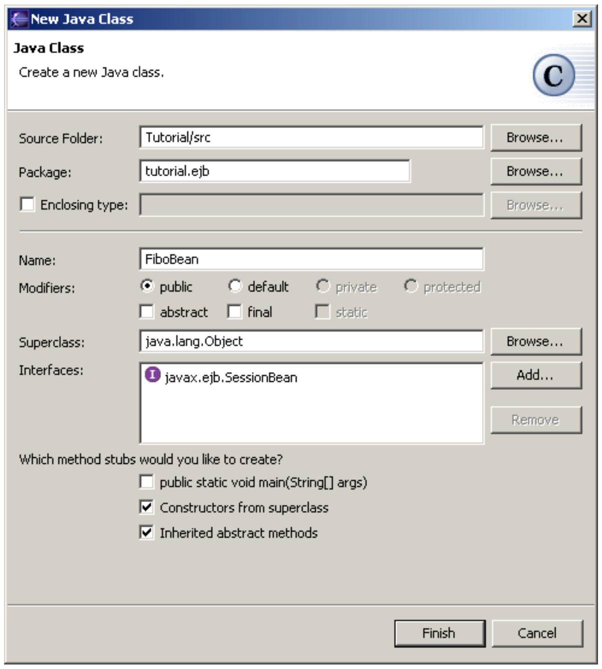


Tutorial
src
JRE System Library [JDK 1.4.1]
javax.servlet.jar
jboss-j2ee.jar

The EJB

The next step is to create an EJB. For simplicity, it will be a stateless session bean, but others types are also easy to write.

Create a new Java Class. The package will be "tutorial.ejb" and the class name "FiboBean". Click on "Add…" to add the "SessionBean" interface. Be sure that only "Constructors from superclass" and "Inherited abstract methods" are checked.


New Java Class

Java Class

Create a new Java class.

Source Folder: Tutorial/src

Package: tutorial.ejb

Enclosing type:

Name: FiboBean

Modifiers: [✓ public] default private protected

abstract final static

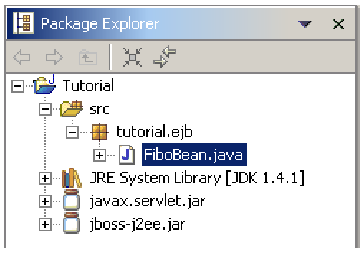
Superclass: java.lang.Object

Interfaces: javax.ejb.SessionBean

Which method stubs would you like to create?

- [ ] public static void main(String[] args)
- [x] Constructors from superclass
- [x] Inherited abstract methods

Click on "Finish". The class is then created and you should have a project like this.



Open the **FiboBean.java** file. We need a compliant EJB, so add an "ejbCreate" method without parameters, as our EJB is stateless.

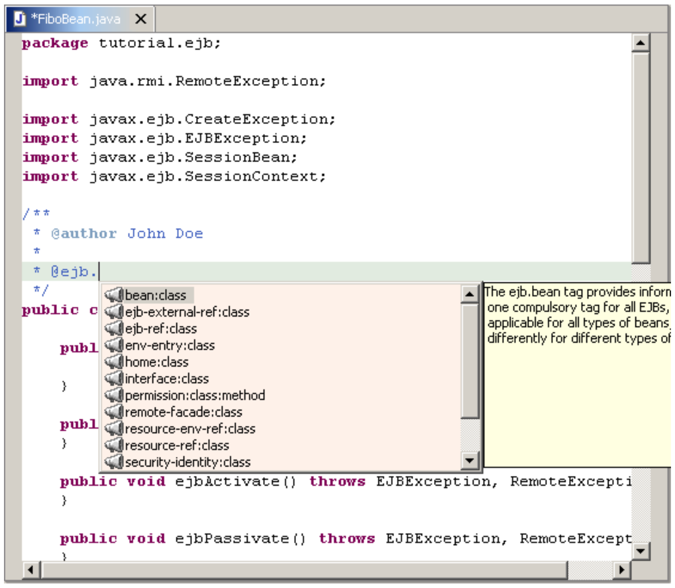
```
    public void ejbCreate() throws CreateException {
    }
```

To be interesting, our EJB need a business method. The method will compute a Fibonacci suite.

```
    public double[] compute(int number) {
        if (number < 0) {
            throw new EJBException("Argument should be positive");
        }

        double[] suite = new double[number + 1];
        suite[0] = 0;

        if (number == 0) {
```

```
                        return suite;
                }

                suite[1] = 1;

                for (int i = 2; i <= number; i++) {
                        suite[i] = suite[i - 1] + suite[i - 2];
                }

                return suite;
        }
```

The next step is to insert XDoclet javadoc related tags for the EJB. Thanks to the code completion, it is an easy operation. In the Java editor go in the Javadoc class paragraph. Type “@ejb.” And press CTRL+Space. You should see the magic of completion in action.



```
package tutorial.ejb;

import java.rmi.RemoteException;

import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

/**
 * @author John Doe
 *
 * @ejb.
 */
public c

    publ

    }

    publ
    }

    public void ejbActivate() throws EJBException, RemoteExcepti
    }

    public void ejbPassivate() throws EJBException, RemoteExcept
    }
```

```
bean:class
ejb-external-ref:class
ejb-ref:class
env-entry:class
home:class
interface:class
permission:class:method
remote-facade:class
resource-env-ref:class
resource-ref:class
security-identity:class
```

The ejb.bean tag provides inforn
one compulsory tag for all EJBs,
applicable for all types of beans
differently for different types of

Complete the attributes of the tag with the following values (press CTRL+Space for each attribute if you want the completion) :

```
 *
 * @ejb.bean  name = "Fibo"
 *            display-name = "Fibo EJB"
 *            description = "EJB that computes Fibonacci suite"
 *            view-type = "remote"
 *            jndi-name = "ejb/tutorial/Fibo"
 */
public class FiboBean implements SessionBean {
```

The "ejbCreate" and the "compute" method need to be tagged. Add the following to the "ejbCreate" method :

```
        /**
         * Default create method
         * @throws CreateException
```

```
	 * @ejb.create-method
	 */
	public void ejbCreate() throws CreateException {
```

And the following to the "compute" method :

```
	/**
	 * @param number
	 * @return
	 *
	 * @ejb.interface-method view-type = "remote"
	 */
	public double[] compute(int number) {
```

After that, the file should look like this. Now, we are ready to run XDoclet on the file generate all the EJB stuff.

```
package tutorial.ejb;

import java.rmi.RemoteException;

import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

/**
 * @author John Doe
 *
 * @ejb.bean  name = "Fibo"
 *            display-name = "Fibo EJB"
 *            description = "EJB that computes Fibonacci suite"
 *            view-type = "remote"
 *            jndi-name = "ejb/tutorial/Fibo"
 */
public class FiboBean implements SessionBean {

	public FiboBean() {
		super();
	}

	/**
	 * Default create method
	 * @throws CreateException
	 * @ejb.create-method
	 */
	public void ejbCreate() throws CreateException {
	}

	public void ejbActivate() throws EJBException, RemoteException {
	}
	public void ejbPassivate() throws EJBException, RemoteException {
	}

	public void ejbRemove() throws EJBException, RemoteException {
	}

	public void setSessionContext(SessionContext arg0)
		throws EJBException, RemoteException {
	}

	/**
	 * @param number
	 * @return
	 *
	 * @ejb.interface-method view-type = "remote"
	 */
	public double[] compute(int number) {
		if (number < 0) {
			throw new EJBException("Argument should be positive");
		}

		double[] suite = new double[number + 1];
		suite[0] = 0;

		if (number == 0) {
			return suite;
		}

		suite[1] = 1;

		for (int i = 2; i <= number; i++) {
			suite[i] = suite[i - 1] + suite[i - 2];
		}

		return suite;
	}
}
```

Generation of the EJB related files

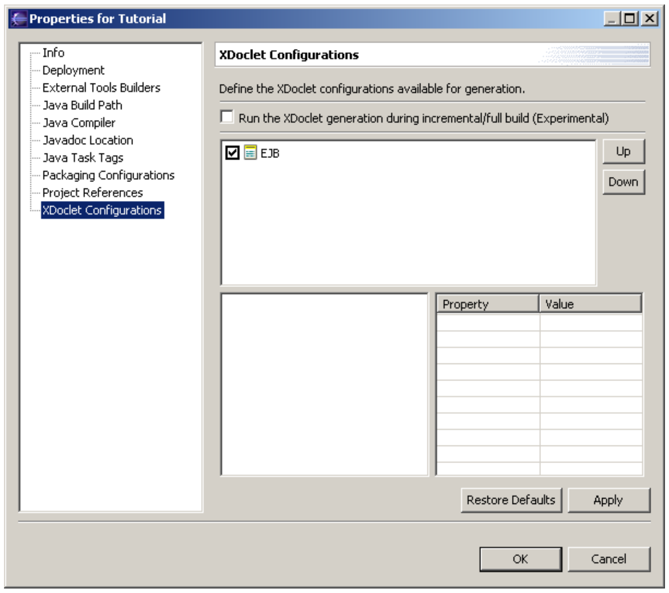
To generate the EJB related classes and descriptors, we need to make some XDoclet configuration. With JBoss IDE, you can define several XDoclet generation configurations that will be run against the project.

Edit the project properties by right clicking on the project and select "Properties".

In the property page, select "XDoclet configurations".

Right-click in the upper area to pop-up the menu and choose "Add". Type "EJB" in the dialog and click "Ok".

You have created a new generation configuration named "EJB".



XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

☑ EJB

Select the "EJB" configuration.

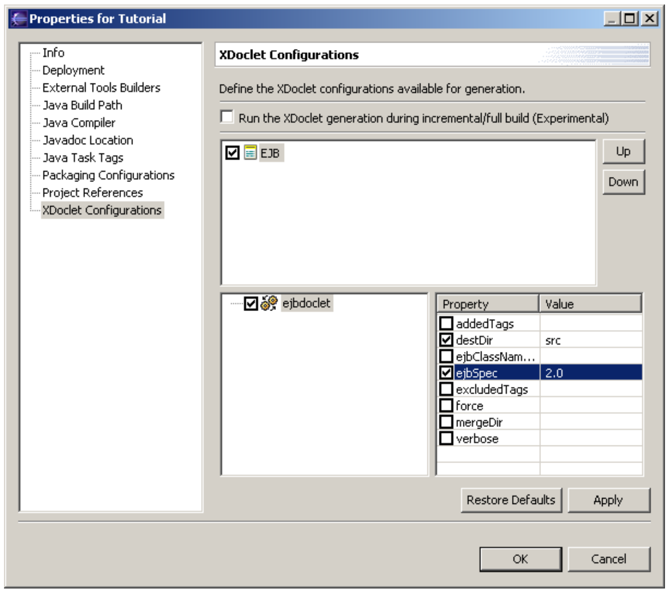
In the lower-left area, right-click to popup the menu and choose "Add Doclet".

A list of available doclets will appear. Choose "ejbdoclet" and click "Ok".

On the lower-right area, you see the properties of the "ejbdoclet". Set them to :

- "destDir" with "src" and ckeck it
- "ejbSpec" with "2.0" and ckeck it

Our configuration now contains an "ejbdoclet" that will produce files in "src" folder and for the EJB 2.0 specifications.



XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

☑ EJB

☑ ejbdoclet

Property	Value
☐ addedTags	
☑ destDir	src
☐ ejbClassNam...	
☑ ejbSpec	2.0
☐ excludedTags	
☐ force	
☐ mergeDir	
☐ verbose	

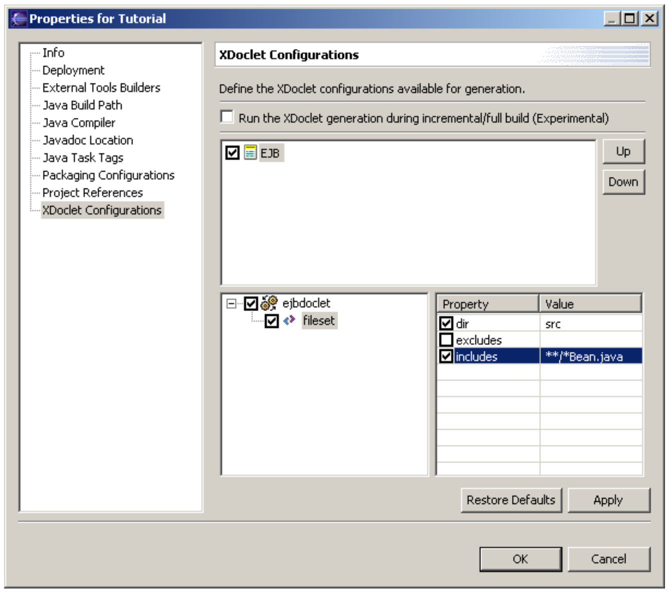
In the lower-left area, right-click on "ejbdoclet" to popup the menu and choose "Add ".

A list of available subtasks will appear. Choose "fileset" and click "Ok".

On the lower-right area, you see the properties of the " fileset ". Set them to :

- "dir" with "src" and ckeck it
- uncheck "excludes"
- "includes" with "**/*Bean.java" and ckeck it

Our configuration now contains an "ejbdoclet" with a "fileset" based on "src" and that filters source files to only pick up the EJB ones.



XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

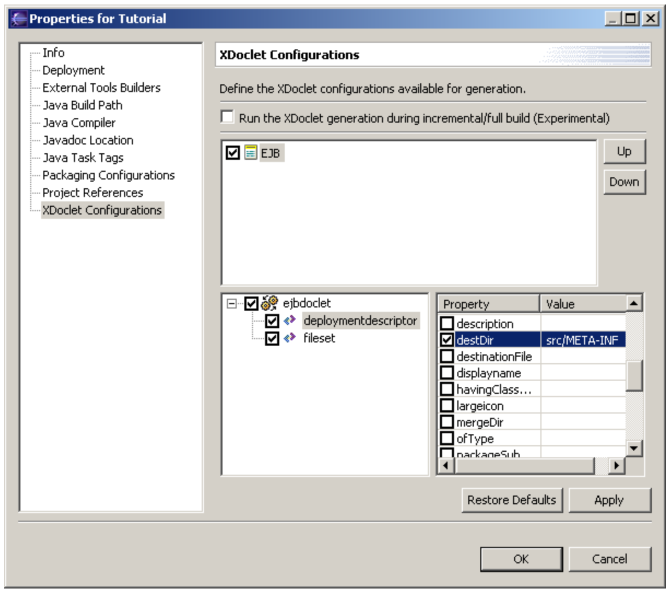
☑ EJB

☑ ejbdoclet
 ☑ fileset

Property	Value
☑ dir	src
☐ excludes	
☑ includes	**/*Bean.java

Repeat the operation to add a new subtask "deploymentdescriptor". The property to set is "destDir" with "src/META-INF". Don't forget to check it.

We will generate the deployment descriptor in the "src/META-INF" folder to remain simple.



XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

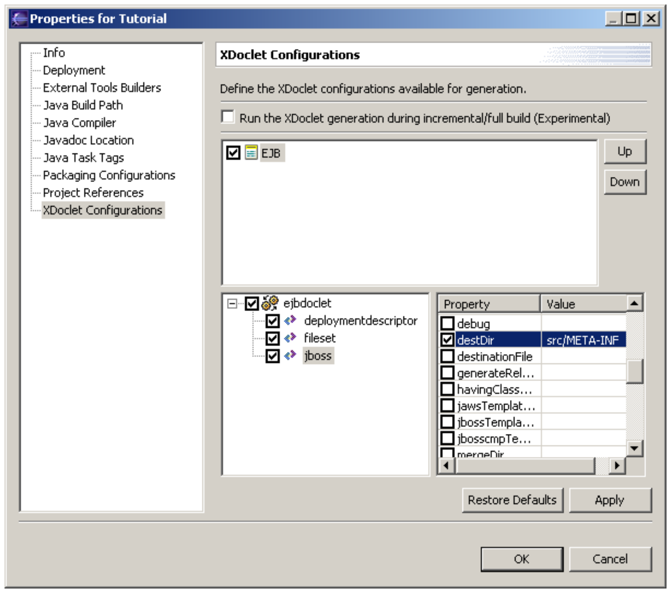
☑ EJB

☑ ejbdoclet
 ☑ deploymentdescriptor
 ☑ fileset

Property	Value
☐ description	
☑ destDir	src/META-INF
☐ destinationFile	
☐ displayname	
☐ havingClass...	
☐ largeicon	
☐ mergeDir	
☐ ofType	
☐ packageSub	

Repeat the operation to add a new subtask "jboss". The properties to set are "destDir" with "src/META-INF" and "Version" with "3.0". Don't forget to check them.

We will generate the jboss deployment descriptor in the "src/META-INF" folder and for the 3.0 version.



XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

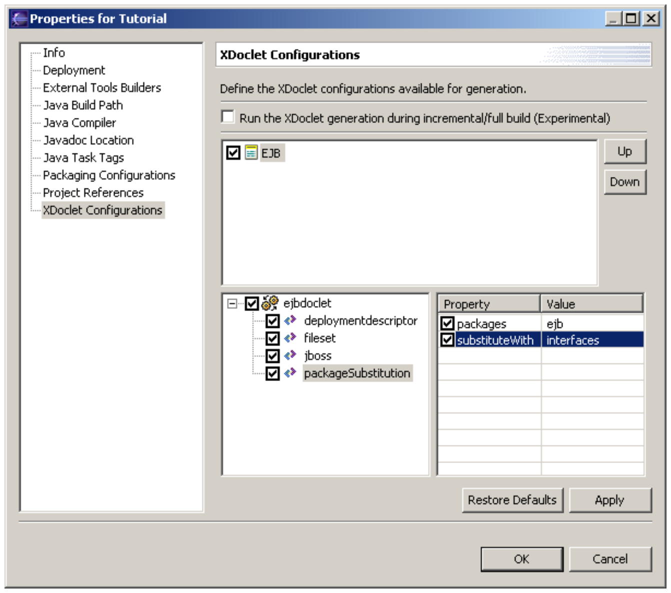
☑ EJB

- ☑ ejbdoclet
  - ☑ deploymentdescriptor
  - ☑ fileset
  - ☑ jboss

Property	Value
☐ debug	
☑ destDir	src/META-INF
☐ destinationFile	
☐ generateRel...	
☐ havingClass...	
☐ jawsTemplat...	
☐ jbossTempla...	
☐ jbosscmpTe...	
☐ mergeDir	

Repeat the operation to add a new subtask "packageSubstitution". The properties to set are "packages" with "ejb" and "substituteWith" with "interfaces". Don't forget to check them.

We will generate the EJB related classes in a different package (EJB lies in tutorial.ejb, interfaces will be generated in tutorial.interfaces).



XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

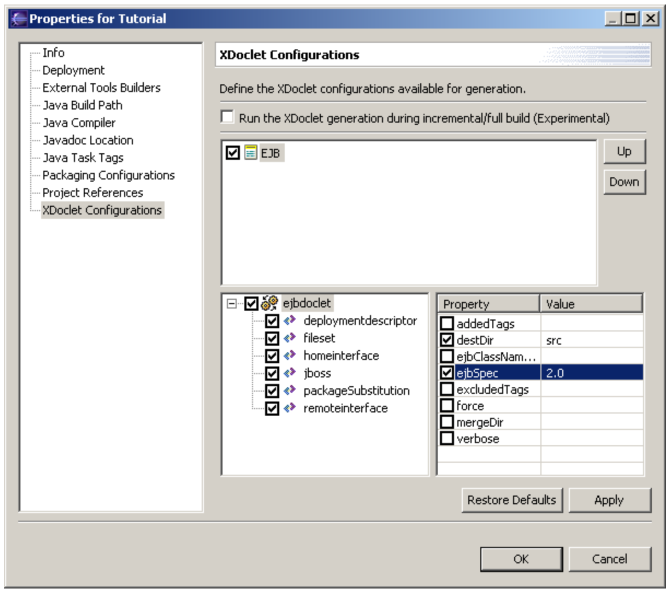
☑ EJB

- ☑ ejbdoclet
  - ☑ deploymentdescriptor
  - ☑ fileset
  - ☑ jboss
  - ☑ packageSubstitution

Property	Value
☑ packages	ejb
☑ substituteWith	interfaces

Repeat the operation to add new subtasks "remoteinterface" and "homeinterface". No properties are to be set.

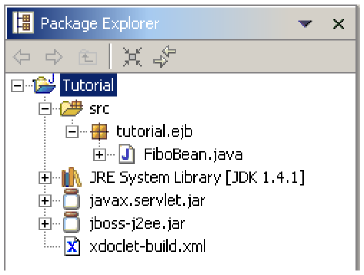
These subtasks are to generate the EJB Home and Remote interfaces.



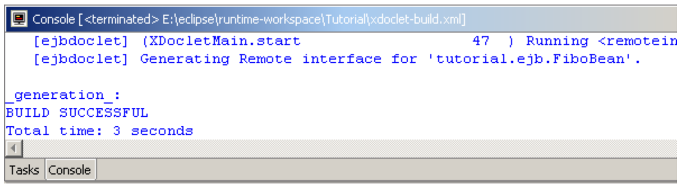
- ☑ ejbdoclet
  - ☑ deploymentdescriptor
  - ☑ fileset
  - ☑ homeinterface
  - ☑ jboss
  - ☑ packageSubstitution
  - ☑ remoteinterface

Property	Value
☐ addedTags	
☑ destDir	src
☐ ejbClassNam...	
☑ ejbSpec	2.0
☐ excludedTags	
☐ force	
☐ mergeDir	
☐ verbose	

Click "Ok" to save the generation configuration. This action will generate an Ant build file "xdoclet-build.xml" in the project. This file contains the generation configuration ready to be launch.

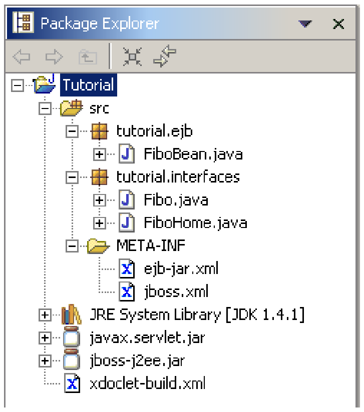


Right-click on the project and select "Run XDoclet". The XDoclet generation will display its output in the console. The output looks like this one.


Console [<terminated> E:\eclipse\runtime-workspace\Tutorial\xdoclet-build.xml]
Tasks
Console

```
    [ejbdoclet] (XDocletMain.start                  47  ) Running <remotein
    [ejbdoclet] Generating Remote interface for 'tutorial.ejb.FiboBean'.

_generation_:
BUILD SUCCESSFUL
Total time: 3 seconds
```

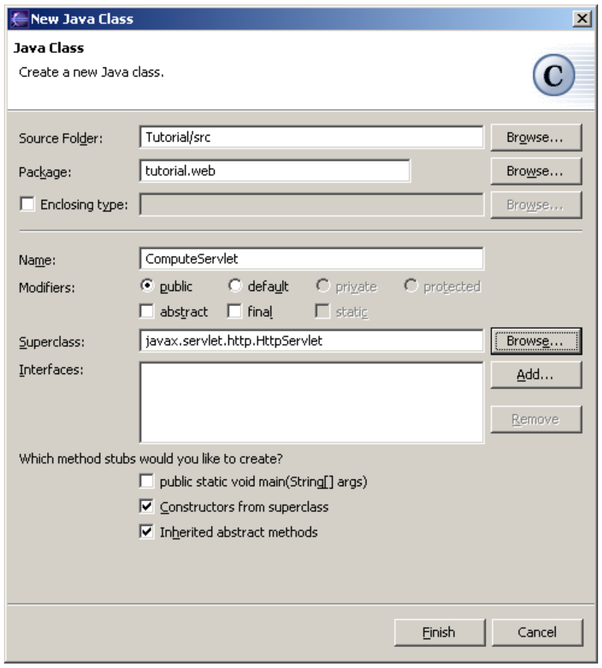
After the generation, you should have a project like this. Note that a "tutorial.interfaces" package has been created with new classes inside. There is also a "META-INF" folder with the deployment descriptors (standard and jboss ones).



The Servlet and the Web-App

Having an EJB is not enough. We will write a servlet that access this EJB to perform the actual computation of the Fibonacci suite.

Create a new Java Class. The package will be “tutorial.web” and the class name “ComputeServlet”. Click on “Browse…” to add the “HTTPServlet” superclass. Be sure that only “Constructors from superclass” and “Inherited abstract methods” are checked.


New Java Class

Java Class

Create a new Java class.

Source Folder: Tutorial/src

Package: tutorial.web

Enclosing type: [ ]

Name: ComputeServlet

Modifiers: [✓ public] default private protected; abstract [ ] final [ ] static [ ]

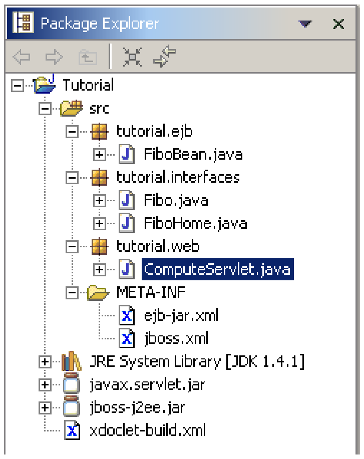
Superclass: javax.servlet.http.HttpServlet

Interfaces:

Which method stubs would you like to create?

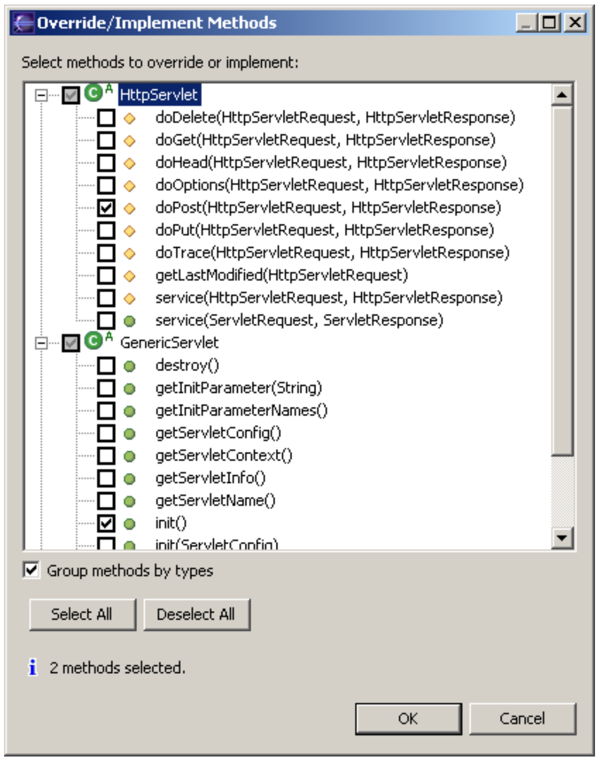
- [ ] public static void main(String[] args)
- [✓] Constructors from superclass
- [✓] Inherited abstract methods

Click on “Finish”. The class is then created and you should have a project like this.



Open the **ComputeServlet.java** file. To be interesting, our servlet some initialization and some processing code.

Right-click in the editor area and select "Source > Override/Implement Methods...". Check "init" and "doPost" methods and click "Ok". Two method stubs will be generated.


Override/Implement Methods
Select methods to override or implement:
HttpServlet
doDelete(HttpServletRequest, HttpServletResponse)
doGet(HttpServletRequest, HttpServletResponse)
doHead(HttpServletRequest, HttpServletResponse)
doOptions(HttpServletRequest, HttpServletResponse)
doPost(HttpServletRequest, HttpServletResponse)
doPut(HttpServletRequest, HttpServletResponse)
doTrace(HttpServletRequest, HttpServletResponse)
getLastModified(HttpServletRequest)
service(HttpServletRequest, HttpServletResponse)
service(ServletRequest, ServletResponse)
GenericServlet
destroy()
getInitParameter(String)
getInitParameterNames()
getServletConfig()
getServletContext()
getServletInfo()
getServletName()
init()
init(ServletConfig)
Group methods by types
Select All
Deselect All
2 methods selected.
OK
Cancel

Add the two following private members.

```
	private FiboHome home;
	private String value;
```

Define the "init" method like this. This code is responsible for the initialization of the EJB Home interface and the for grab of the environment entry.

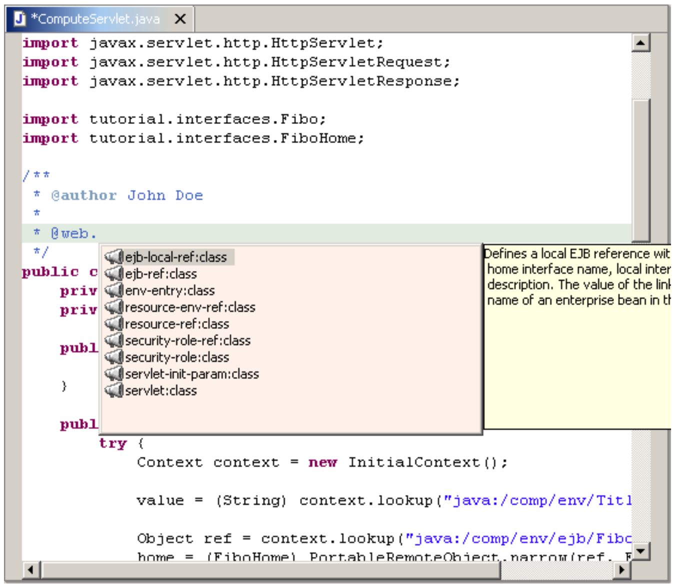
```
	public void init() throws ServletException {
		try {
			Context context = new InitialContext();

			value = (String) context.lookup("java:/comp/env/Title");

			Object ref = context.lookup("java:/comp/env/ejb/Fibo");
			home = (FiboHome) PortableRemoteObject.narrow(ref,
FiboHome.class);
		} catch (Exception e) {
			throw new ServletException("Lookup of java:/comp/env/ failed");
		}
	}
```

Define the "doPost" method like this. The code will parse the request to get the limit parameter, create an instance of the EJB, performs computation, releases the instance and output the result as HTML.

```
	protected void doPost(
		HttpServletRequest request,
		HttpServletResponse response)
		throws ServletException, IOException {
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();

		out.println("<html><head><title>");
		out.println(value);
		out.println("</title></head>");
		out.println("<body>");

		out.println("<h1>");
		out.println(value);
		out.println("</h1>");

		try {
			Fibo bean = home.create();
			int limit = 0;
			String value = request.getParameter("limit");
			if (value != null) {
				try {
					limit = Integer.parseInt(value);
				} catch (Exception e) {
				}
			}
			double[] result = bean.compute(limit);
			bean.remove();

			out.println("<p>");
			out.print("The ");
```

```
                    out.print(limit);
                    out.print(" first Fibonacci numbers ");

                    for (int i = 0; i < result.length; i++) {
                          out.println("<br>");
                          out.println(i);
                          out.println(" : ");
                          out.println(result[i]);
                    }

                    out.println("</p>");
              } catch (Exception e) {
                    out.println(e.getMessage());
                    e.printStackTrace(out);
              } finally {
                    out.println("</body></html>");
                    out.close();
              }
        }
```

The next step is to insert XDoclet javadoc related tags for the Servlet. In the Java editor go in the Javadoc class paragraph. Type "@web." And press CTRL+Space. You should see the magic of completion in action.



```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import tutorial.interfaces.Fibo;
import tutorial.interfaces.FiboHome;

/**
 * @author John Doe
 *
 * @web.
 */
public c
    priv
    priv

    publ

    }

    publ
        try {
            Context context = new InitialContext();

            value = (String) context.lookup("java:/comp/env/Titl

            Object ref = context.lookup("java:/comp/env/ejb/Fibc
            home = (FiboHome) PortableRemoteObject.narrow(ref, F
```

Defines a local EJB reference wit
home interface name, local inter
description. The value of the lin
name of an enterprise bean in th

Complete the attributes of the tag with the following values (press CTRL+Space for each attribute if you want the completion) :

```
 *
 * @web.servlet          name = "ComputeServlet"
 *                      display-name = "Computation Servlet"
 *                      description = "Servlet that compute Fibonacci suite"
 *
 * @web.servlet-mapping  url-pattern = "/Compute"
 *
 * @web.env-entry         name = "Title"
 *                        type = "java.lang.String"
 *                        value = "Fibonacci computation"
 *                        description = "Example of Env Entry"
 *
 * @web.ejb-ref           name = "ejb/Fibo"
 *                        type = "Session"
 *                        home = "tutorial.interfaces.FiboHome"
 *                        remote = "tutorial.interfaces.Fibo"
 *                        description = "Reference to the Fibo EJB"
 *
 * @jboss.ejb-ref-jndi    ref-name = "ejb/Fibo"
 *                        jndi-name = "ejb/tutorial/Fibo"
 */
public class ComputeServlet extends HttpServlet {
```

After that, the file should look like this. Now, we are ready to run XDoclet on the file generate all the Web stuff.

```
package tutorial.web;

import java.io.IOException;
import java.io.PrintWriter;

import javax.naming.Context;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import tutorial.interfaces.Fibo;
import tutorial.interfaces.FiboHome;

/**
 * @author John Doe
 *
 * @web.servlet           name = "ComputeServlet"
 *                        display-name = "Computation Servlet"
 *                        description = "Servlet that compute Fibonacci suite"
 *
 * @web.servlet-mapping   url-pattern = "/Compute"
 *
 * @web.env-entry         name = "Title"
 *                        type = "java.lang.String"
 *                        value = "Fibonacci computation"
 *                        description = "Example of Env Entry"
 *
 * @web.ejb-ref           name = "ejb/Fibo"
 *                        type = "Session"
 *                        home = "tutorial.interfaces.FiboHome"
 *                         remote = "tutorial.interfaces.Fibo"
 *                         description = "Reference to the Fibo EJB"
 *
 * @jboss.ejb-ref-jndi     ref-name = "ejb/Fibo"
 *                         jndi-name = "ejb/tutorial/Fibo"
 */
public class ComputeServlet extends HttpServlet {
      private FiboHome home;
      private String value;

      public ComputeServlet() {
            super();
      }

      public void init() throws ServletException {
            try {
                  Context context = new InitialContext();

                  value = (String) context.lookup("java:/comp/env/Title");

                  Object ref = context.lookup("java:/comp/env/ejb/Fibo");
                  home = (FiboHome) PortableRemoteObject.narrow(ref,
FiboHome.class);
            } catch (Exception e) {
                  throw new ServletException("Lookup of java:/comp/env/ failed");
            }
      }

      protected void doPost(
            HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            out.println("<html><head><title>");
            out.println(value);
            out.println("</title></head>");
            out.println("<body>");

            out.println("<h1>");
            out.println(value);
            out.println("</h1>");

            try {
                  Fibo bean = home.create();
                  int limit = 0;
                  String value = request.getParameter("limit");
                  if (value != null) {
                        try {
                              limit = Integer.parseInt(value);
                        } catch (Exception e) {
                        }
                  }
                  double[] result = bean.compute(limit);
                  bean.remove();

                  out.println("<p>");
                  out.print("The ");
                  out.print(limit);
                  out.print(" first Fibonacci numbers ");

                  for (int i = 0; i < result.length; i++) {
                        out.println("<br>");
                        out.println(i);
                        out.println(" : ");
                        out.println(result[i]);
                  }

                  out.println("</p>");
            } catch (Exception e) {
                  out.println(e.getMessage());
                  e.printStackTrace(out);
            } finally {
                  out.println("</body></html>");
                  out.close();
            }
      }
}
```

Generation of the Servlet related files

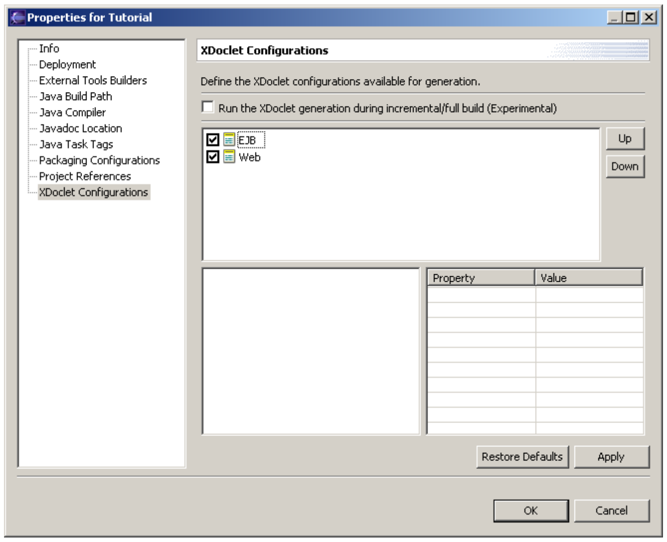
To generate the Web descriptors, we need to make some XDoclet configuration. Like EJB, we will define a generation configuration for the Web stuff.

Edit the project properties by right clicking on the project and select "Properties".

In the property page, select "XDoclet configurations".

Right-click in the upper area to pop-up the menu and choose "Add". Type "Web" in the dialog and click "Ok".

You have created a new generation configuration named "Web".


Properties for Tutorial

XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

☑ EJB
☑ Web

Select the “Web” configuration.

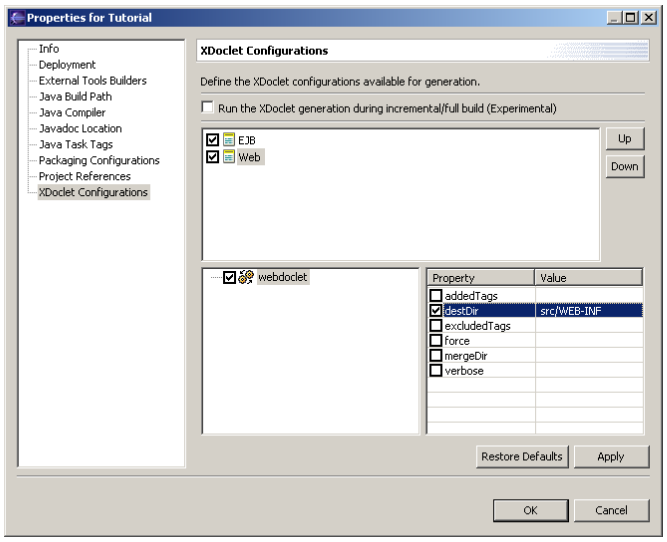
In the lower-left area, right-click to popup the menu and choose "Add Doclet".

A list of available doclets will appear. Choose "webdoclet" and click "Ok".

On the lower-right area, you see the properties of the " webdoclet ". Set them to :

- "destDir" with "src/WEB-INF " and ckeck it

Our configuration now contains an "webdoclet" that will produce files in "src/WEB-INF" folder.



Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

☑ EJB
☑ Web

☑ webdoclet

Property	Value
☐ addedTags	
☑ destDir	src/WEB-INF
☐ excludedTags	
☐ force	
☐ mergeDir	
☐ verbose	

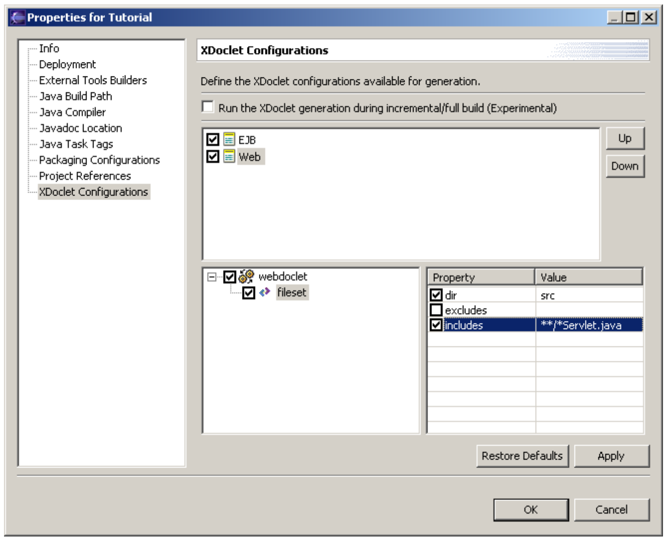
In the lower-left area, right-click on "webdoclet" to popup the menu and choose "Add ".

A list of available subtasks will appear. Choose "fileset" and click "Ok".

On the lower-right area, you see the properties of the " fileset ". Set them to :

- "dir" with "src" and ckeck it
- uncheck "excludes"
- "includes" with "**/*Servlet.java" and ckeck it

Our configuration now contains an "webdoclet" with a "fileset" based on "src" and that filters source files to only pick up the Servlet ones.



Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

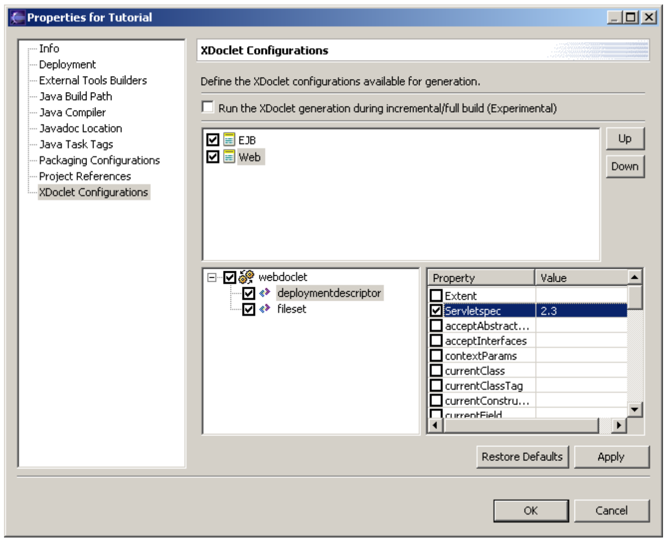
☑ EJB
☑ Web

☑ webdoclet
☑ fileset

Property	Value
☑ dir	src
☐ excludes	
☑ includes	**/*Servlet.java

Repeat the operation to add a new subtask "deploymentdescriptor". The property to set is "Servletspec" with "2.3". Don't forget to check it.

We will generate the deployment descriptor in the "src/WEB-INF" folder to remain simple (property inherit from "webdoclet").



XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

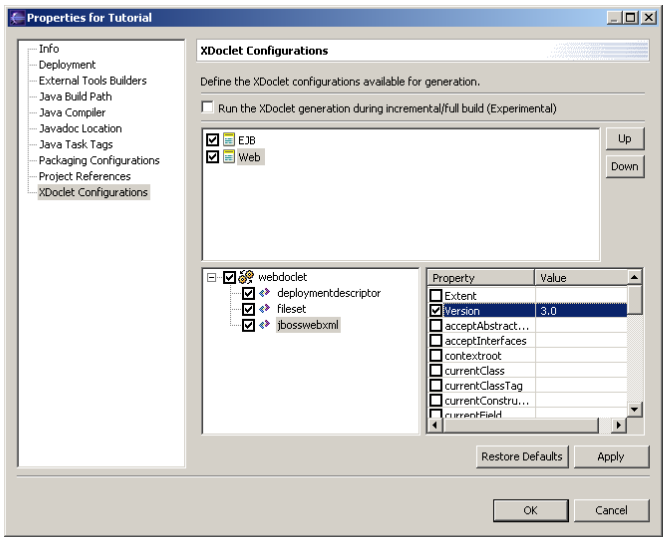
☑ EJB
☑ Web

☑ webdoclet
  ☑ deploymentdescriptor
  ☑ fileset

Property	Value
☐ Extent	
☑ Servletspec	2.3
☐ acceptAbstract...	
☐ acceptInterfaces	
☐ contextParams	
☐ currentClass	
☐ currentClassTag	
☐ currentConstru...	
☐ currentField	

Repeat the operation to add a new subtask "jbosswebxml". The property to set is "Version" with "3.0". Don't forget to check it.

We will generate the jboss deployment descriptor in the "src/WEB-INF" folder (property inherit from "webdoclet").



XDoclet Configurations

Define the XDoclet configurations available for generation.

☐ Run the XDoclet generation during incremental/full build (Experimental)

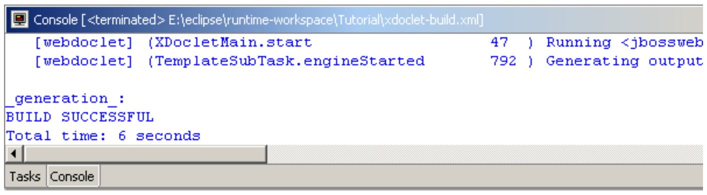
☑ EJB
☑ Web

☑ webdoclet
  ☑ deploymentdescriptor
  ☑ fileset
  ☑ jbosswebxml

Property	Value
☐ Extent	
☑ Version	3.0
☐ acceptAbstract...	
☐ acceptInterfaces	
☐ contextroot	
☐ currentClass	
☐ currentClassTag	
☐ currentConstru...	
☐ currentField	

Click "Ok" to save the generation configuration. This action will update the Ant build file "xdoclet-build.xml" in the project. This file now contains the generation configurations for the EJB and for the Servlet.

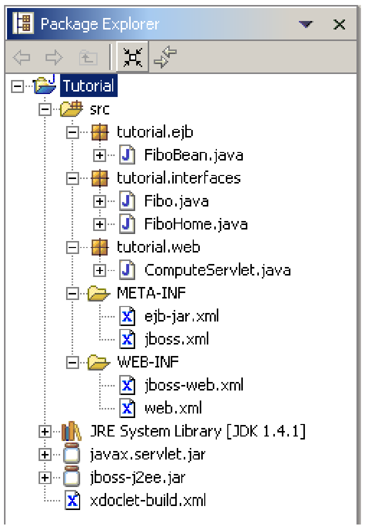
Right-click on the project and select "Run XDoclet". The XDoclet generation will display its output in the console. The output looks like this one.



Console [<terminated> E:\eclipse\runtime-workspace\Tutorial\xdoclet-build.xml]

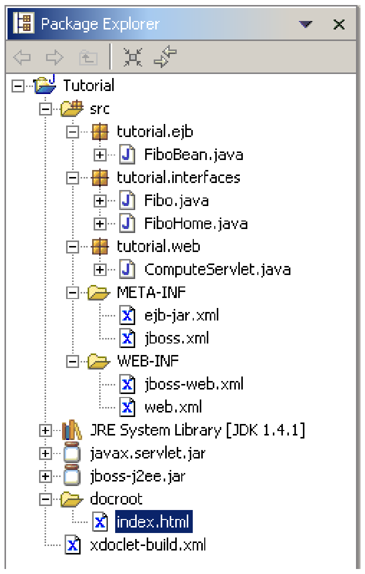
```
   [webdoclet] (XDocletMain.start            47  ) Running <jbossweb
   [webdoclet] (TemplateSubTask.engineStarted 792 ) Generating output

_generation_:
BUILD SUCCESSFUL
Total time: 6 seconds
```

After the generation, you should have a project like this. Note that a "WEB-INF" folder has been created with the Web deployment descriptor (standard and jboss ones).



A Servlet is useless unless some parameters are passed. To pass these parameters to the Servlet, a HTML page with a form is needed.

Create a "docroot" folder under the root of the project. Create an empty file named "index.html". The "index.html" file is intended to be the default page for the Web application and contains a form that will be posted to the Servlet.



The content of "index.html" is the following :

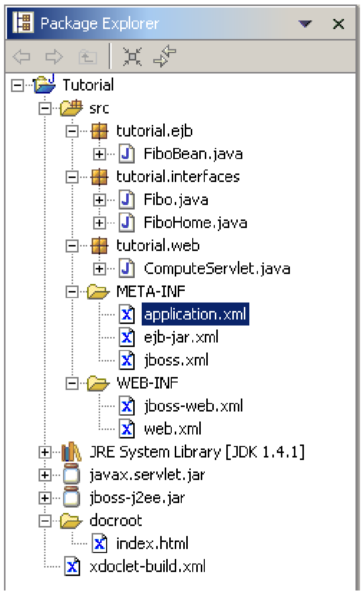
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
   <head>
      <title>
         Fibonacci Application
      </title>
```

```
    </head>
    <body>
       <h1>Fibonacci Form</h1>
       <form action="Compute" method="POST" >
          <table cellspacing="2" cellpadding="2" border="0">
             <tr>
                <td>
                   Limit :
                </td>
                <td>
                   <input type="text" name="limit" value="50">
                </td>
             </tr>
             <tr>
                <td>
                   <input type="submit" name="Compute" value="Compute">
                </td>
                <td>
                   <input type="Reset">
                </td>
             </tr>
          </table>
       </form>
    </body>
</html>
```

The J2EE Application

This project is intended to be a complete J2EE application. We are going to create some additionnal files to have all the material needed to build it.

In the "src/META-INF" folder, create a file named "application.xml". The "application.xml" file is the J2EE application descriptor that will point to the EJB package and to the War package.



```
Tutorial
  src
    tutorial.ejb
      FiboBean.java
    tutorial.interfaces
      Fibo.java
      FiboHome.java
    tutorial.web
      ComputeServlet.java
    META-INF
      application.xml
      ejb-jar.xml
      jboss.xml
    WEB-INF
      jboss-web.xml
      web.xml
  JRE System Library [JDK 1.4.1]
  javax.servlet.jar
  jboss-j2ee.jar
  docroot
    index.html
  xdoclet-build.xml
```

The content of "application.xml" is the following :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application
1.3//EN" "http://java.sun.com/dtd/application_1_3.dtd">
<application>
   <display-name>Sum Application</display-name>
   <module>
      <ejb>FiboEJB.jar</ejb>
   </module>
   <module>
      <web>
         <web-uri>FiboWeb.war</web-uri>
         <context-root>/fibo</context-root>
      </web>
   </module>
</application>
```

The Packaging

JBoss-IDE provides an easy way to configure the packaging of various archive. There is no restriction of what can be packaged.

In this tutorial, four packaging configuration will be defined:

- One for the EJB Jar. It will contain the EJB classes and interfaces, as well as the deployment descriptors.
- One for the EJB client Jar. It will contain the EJB interfaces.
- One for the Web Application War. It will contain the Servlet class, the EJB client Jar, as well as the deployment descriptors.
- One for the J2EE application Ear. It will contain the EJB Jar and the Web Application War, as well as the deployment descriptor.

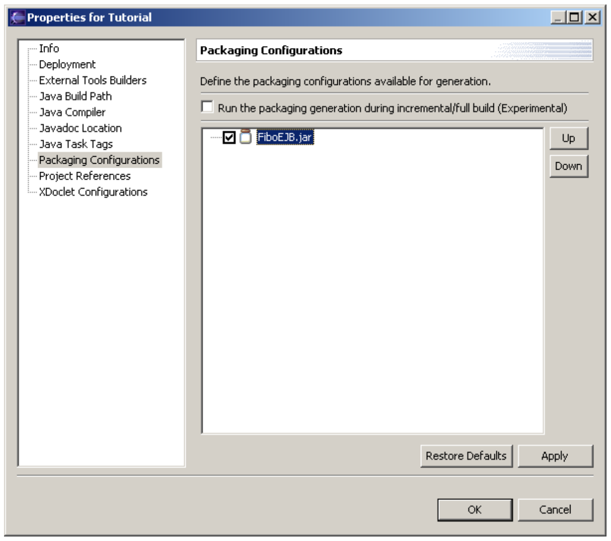
When launched, these four packaging configurations will create the J2EE application ready to be deployed.

Edit the project properties by right clicking on the project and select "Properties".

In the property page, select "Packaging configurations".

Right-click in the area to pop-up the menu and choose "Add Archive". Type "FiboEJB.jar" in the dialog and click "Ok".

You have created a new packging configuration that will produce the "FiboEJB.jar" file.



Packaging Configurations

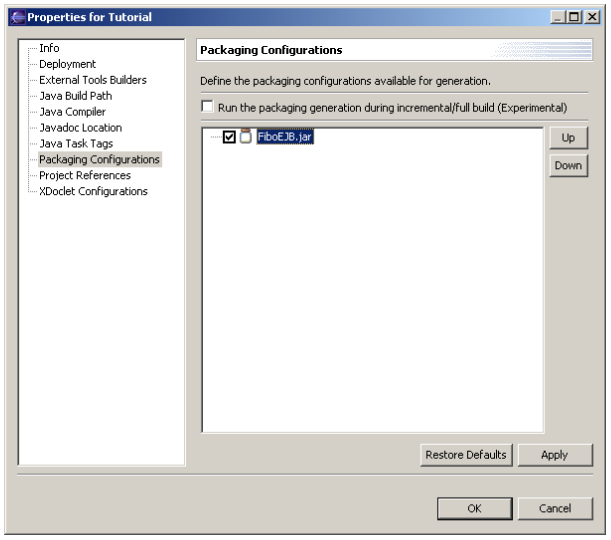
Define the packaging configurations available for generation.

☐ Run the packaging generation during incremental/full build (Experimental)

☑ FiboEJB.jar

We want to add the EJB classes and interfaces. Eclipse has generated the compiled classes into the src/bin folder (declared as the default output dir of the project).

Select the "FiboEJB.jar" item and right-click in the area to pop-up the menu and choose "Add Folder". A "Folder Selection" dialog appears.



Packaging Configurations

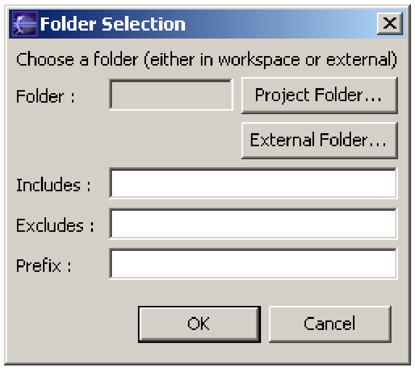
Define the packaging configurations available for generation.

☐ Run the packaging generation during incremental/full build (Experimental)

☑ FiboEJB.jar

This dialog allows to select which folder (local to workspace or in the file system) to include into the package, to specify include and exclude filters (A la Ant) and to set a prefix that will be append when building the package.

Click on "Project Folder". A "Folder Chooser" dialog appears.



Choose a folder (either in workspace or external)

Folder :

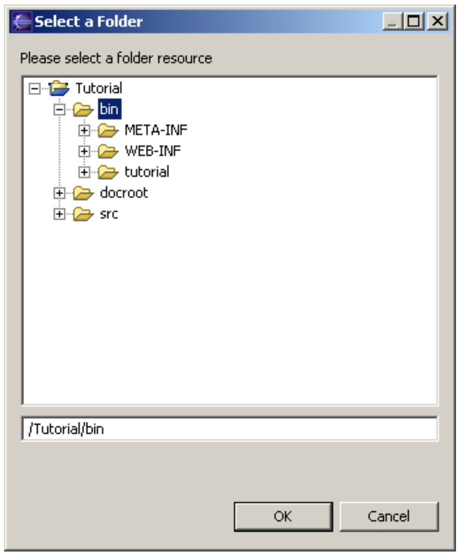
Includes :

Excludes :

Prefix :

This dialog allows selecting which folder to include. This folder can be choosen among all the opened projects.

Select the “/Tutorial/bin” folder and click “Ok”.

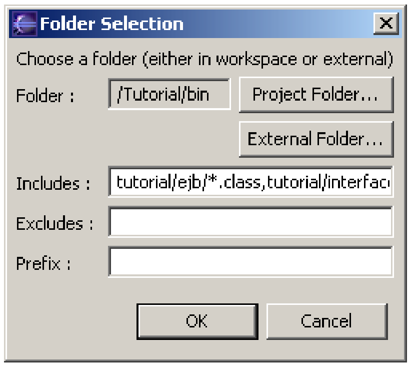


The folder is now “/Tutorial/bin”.

As we only want the EJB classes and interfaces, specify the following as include filter:

“tutorial/ejb/*.class,tutorial/interfaces/*.class”

Click on “Ok”.

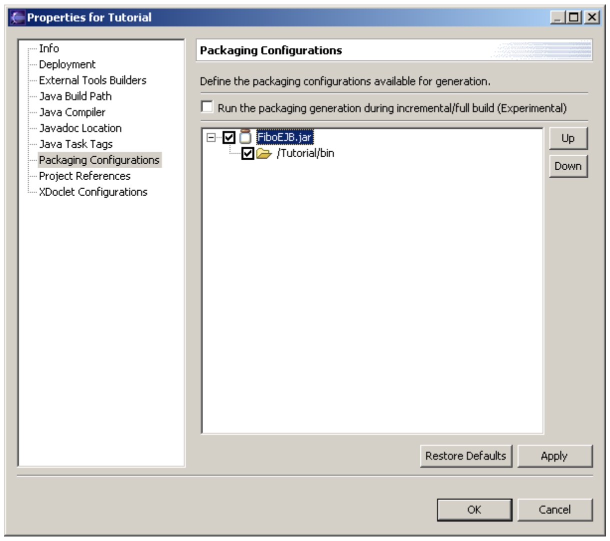


Folder : /Tutorial/bin

Includes : tutorial/ejb/*.class,tutorial/interfac

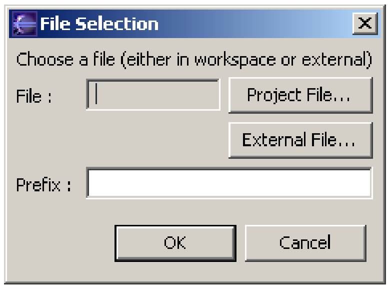
We now want to add the standard EJB deployment descriptor.

Select the “FiboEJB.jar” item and right-click in the area to pop-up the menu and choose "Add File". A “File Selection” dialog appears.



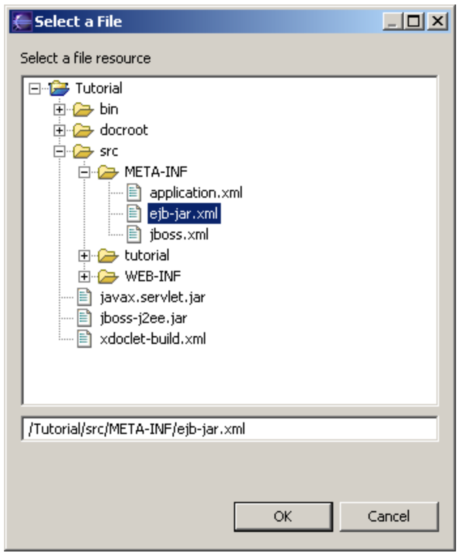
This dialog allows to select which file (local to workspace or in the file system) to include into the package and to set a prefix which will be append when building the package.

Click on “Project File”. A “File Chooser” dialog appears.



This dialog allows to select which file to include. This file can be choosen among all the opened projects.

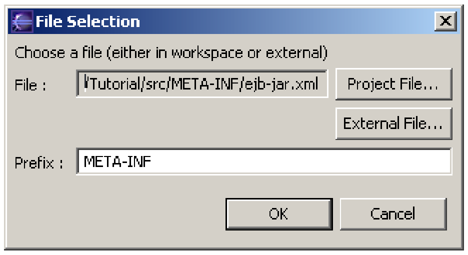
Select the “/Tutorial/src/META-INF/ejb-jar.xml” folder and click “Ok”.



The file is now "/Tutorial/src/META-INF/ejb-jar.xml".

The "ejb-jar.xml" must be located under the "META-INF" of the EJB package. Set the prefix to "META-INF".

Click on "Ok".


File Selection

Choose a file (either in workspace or external)

File : /Tutorial/src/META-INF/ejb-jar.xml

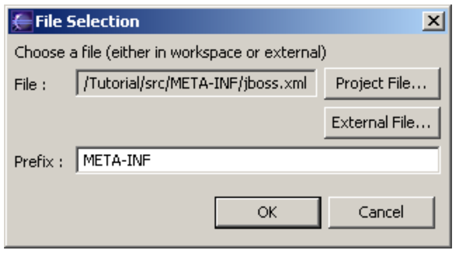
Prefix : META-INF

To add the specific EJB deployment descriptor, select the "FiboEJB.jar" item and right-click in the area to pop-up the menu and choose "Add File".

The file to choose is "/Tutorial/src/META-INF/jboss.xml".

The "jboss.xml" must be located under the "META-INF" of the EJB package. Set the prefix to "META-INF".

Click on "Ok".

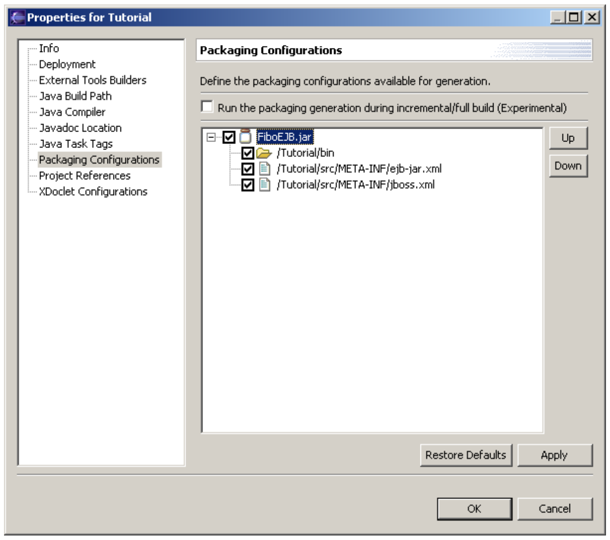

File Selection

Choose a file (either in workspace or external)

File : /Tutorial/src/META-INF/jboss.xml

Prefix : META-INF

The packaging configuration for the "FiboEJB.jar" is now completed.


Properties for Tutorial

Packaging Configurations

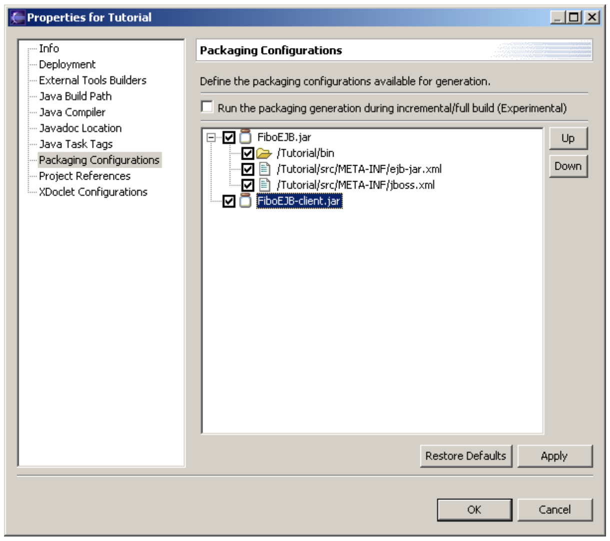
Define the packaging configurations available for generation.

Run the packaging generation during incremental/full build (Experimental)

```
FiboEJB.jar
    /Tutorial/bin
    /Tutorial/src/META-INF/ejb-jar.xml
    /Tutorial/src/META-INF/jboss.xml
```

Right-click in the area to pop-up the menu and choose "Add Archive". Type "FiboEJB-client.jar" in the dialog and click "Ok".

You have created a new packging configuration that will produce the “FiboEJB-client.jar” file.



Packaging Configurations

Define the packaging configurations available for generation.

☐ Run the packaging generation during incremental/full build (Experimental)

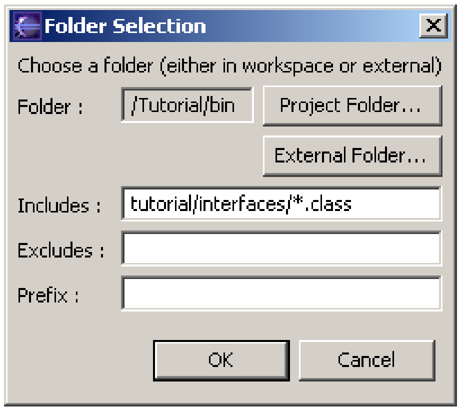
```
☑ FiboEJB.jar
    ☑ /Tutorial/bin
    ☑ /Tutorial/src/META-INF/ejb-jar.xml
    ☑ /Tutorial/src/META-INF/jboss.xml
☑ FiboEJB-client.jar
```

Select the “FiboEJB-client.jar” item and right-click in the area to pop-up the menu and choose "Add Folder". A “Folder Selection” dialog appears.

Click on “Project Folder” and select the “/Tutorial/bin” folder from the “Folder Chooser” dialog.

As we only want the EJB interfaces, set the include filter to “tutorial/interfaces/*.class”.

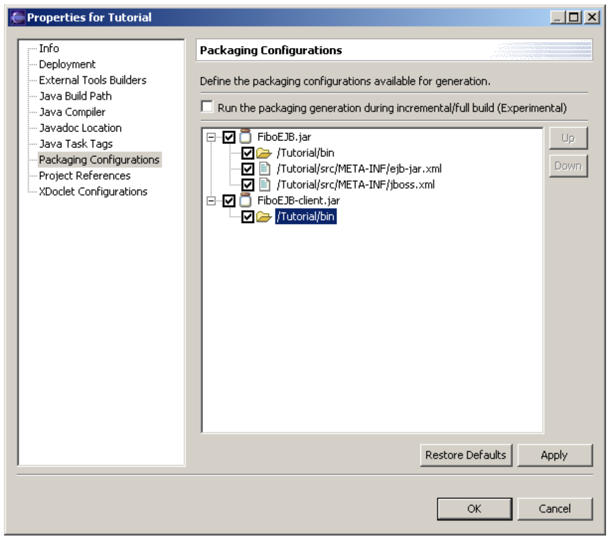
Click on “Ok”.



Choose a folder (either in workspace or external)

Folder :	/Tutorial/bin
Includes :	tutorial/interfaces/*.class
Excludes :	
Prefix :	

The packaging configuration for the “FiboEJB-client.jar” is now completed.



Packaging Configurations

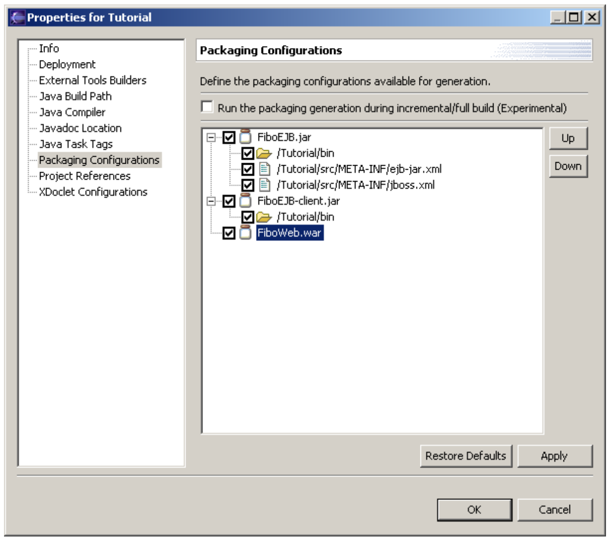
Define the packaging configurations available for generation.

☐ Run the packaging generation during incremental/full build (Experimental)

```
☑ FiboEJB.jar
    ☑ /Tutorial/bin
    ☑ /Tutorial/src/META-INF/ejb-jar.xml
    ☑ /Tutorial/src/META-INF/jboss.xml
☑ FiboEJB-client.jar
    ☑ /Tutorial/bin
```

Right-click in the area to pop-up the menu and choose "Add Archive". Type "FiboWeb.war" in the dialog and click "Ok".

You have created a new packging configuration that will produce the “FiboWeb.war” file.


Properties for Tutorial

Packaging Configurations

Define the packaging configurations available for generation.

☐ Run the packaging generation during incremental/full build (Experimental)

- ☑ FiboEJB.jar
  - ☑ /Tutorial/bin
  - ☑ /Tutorial/src/META-INF/ejb-jar.xml
  - ☑ /Tutorial/src/META-INF/jboss.xml
- ☑ FiboEJB-client.jar
  - ☑ /Tutorial/bin
- ☑ FiboWeb.war

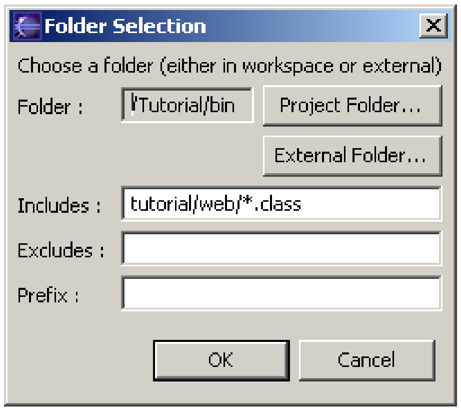
Select the “FiboWeb.war” item and right-click in the area to pop-up the menu and choose "Add Folder". A “Folder Selection” dialog appears.

Click on “Project Folder” and select the “/Tutorial/bin” folder from the “Folder Chooser” dialog.

As we only want the Servlet class, set the include filter to “tutorial/web/*.class”.

The classes must be located under the “WEB-INF/classes” of the War package. Set the prefix to “WEB-INF/classes”.

Click on “Ok”.


Folder Selection

Choose a folder (either in workspace or external)

Folder : /Tutorial/bin

Includes : tutorial/web/*.class

Excludes :

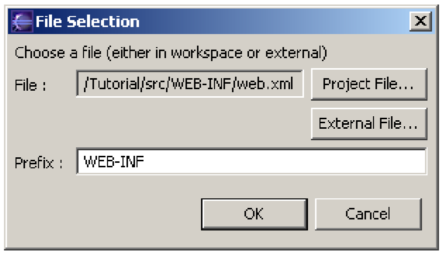
Prefix :

To add the standard Web deployment descriptor, select the “FiboWeb.war” item and right-click in the area to pop-up the menu and choose "Add File". A “File Selection” dialog appears.

The file to choose is “/Tutorial/src/WEB-INF/web.xml”.

The “web.xml” must be located under the “WEB-INF” of the War package. Set the prefix to “WEB-INF”.

Click on “Ok”.


File Selection

Choose a file (either in workspace or external)

File : /Tutorial/src/WEB-INF/web.xml

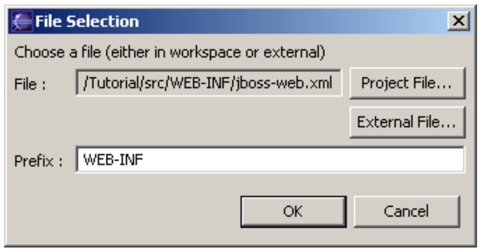
Prefix : WEB-INF

To add the specific Web deployment descriptor, select the “FiboWeb.war” item and right-click in the area to pop-up the menu and choose "Add File". A “File Selection” dialog appears.

The file to choose is “/Tutorial/src/WEB-INF/jboss-web.xml”.

The “jboss-web.xml” must be located under the “WEB-INF” of the War package. Set the prefix to “WEB-INF”.

Click on “Ok”.



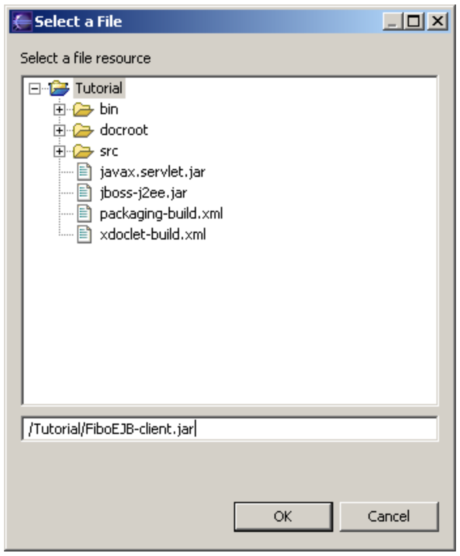
Choose a file (either in workspace or external)

File : /Tutorial/src/WEB-INF/jboss-web.xml

Prefix : WEB-INF

To add the EJB Client Jar, select the “FiboWeb.war” item and right-click in the area to pop-up the menu and choose "Add File". A “File Selection” dialog appears.

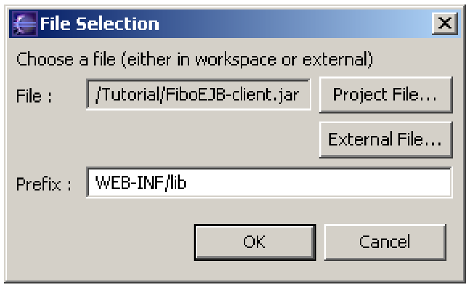
The file to choose is “/Tutorial/FiboEJB-client.jar”. But it doesn’t exist yet as the packaging has not been run. Instead of selecting it, go in the text field and type the name of the file “/Tutorial/FiboEJB-client.jar”. Even if the file doesn’t exist, it can be added to a packaging configuration.



/Tutorial/FiboEJB-client.jar

The “FiboEJB-client.jar” must be located under the “WEB-INF/lib” of the War package. Set the prefix to “WEB-INF/lib”.

Click on “Ok”.



Choose a file (either in workspace or external)

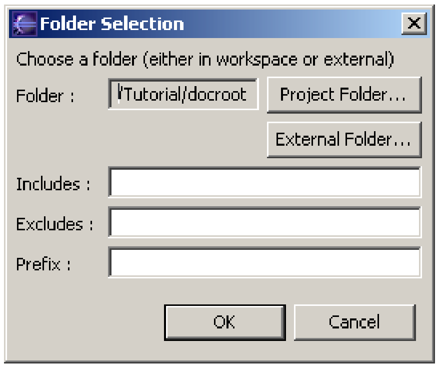
File : /Tutorial/FiboEJB-client.jar

Prefix : WEB-INF/lib

Select the “FiboWeb.war” item and right-click in the area to pop-up the menu and choose "Add Folder". A “Folder Selection” dialog appears.

Click on “Project Folder” and select the “/Tutorial/docroot” folder from the “Folder Chooser” dialog. This is the content of the Web Application.

Click on “Ok”.



Choose a folder (either in workspace or external)

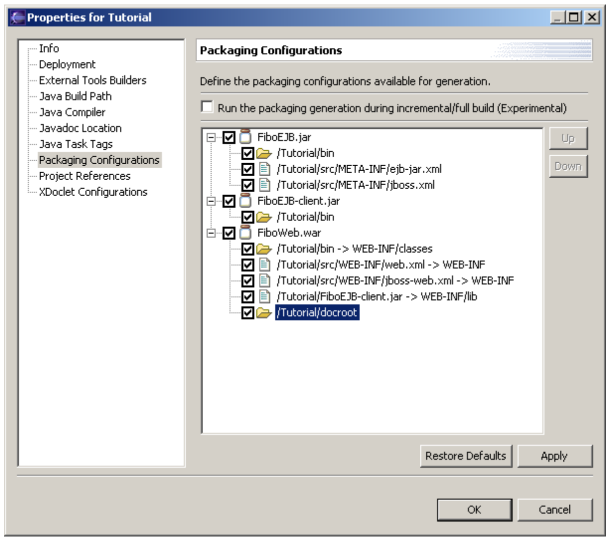
Folder : /Tutorial/docroot

Includes :

Excludes :

Prefix :

The packaging configuration for the “FiboWeb.war” is now completed.



Packaging Configurations

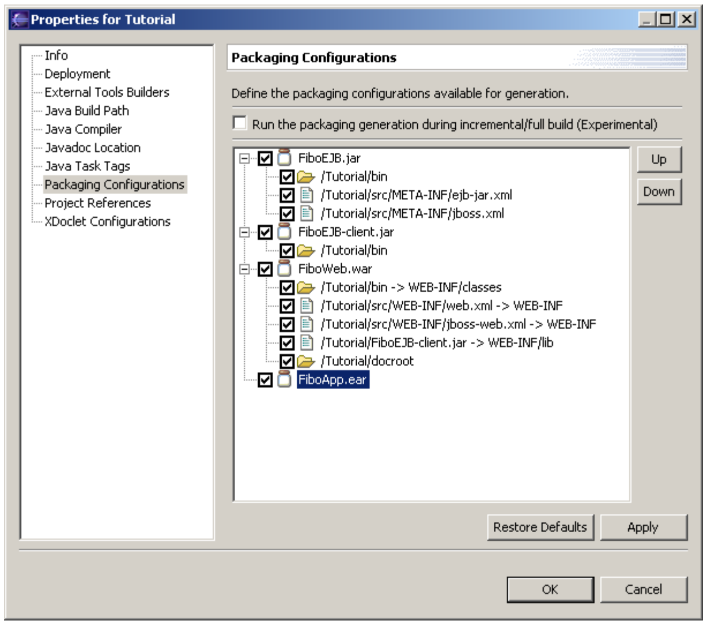
Define the packaging configurations available for generation.

☐ Run the packaging generation during incremental/full build (Experimental)

```
☑ FiboEJB.jar
    ☑ /Tutorial/bin
    ☑ /Tutorial/src/META-INF/ejb-jar.xml
    ☑ /Tutorial/src/META-INF/jboss.xml
☑ FiboEJB-client.jar
    ☑ /Tutorial/bin
☑ FiboWeb.war
    ☑ /Tutorial/bin -> WEB-INF/classes
    ☑ /Tutorial/src/WEB-INF/web.xml -> WEB-INF
    ☑ /Tutorial/src/WEB-INF/jboss-web.xml -> WEB-INF
    ☑ /Tutorial/FiboEJB-client.jar -> WEB-INF/lib
    ☑ /Tutorial/docroot
```

Right-click in the area to pop-up the menu and choose "Add Archive". Type "FiboApp.ear" in the dialog and click "Ok".

You have created a new packging configuration that will produce the "FiboApp.ear" file.



Define the packaging configurations available for generation.

☐ Run the packaging generation during incremental/full build (Experimental)

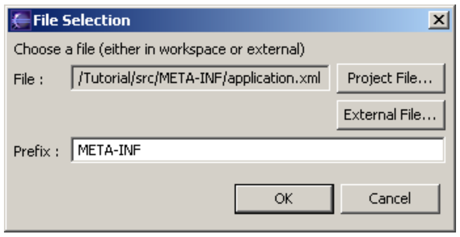
```
☑ FiboEJB.jar
    ☑ /Tutorial/bin
    ☑ /Tutorial/src/META-INF/ejb-jar.xml
    ☑ /Tutorial/src/META-INF/jboss.xml
☑ FiboEJB-client.jar
    ☑ /Tutorial/bin
☑ FiboWeb.war
    ☑ /Tutorial/bin -> WEB-INF/classes
    ☑ /Tutorial/src/WEB-INF/web.xml -> WEB-INF
    ☑ /Tutorial/src/WEB-INF/jboss-web.xml -> WEB-INF
    ☑ /Tutorial/FiboEJB-client.jar -> WEB-INF/lib
    ☑ /Tutorial/docroot
☑ FiboApp.ear
```

To add the application deployment descriptor, select the "FiboApp.ear" item and right-click in the area to pop-up the menu and choose "Add File". A "File Selection" dialog appears.

The file to choose is "/Tutorial/src/META-INF/application.xml".

The "application.xml" must be located under the "META -INF" of the Ear package. Set the prefix to "META -INF".

Click on "Ok".



Choose a file (either in workspace or external)

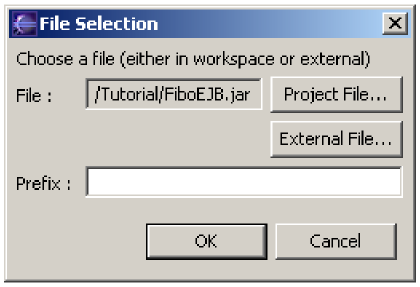
File : /Tutorial/src/META-INF/application.xml

Prefix : META-INF

To add the EJB module, select the "FiboApp.ear" item and right-click in the area to pop-up the menu and choose "Add File". A "File Selection" dialog appears.

The file to choose is "/Tutorial/FiboEJB.jar". But it doesn't exist yet as the packaging has not been run. Instead of selecting it, go in the text field and type the name of the file "/Tutorial/FiboEJB.jar". Even if the file doesn't exist, it can be added to a packaging configuration.

Click on "Ok".



Choose a file (either in workspace or external)

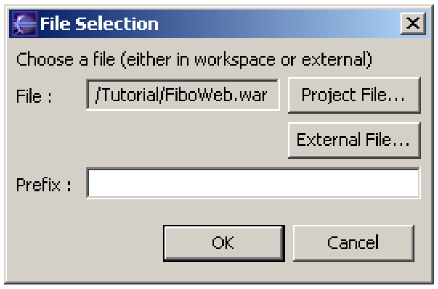
File : /Tutorial/FiboEJB.jar

Prefix :

To add the Webmodule, select the “FiboApp.ear” item and right-click in the area to pop-up the menu and choose "Add File". A “File Selection” dialog appears.

The file to choose is “/Tutorial/FiboWeb.war”. But it doesn’t exist yet as the packaging has not been run. Instead of selecting it, go in the text field and type the name of the file “/Tutorial/ FiboWeb.war”. Even if the file doesn’t exist, it can be added to a packaging configuration.

Click on “Ok”.

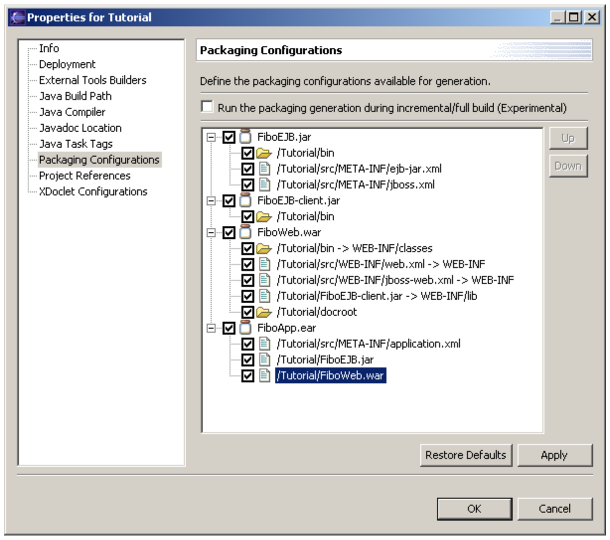


Choose a file (either in workspace or external)

File : /Tutorial/FiboWeb.war

Prefix :

The packaging configuration for the “FiboApp.ear” is now completed.



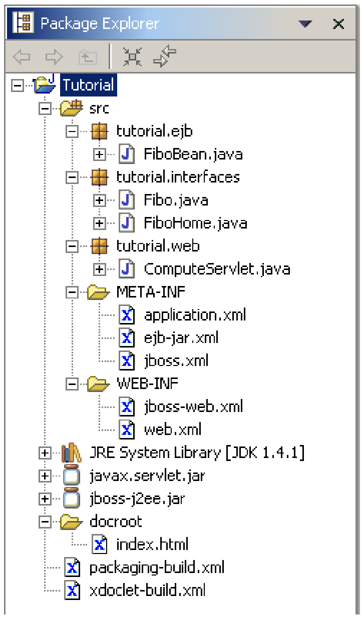
Packaging Configurations

Define the packaging configurations available for generation.

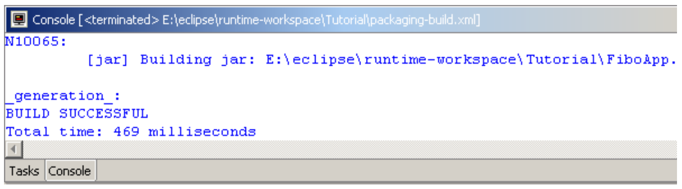
Run the packaging generation during incremental/full build (Experimental)

```
FiboEJB.jar
    /Tutorial/bin
    /Tutorial/src/META-INF/ejb-jar.xml
    /Tutorial/src/META-INF/jboss.xml
FiboEJB-client.jar
    /Tutorial/bin
FiboWeb.war
    /Tutorial/bin -> WEB-INF/classes
    /Tutorial/src/WEB-INF/web.xml -> WEB-INF
    /Tutorial/src/WEB-INF/jboss-web.xml -> WEB-INF
    /Tutorial/FiboEJB-client.jar -> WEB-INF/lib
    /Tutorial/docroot
FiboApp.ear
    /Tutorial/src/META-INF/application.xml
    /Tutorial/FiboEJB.jar
    /Tutorial/FiboWeb.war
```

Click "Ok" to save the packaging configuration. This action will create the Ant build file "packaging-build.xml" in the project. This file now contains the packaging configurations for the full J2EE Application.

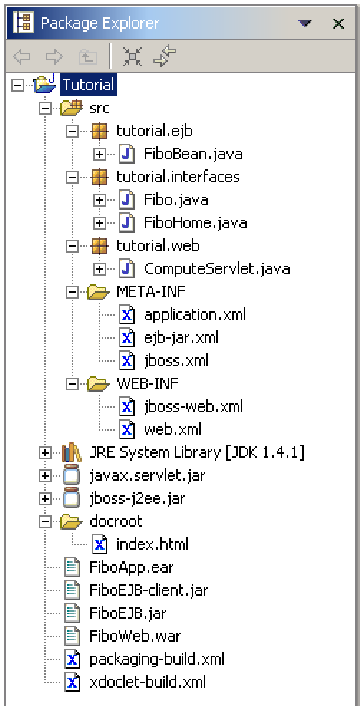


Right-click on the project and select "Run Packaging". The packaging will display its output in the console. The output looks like this one.


Console [<terminated> E:\eclipse\runtime-workspace\Tutorial\packaging-build.xml]
Tasks
Console

```
N10065:
         [jar] Building jar: E:\eclipse\runtime-workspace\Tutorial\FiboApp.

_generation_:
BUILD SUCCESSFUL
Total time: 469 milliseconds
```

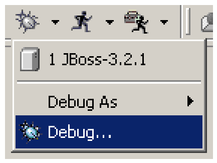
After the execution, you should have a project like this.



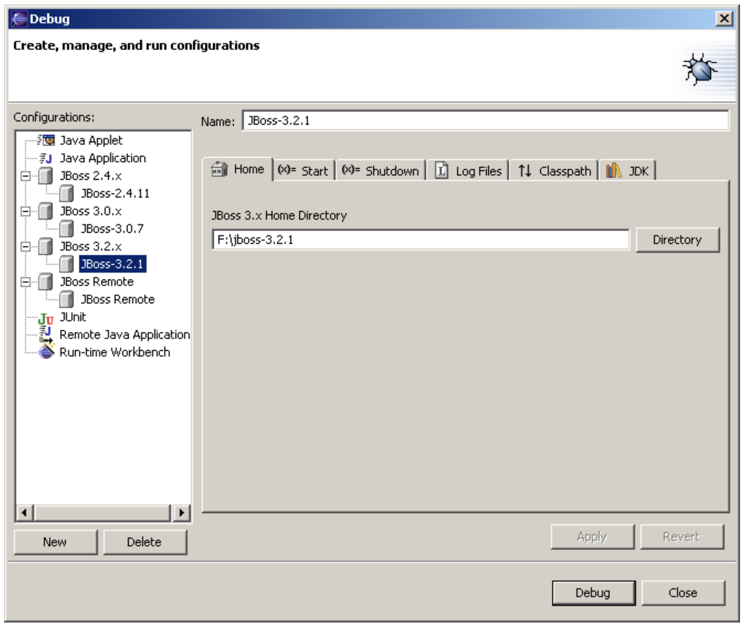
JBoss Configuration and Launch

Now, it is time to configure the JBoss server if not done.

Click on the debug shortcut and select "Debug…" to open the debug configurations.



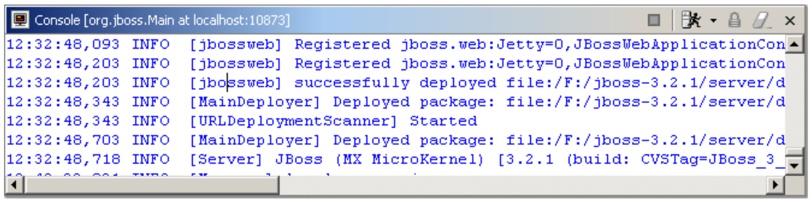
The debug dialog allow to configure the available JBoss configurations that will be used for debugging.



Name: JBoss-3.2.1

JBoss 3.x Home Directory: F:\jboss-3.2.1

Select the configuration you want to launch and click on "Debug" and you will see JBoss starting, the output is made in the console.



```
12:32:48,093 INFO  [jbossweb] Registered jboss.web:Jetty=0,JBossWebApplicationCon
12:32:48,203 INFO  [jbossweb] Registered jboss.web:Jetty=0,JBossWebApplicationCon
12:32:48,203 INFO  [jbossweb] successfully deployed file:/F:/jboss-3.2.1/server/d
12:32:48,343 INFO  [MainDeployer] Deployed package: file:/F:/jboss-3.2.1/server/d
12:32:48,343 INFO  [URLDeploymentScanner] Started
12:32:48,703 INFO  [MainDeployer] Deployed package: file:/F:/jboss-3.2.1/server/d
12:32:48,718 INFO  [Server] JBoss (MX MicroKernel) [3.2.1 (build: CVSTag=JBoss_3_
```

Deployment

The deployment within JBoss-IDE can be done in two ways:

- A file-system copy. It is like copy and paste the resource from a file explorer.
- A local deployment through the MainDeployer MBean (Experimental). The URL of the resource is sent to the MainDeployer Mbean, which deploys and watches it.

In addition, the deployment target is stored during the workbench session. This means that if you have deployed a package on a target, you can redeploy or undeploy it without specifying the target.

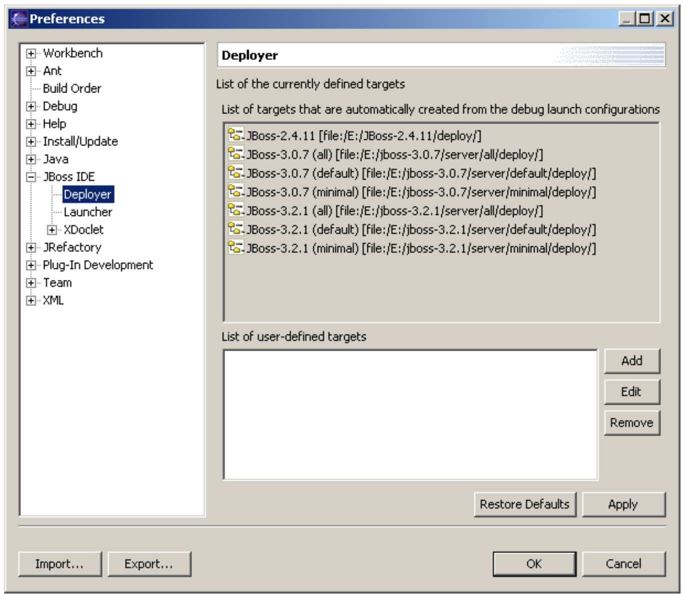
The Deployer plugin automatically creates file-system targets from the debug configurations. Other deployment target can be defined.

Select "Window > Preferences". The workbench preferences appears.

Select "JBoss-IDE > Deployer" to display the defined deployment targets.

The upper area contains the file system targets build upon the debug configuration defined.

The lower area contains the user-defined deployment targets.



List of the currently defined targets

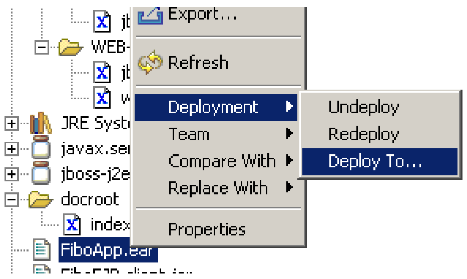
List of targets that are automatically created from the debug launch configurations

```
JBoss-2.4.11 [file:/E:/JBoss-2.4.11/deploy/]
JBoss-3.0.7 (all) [file:/E:/jboss-3.0.7/server/all/deploy/]
JBoss-3.0.7 (default) [file:/E:/jboss-3.0.7/server/default/deploy/]
JBoss-3.0.7 (minimal) [file:/E:/jboss-3.0.7/server/minimal/deploy/]
JBoss-3.2.1 (all) [file:/E:/jboss-3.2.1/server/all/deploy/]
JBoss-3.2.1 (default) [file:/E:/jboss-3.2.1/server/default/deploy/]
JBoss-3.2.1 (minimal) [file:/E:/jboss-3.2.1/server/minimal/deploy/]
```

List of user-defined targets

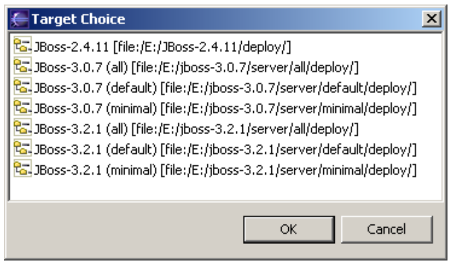
We assume that we want to deploy to a pre-defined JBoss instance and we don't define custom deployment targets.

The deployment is fairly simple. Right click on the "FiboApp.ear" file and select the "Deployment" > "Deploy To..." item.



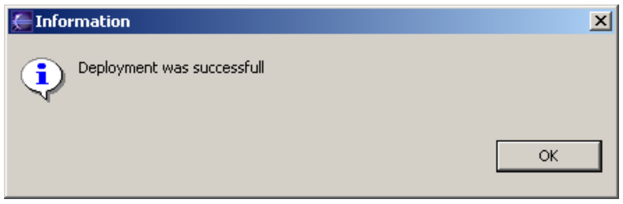
A dialog box appears with the list of the deployment targets. It contains both the default and the user-defined deployment targets.

Select the one you are interested in.



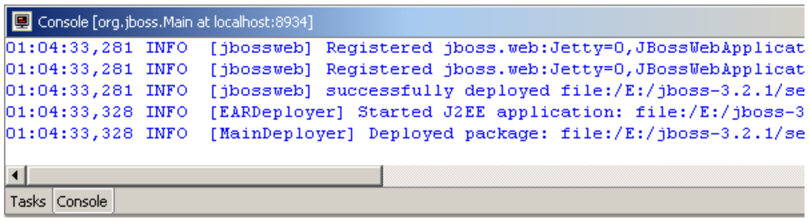
```
JBoss-2.4.11 [file:/E:/JBoss-2.4.11/deploy/]
JBoss-3.0.7 (all) [file:/E:/jboss-3.0.7/server/all/deploy/]
JBoss-3.0.7 (default) [file:/E:/jboss-3.0.7/server/default/deploy/]
JBoss-3.0.7 (minimal) [file:/E:/jboss-3.0.7/server/minimal/deploy/]
JBoss-3.2.1 (all) [file:/E:/jboss-3.2.1/server/all/deploy/]
JBoss-3.2.1 (default) [file:/E:/jboss-3.2.1/server/default/deploy/]
JBoss-3.2.1 (minimal) [file:/E:/jboss-3.2.1/server/minimal/deploy/]
```

If the deployment is successful, a dialog should pop-up to tell it.



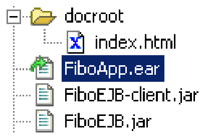
Deployment was successfull

In the console view, you should see some deployment activity. The J2EE application is now deployed.



```
01:04:33,281 INFO  [jbossweb] Registered jboss.web:Jetty=0,JBossWebApplicat
01:04:33,281 INFO  [jbossweb] Registered jboss.web:Jetty=0,JBossWebApplicat
01:04:33,281 INFO  [jbossweb] successfully deployed file:/E:/jboss-3.2.1/se
01:04:33,328 INFO  [EARDeployer] Started J2EE application: file:/E:/jboss-3
01:04:33,328 INFO  [MainDeployer] Deployed package: file:/E:/jboss-3.2.1/se
```

When a resource is deployed, a small decorator appears in the top-left corner of the icon.

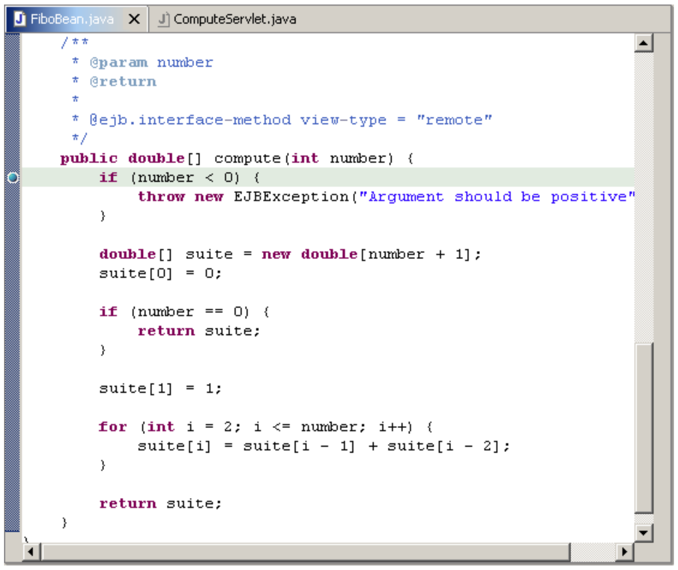


Debugging

Prior to the debugging, we need to set some breakpoints inside the code.

Open the FiboBean.java file. Double click in left column to create a breakpoint.

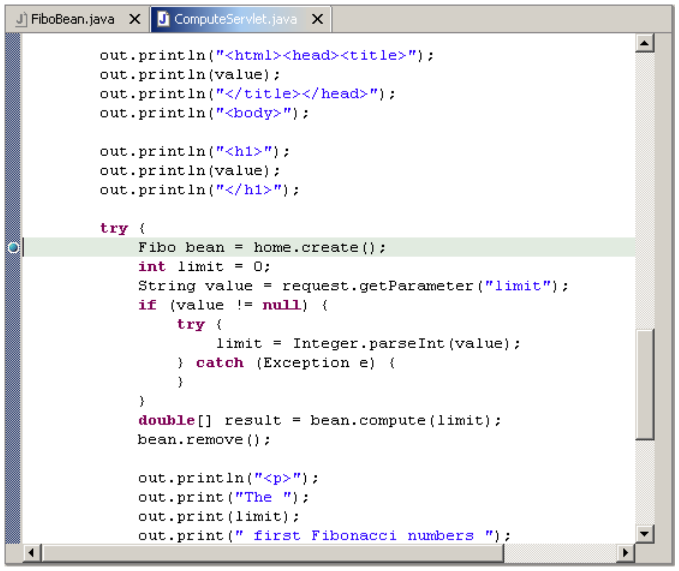
In the example, the breakpoint is set in front of the test.



```
    /**
     * @param number
     * @return
     *
     * @ejb.interface-method view-type = "remote"
     */
    public double[] compute(int number) {
        if (number < 0) {
            throw new EJBException("Argument should be positive"
        }

        double[] suite = new double[number + 1];
        suite[0] = 0;

        if (number == 0) {
            return suite;
        }

        suite[1] = 1;

        for (int i = 2; i <= number; i++) {
            suite[i] = suite[i - 1] + suite[i - 2];
        }

        return suite;
    }
```

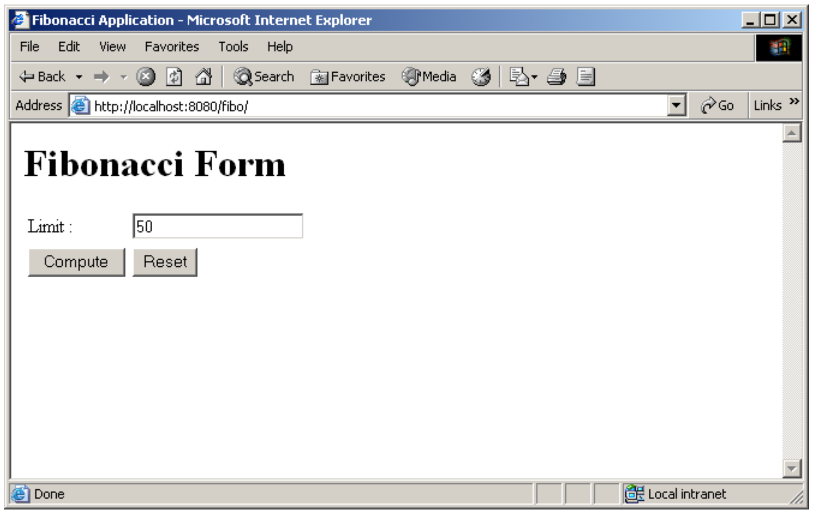
Open the **ComputeServlet.java** file. Double click in left column to create a breakpoint.

In the example, the breakpoint is set in front of the EJB creation.



```
        out.println("<html><head><title>");
        out.println(value);
        out.println("</title></head>");
        out.println("<body>");

        out.println("<h1>");
        out.println(value);
        out.println("</h1>");

        try {
            Fibo bean = home.create();
            int limit = 0;
            String value = request.getParameter("limit");
            if (value != null) {
                try {
                    limit = Integer.parseInt(value);
                } catch (Exception e) {
                }
            }
            double[] result = bean.compute(limit);
            bean.remove();

            out.println("<p>");
            out.print("The ");
            out.print(limit);
            out.print(" first Fibonacci numbers ");
```

Open a web browser and type **http://localhost:8080/fibo/**. The host/port can change if the web server listens on another host/port. You should a simple form like the one above. Enter a positive value in the field and press "Compute".

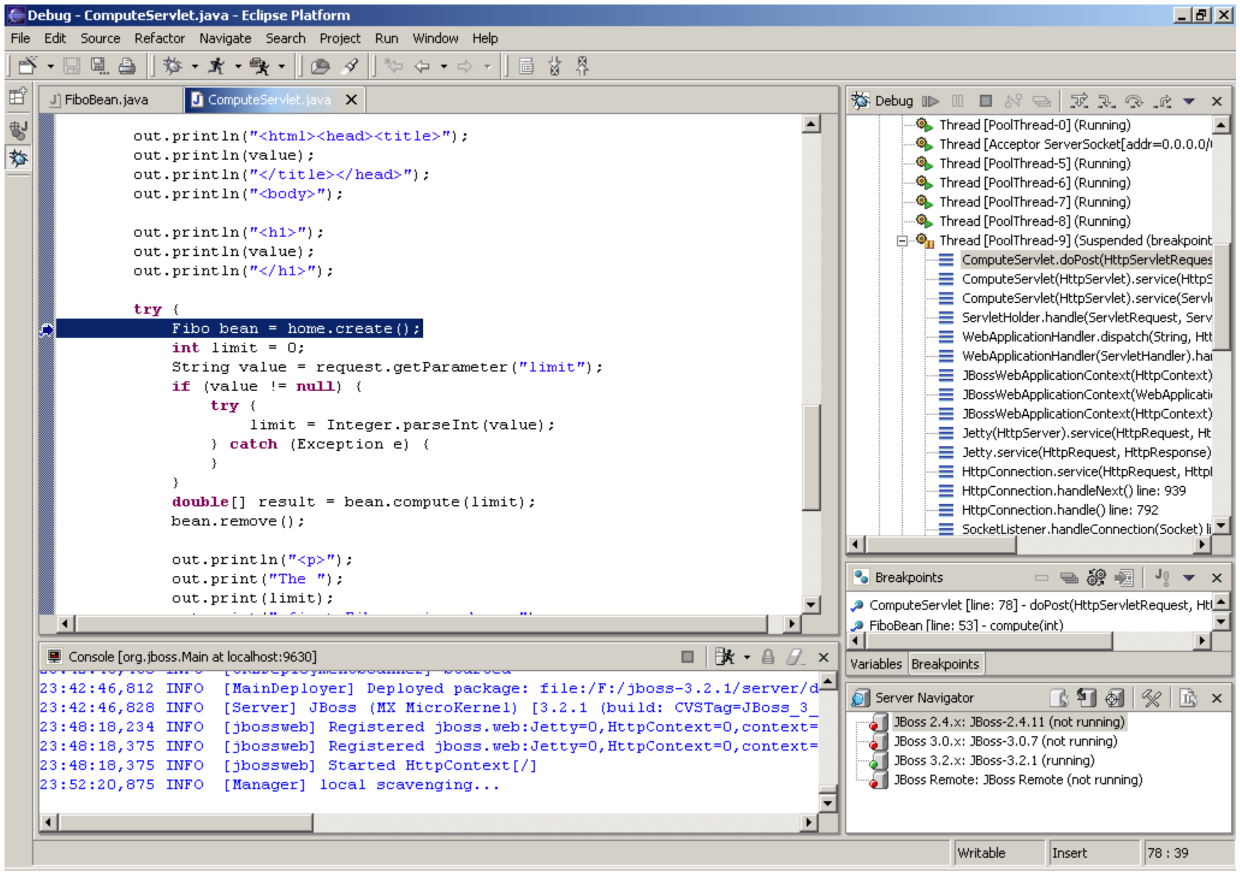

Fibonacci Application - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/fibo/
Go Links
Done
Local intranet

Fibonacci Form

Limit : 50

Compute Reset

Switch to your Eclipse workbench. You should see that execution has been suspended on the first breakpoint (in the servlet). You can go step by step in the code or go on with execution.


Debug - ComputeServlet.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
FiboBean.java
ComputeServlet.java
Debug
Breakpoints
Variables
Breakpoints
Server Navigator
Writable
Insert
78 : 39

```
        out.println("<html><head><title>");
        out.println(value);
        out.println("</title></head>");
        out.println("<body>");

        out.println("<h1>");
        out.println(value);
        out.println("</h1>");

        try {
            Fibo bean = home.create();
            int limit = 0;
            String value = request.getParameter("limit");
            if (value != null) {
                try {
                    limit = Integer.parseInt(value);
                } catch (Exception e) {
                }
            }
            double[] result = bean.compute(limit);
            bean.remove();

            out.println("<p>");
            out.print("The ");
            out.print(limit);
```

```
Thread [PoolThread-0] (Running)
Thread [Acceptor ServerSocket[addr=0.0.0.0/
Thread [PoolThread-5] (Running)
Thread [PoolThread-6] (Running)
Thread [PoolThread-7] (Running)
Thread [PoolThread-8] (Running)
Thread [PoolThread-9] (Suspended (breakpoint
    ComputeServlet.doPost(HttpServletReques
    ComputeServlet(HttpServlet).service(HttpS
    ComputeServlet(HttpServlet).service(Servl
    ServletHolder.handle(ServletRequest, Serv
    WebApplicationHandler.dispatch(String, Htt
    WebApplicationHandler(ServletHandler).ha
    JBossWebApplicationContext(HttpContext)
    JBossWebApplicationContext(WebApplicati
    JBossWebApplicationContext(HttpContext)
    Jetty(HttpServer).service(HttpRequest, Ht
    Jetty.service(HttpRequest, HttpResponse)
    HttpConnection.service(HttpRequest, Httpl
    HttpConnection.handleNext() line: 939
    HttpConnection.handle() line: 792
    SocketListener.handleConnection(Socket) li
```

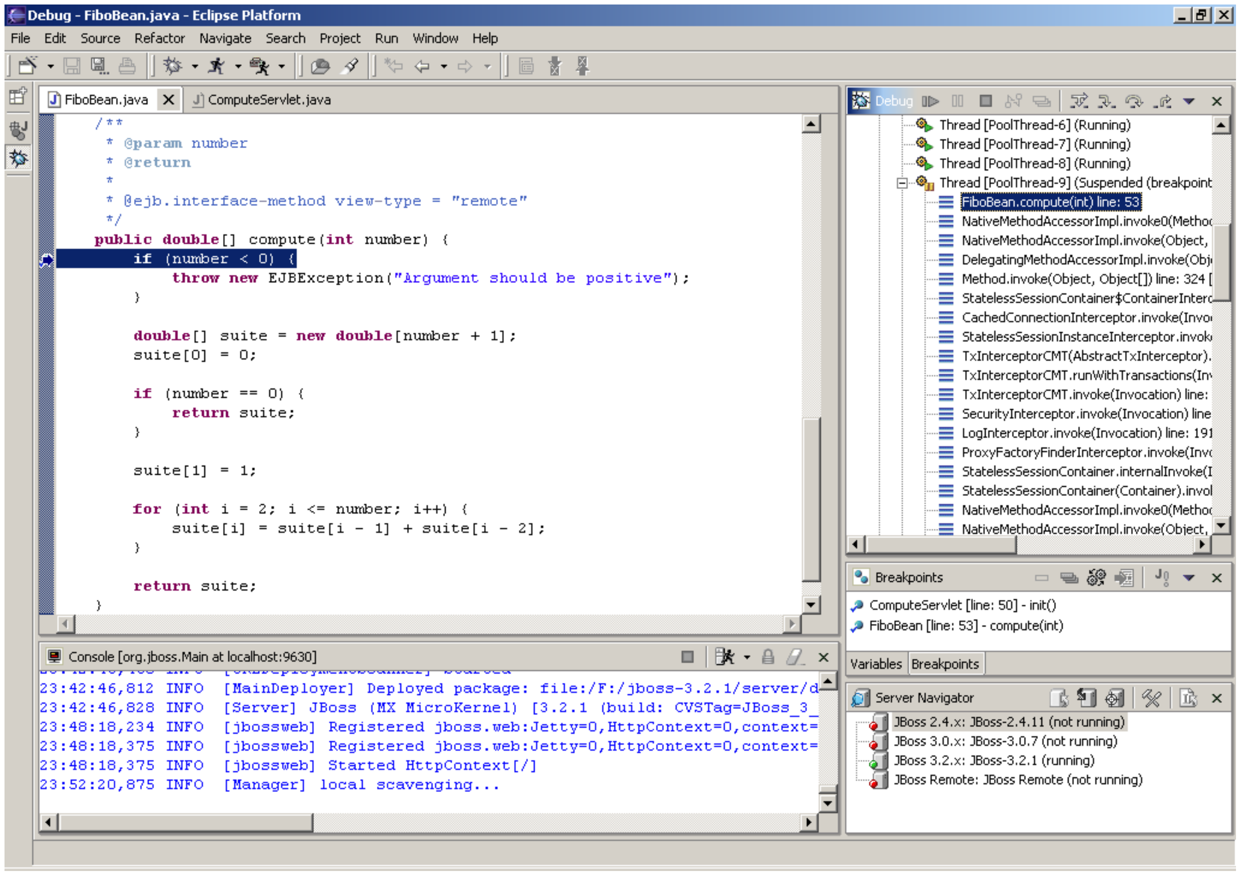
```
ComputeServlet [line: 78] - doPost(HttpServletRequest, Ht
FiboBean [line: 53] - compute(int)
```

```
JBoss 2.4.x: JBoss-2.4.11 (not running)
JBoss 3.0.x: JBoss-3.0.7 (not running)
JBoss 3.2.x: JBoss-3.2.1 (running)
JBoss Remote: JBoss Remote (not running)
```

Console [org.jboss.Main at localhost:9630]

```
23:42:46,812 INFO  [MainDeployer] Deployed package: file:/F:/jboss-3.2.1/server/d
23:42:46,828 INFO  [Server] JBoss (MX MicroKernel) [3.2.1 (build: CVSTag=JBoss_3_
23:48:18,234 INFO  [jbossweb] Registered jboss.web:Jetty=0,HttpContext=0,context=
23:48:18,375 INFO  [jbossweb] Registered jboss.web:Jetty=0,HttpContext=0,context=
23:48:18,375 INFO  [jbossweb] Started HttpContext[/]
23:52:20,875 INFO  [Manager] local scavenging...
```

Another suspension must occured when hitting the second breakpoint (in the EJB). You can go step by step in the code or go on with execution.


Debug - FiboBean.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
FiboBean.java
ComputeServlet.java
Debug
Breakpoints
Variables
Breakpoints
Server Navigator

```java
    /**
     * @param number
     * @return
     *
     * @ejb.interface-method view-type = "remote"
     */
    public double[] compute(int number) {
        if (number < 0) {
            throw new EJBException("Argument should be positive");
        }

        double[] suite = new double[number + 1];
        suite[0] = 0;

        if (number == 0) {
            return suite;
        }

        suite[1] = 1;

        for (int i = 2; i <= number; i++) {
            suite[i] = suite[i - 1] + suite[i - 2];
        }

        return suite;
    }
```

Thread [PoolThread-6] (Running)
Thread [PoolThread-7] (Running)
Thread [PoolThread-8] (Running)
Thread [PoolThread-9] (Suspended (breakpoint
FiboBean.compute(int) line: 53
NativeMethodAccessorImpl.invoke0(Method
NativeMethodAccessorImpl.invoke(Object,
DelegatingMethodAccessorImpl.invoke(Obj
Method.invoke(Object, Object[]) line: 324 [
StatelessSessionContainer$ContainerInter
CachedConnectionInterceptor.invoke(Invo
StatelessSessionInstanceInterceptor.invok
TxInterceptorCMT(AbstractTxInterceptor).
TxInterceptorCMT.runWithTransactions(In
TxInterceptorCMT.invoke(Invocation) line:
SecurityInterceptor.invoke(Invocation) line
LogInterceptor.invoke(Invocation) line: 191
ProxyFactoryFinderInterceptor.invoke(Inv
StatelessSessionContainer.internalInvoke(I
StatelessSessionContainer(Container).invol
NativeMethodAccessorImpl.invoke0(Method
NativeMethodAccessorImpl.invoke(Object,

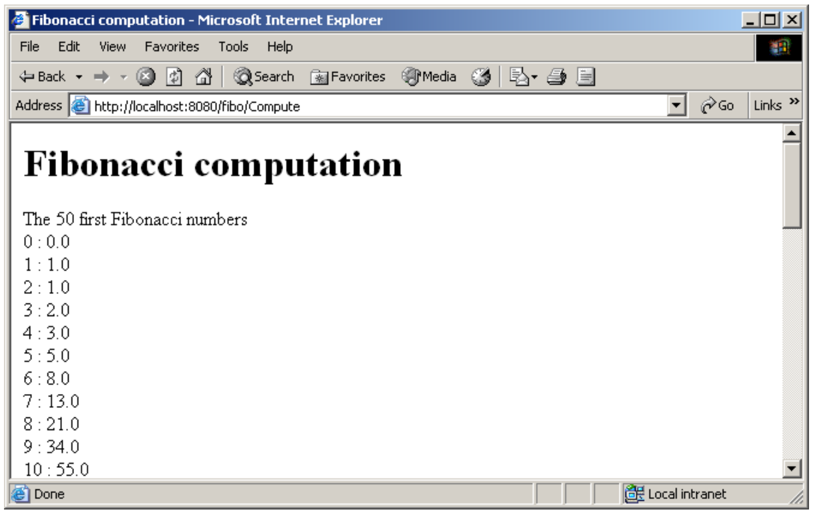
ComputeServlet [line: 50] - init()
FiboBean [line: 53] - compute(int)

Console [org.jboss.Main at localhost:9630]

```
23:42:46,812 INFO  [MainDeployer] Deployed package: file:/F:/jboss-3.2.1/server/d
23:42:46,828 INFO  [Server] JBoss (MX MicroKernel) [3.2.1 (build: CVSTag=JBoss_3_
23:48:18,234 INFO  [jbossweb] Registered jboss.web:Jetty=0,HttpContext=0,context=
23:48:18,375 INFO  [jbossweb] Registered jboss.web:Jetty=0,HttpContext=0,context=
23:48:18,375 INFO  [jbossweb] Started HttpContext[/]
23:52:20,875 INFO  [Manager] local scavenging...
```

JBoss 2.4.x: JBoss-2.4.11 (not running)
JBoss 3.0.x: JBoss-3.0.7 (not running)
JBoss 3.2.x: JBoss-3.2.1 (running)
JBoss Remote: JBoss Remote (not running)

After resuming the execution and back to the browser, the response should be like above.


Fibonacci computation - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
http://localhost:8080/fibo/Compute
Go
Links
Done
Local intranet

Fibonacci computation

The 50 first Fibonacci numbers
0 : 0.0
1 : 1.0
2 : 1.0
3 : 2.0
4 : 3.0
5 : 5.0
6 : 8.0
7 : 13.0
8 : 21.0
9 : 34.0
10 : 55.0

Conclusion

This simple tutorial is intended to give an overview of what is possible with JBoss IDE. We hope that it will be useful for developers who want to go with it.