

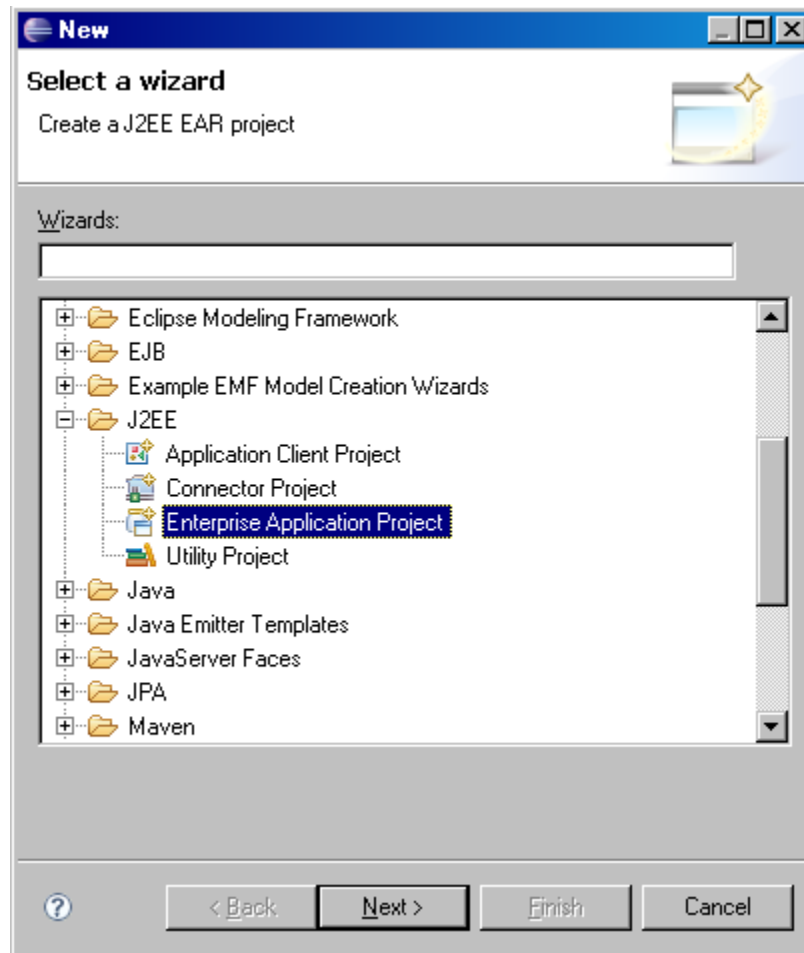
5-minute Tutorial on Enterprise Application Development with Eclipse and Geronimo

So you've got 5 minutes and want to see how much you can do with Eclipse and Geronimo 2, don't you? Install **Eclipse** and **Geronimo Eclipse plugin** as described at [Geronimo Eclipse Plugin Installation Instructions](#) and 5 minutes seem quite enough. Let's check it out!

Read some introductory material on how to define Geronimo in Eclipse at [Geronimo Eclipse Plugin Usage Instructions](#)

Create Enterprise Application Project

Start from creating an enterprise application project. Select **File > New**, select **Project...** (or alternatively press **Ctrl-N**) and in the popup window select **Enterprise Application Project** in **J2EE** category.



Press **Next**.

In the EAR Application Project wizard type in **SampleEAR** as the project name and select **Apache Geronimo v2.0** in **Target Runtime**. Leave the rest as is.

New EAR Application Project

EAR Application Project
Create a EAR application.

Project name:

Project contents:
☒ Use default
Directory:

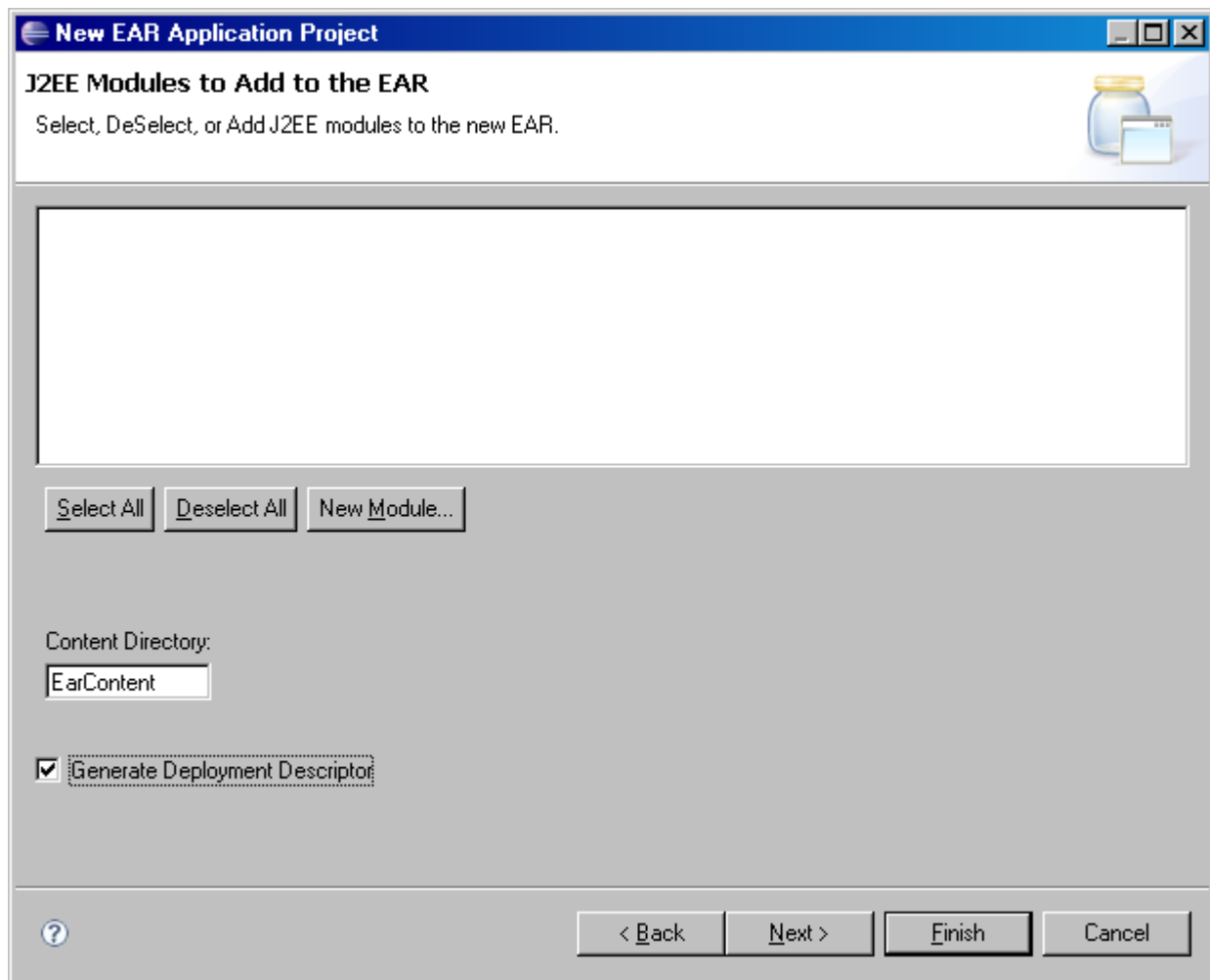
Target Runtime

Configurations

A good starting for working with Apache Geronimo v2.0 runtime. Additional facets can later be installed to add new functionality to the project.

Press **Next** twice.

In the New EAR Application Project window select the **Generate Deployment Descriptor** checkbox.



The dialog box is titled "New EAR Application Project" and contains a section "J2EE Modules to Add to the EAR" with instructions to "Select, DeSelect, or Add J2EE modules to the new EAR." Below this is a large empty list box. At the bottom of the list box are three buttons: "Select All", "Deselect All", and "New Module...". Below these buttons is a "Content Directory:" label followed by a text box containing "EarContent". Below the text box is a checked checkbox labeled "Generate Deployment Descriptor". At the bottom of the dialog are four buttons: a help button (question mark icon), "< Back", "Next >", and "Finish". A "Cancel" button is also present to the right of the "Finish" button.

New EAR Application Project

J2EE Modules to Add to the EAR
Select, DeSelect, or Add J2EE modules to the new EAR.

Select All Deselect All New Module...

Content Directory:
EarContent

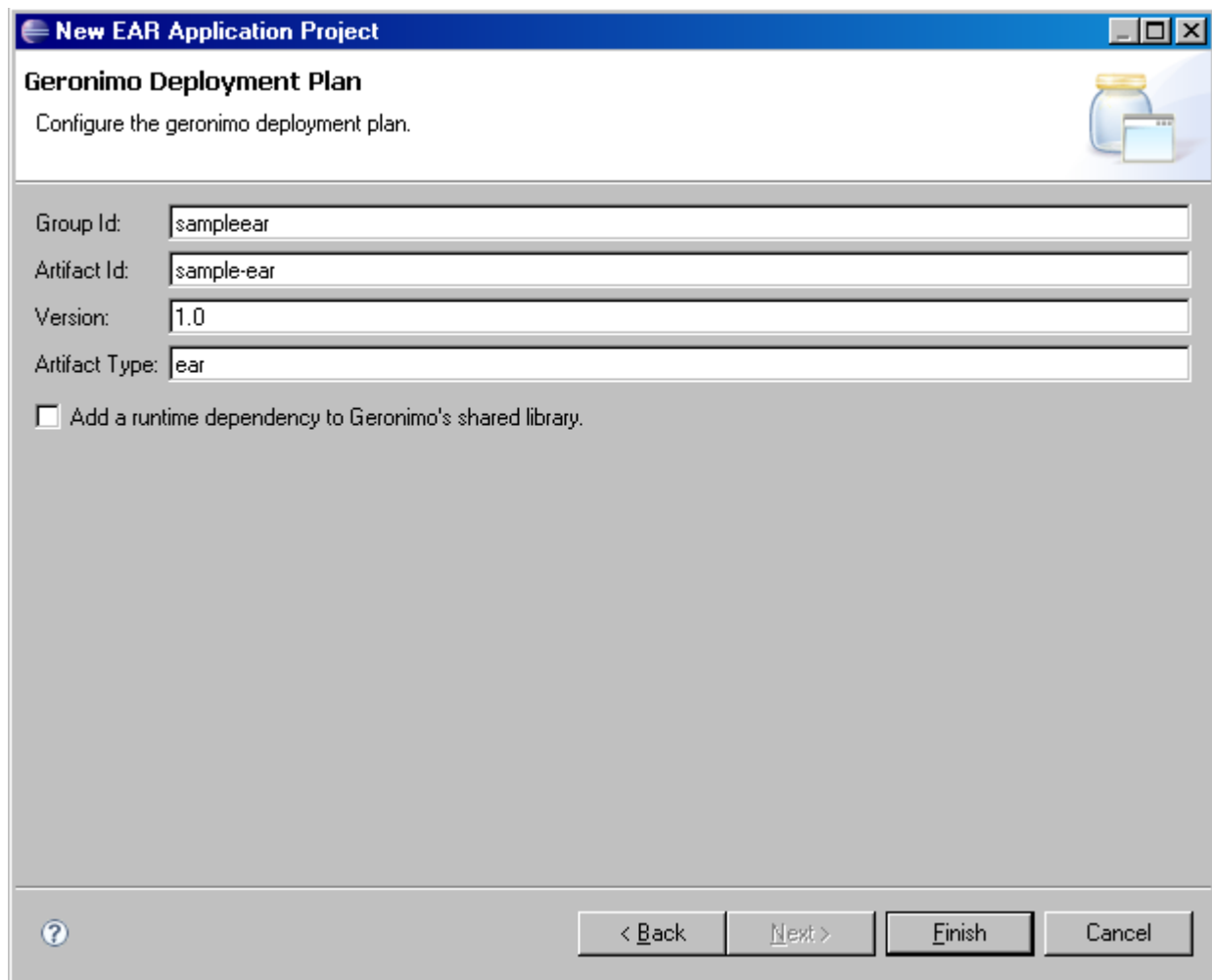
☒ Generate Deployment Descriptor

? < Back Next > Finish Cancel

Press **Next**.

Fill in the Geronimo Deployment Plan's fields with the following values (you want to know more why they're important? Read it on in the Geronimo 1.2 Documentation - [Deployment plans](#)):

- Group Id: **sampleear**
- Artifact Id: **sample-ear**
- Artifact Type: **ear**



The dialog box is titled "New EAR Application Project" with a blue header bar. Below the header, the title "Geronimo Deployment Plan" is displayed, followed by the instruction "Configure the geronimo deployment plan." and a small icon of a jar and a document. The main area contains four text input fields: "Group Id:" with the value "sampleear", "Artifact Id:" with the value "sample-ear", "Version:" with the value "1.0", and "Artifact Type:" with the value "ear". Below these fields is a checkbox labeled "Add a runtime dependency to Geronimo's shared library." which is currently unchecked. At the bottom, there is a question mark icon on the left and four buttons: "< Back", "Next >", "Finish", and "Cancel".

New EAR Application Project

Geronimo Deployment Plan
Configure the geronimo deployment plan.

Group Id: sampleear

Artifact Id: sample-ear

Version: 1.0

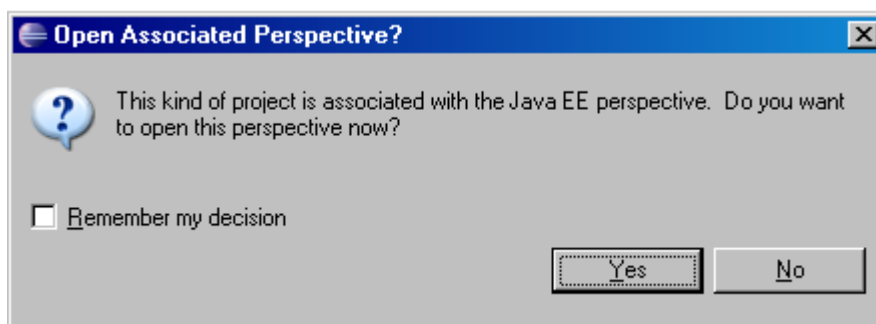
Artifact Type: ear

☐ Add a runtime dependency to Geronimo's shared library.

? < Back Next > Finish Cancel

Press **Finish**.

When asked about changing to the Java EE perspective, press Yes. You may want to select the **Remember my decision** checkbox to avoid dealing with it in the future.



The dialog box is titled "Open Associated Perspective?" with a blue header bar. It contains a question mark icon and the text "This kind of project is associated with the Java EE perspective. Do you want to open this perspective now?". Below this is a checkbox labeled "Remember my decision" which is unchecked. At the bottom right, there are two buttons: "Yes" and "No".

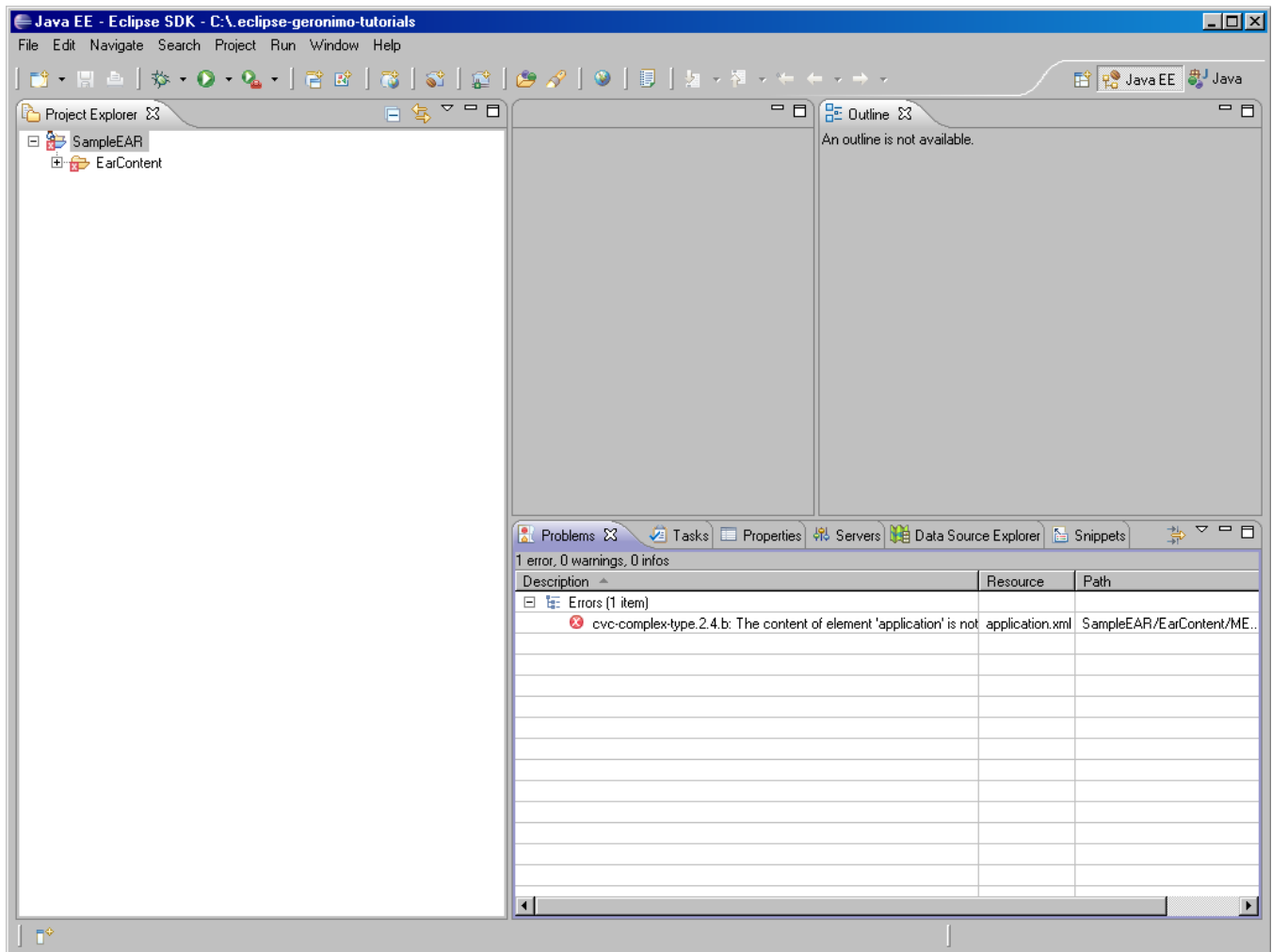
Open Associated Perspective?

? This kind of project is associated with the Java EE perspective. Do you want to open this perspective now?

☐ Remember my decision

Yes No

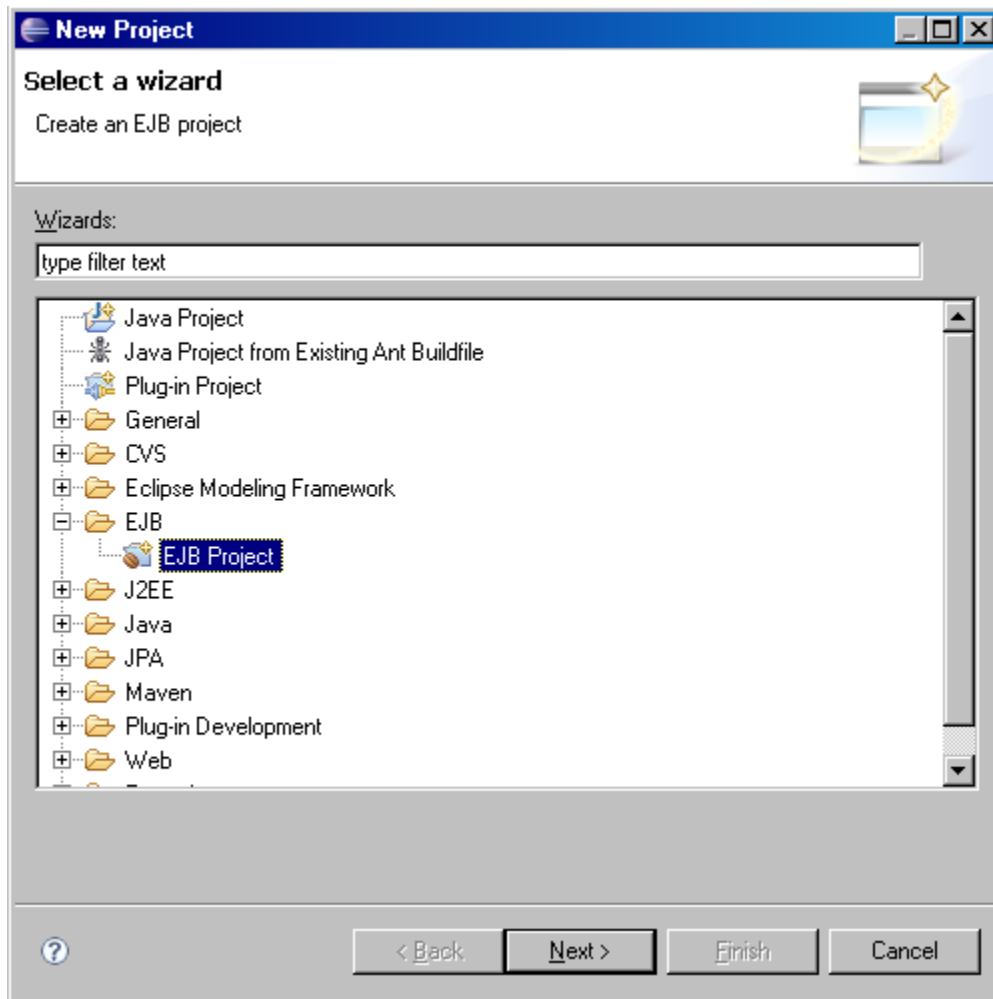
You should now have the following project structure.



Don't worry about the error **cvc-complex-type.2.4.b: The content of element 'application' is not complete...** for now. You'll fix it in the next step when you define an ejb module (and webapp module afterwards).

Create EJB project

The next step is to create an EJB project to hold your EJBs. Press **Ctrl-N** and select **EJB Project** in **EJB** category.



Press **Next**.

In the EJB Project wizard type in **SampleEJB** as the project name and select **Add project to an EAR** checkbox. Leave the rest as is.

New EJB Project

EJB Project
Create an EJB Project and add it to a new or existing Enterprise Application.

Project name:

Project contents:
☒ Use default
Directory:

Target Runtime

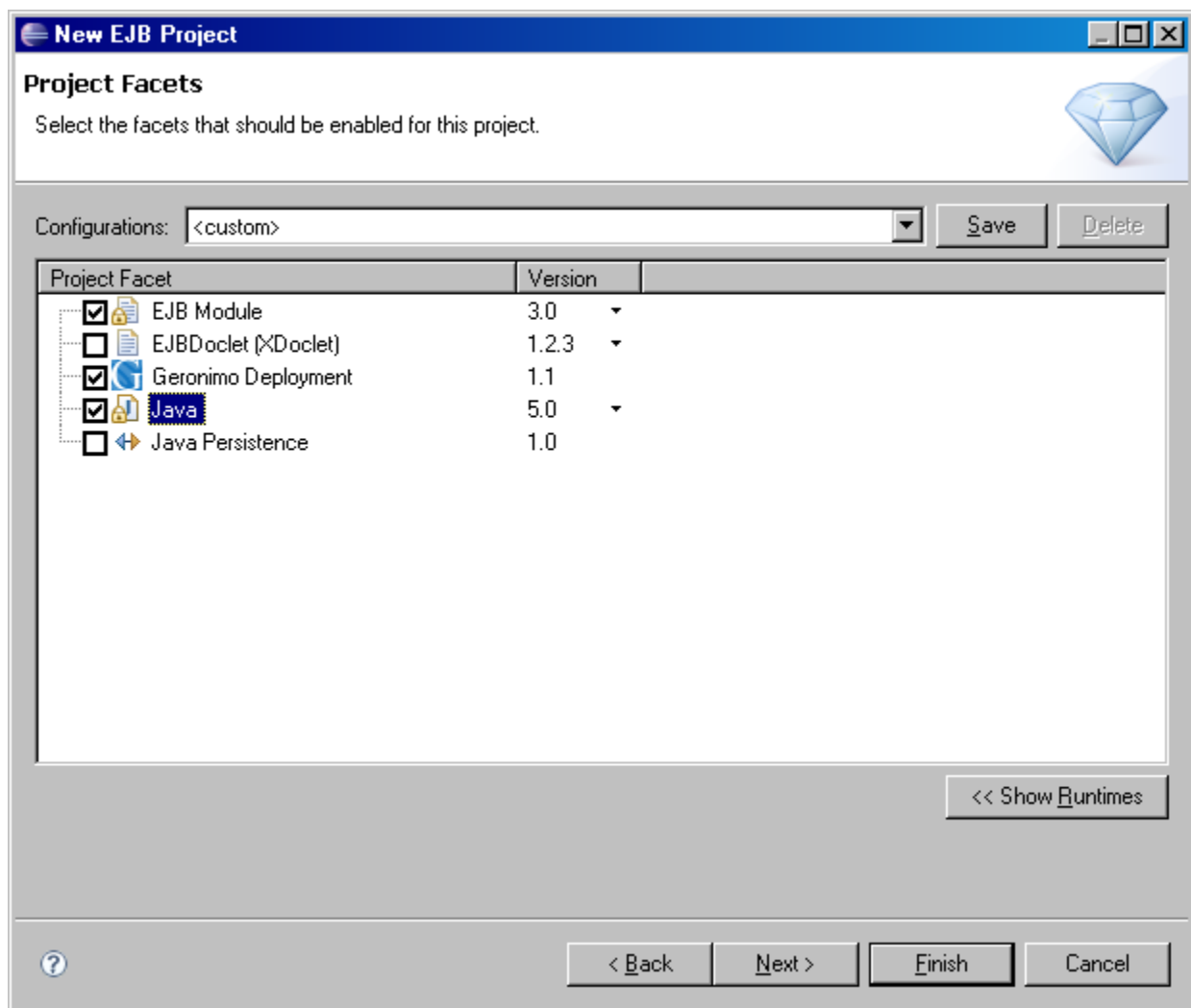
Configurations

A good starting for working with Apache Geronimo v2.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR Membership
☒ Add project to an EAR
EAR Project Name:

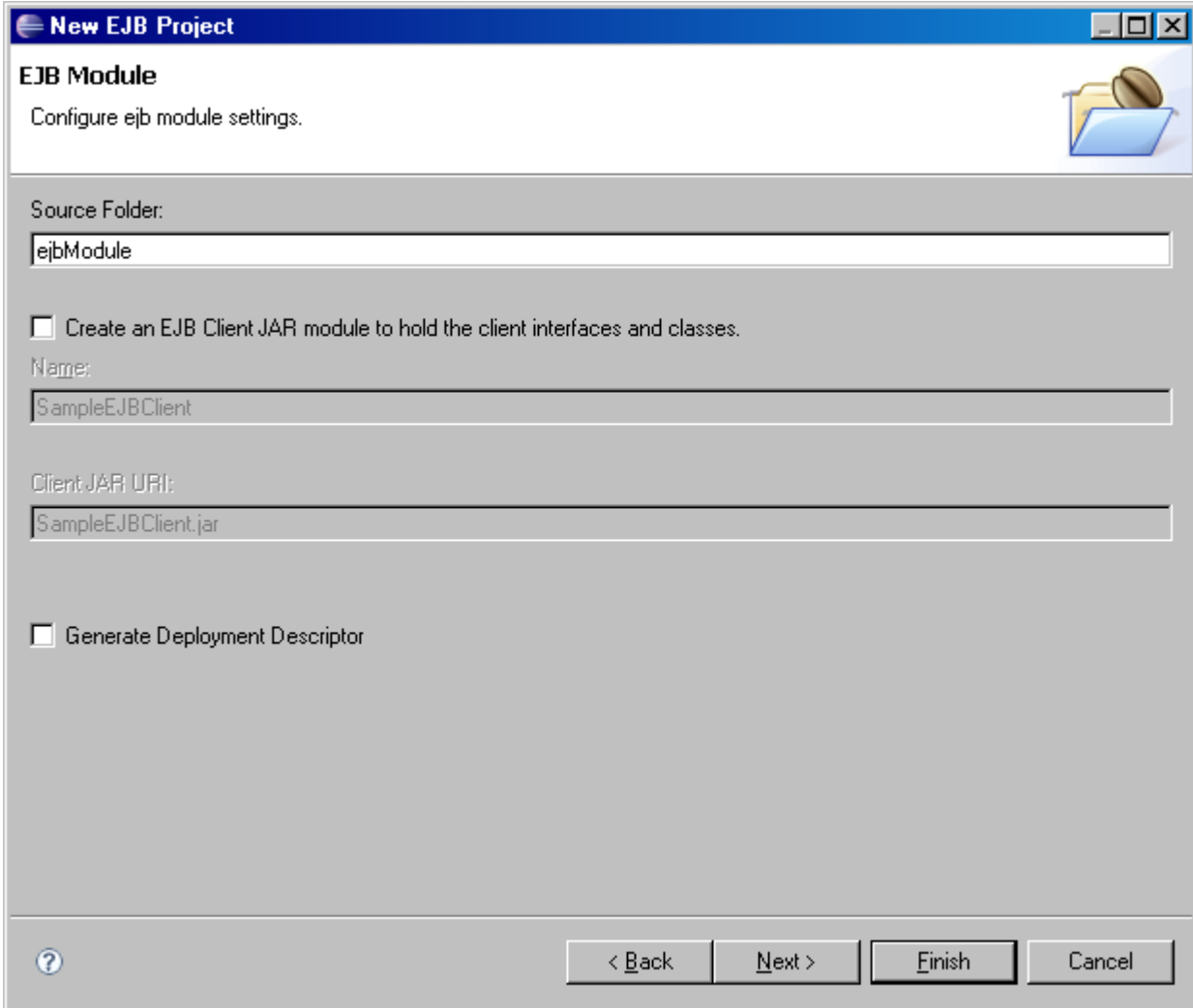
Press **Next**.

Make sure that **5.0** for the **Java** facet in the **Project Facets** popup window's selected.



Press **Next**.

Unselect the **Create an EJB Client JAR module to hold the client interfaces and classes** checkbox. We're not interested in it.

The image shows a 'New EJB Project' dialog box. The title bar is blue with the text 'New EJB Project' and standard window controls. Below the title bar, the text 'EJB Module' is followed by 'Configure ejb module settings.' and a folder icon. The main area contains several fields and checkboxes. The 'Source Folder:' field has 'ejbModule' entered. A checkbox 'Create an EJB Client JAR module to hold the client interfaces and classes.' is unchecked. The 'Name:' field has 'SampleEJBClient' entered. The 'Client JAR URI:' field has 'SampleEJBClient.jar' entered. Another checkbox 'Generate Deployment Descriptor' is unchecked. At the bottom, there is a help icon, and buttons for '< Back', 'Next >', 'Finish', and 'Cancel'.

New EJB Project

EJB Module
Configure ejb module settings.

Source Folder:
ejbModule

☐ Create an EJB Client JAR module to hold the client interfaces and classes.

Name:
SampleEJBClient

Client JAR URI:
SampleEJBClient.jar

☐ Generate Deployment Descriptor

? < Back Next > Finish Cancel

Press **Next**.

Fill in the Geronimo Deployment Plan fields with the following values:


- Group Id: **sampleear**
- Artifact Id: **sample-ejb**
- Artifact Type: **ejb**

New EJB Project

Geronimo Deployment Plan
Configure the geronimo deployment plan.

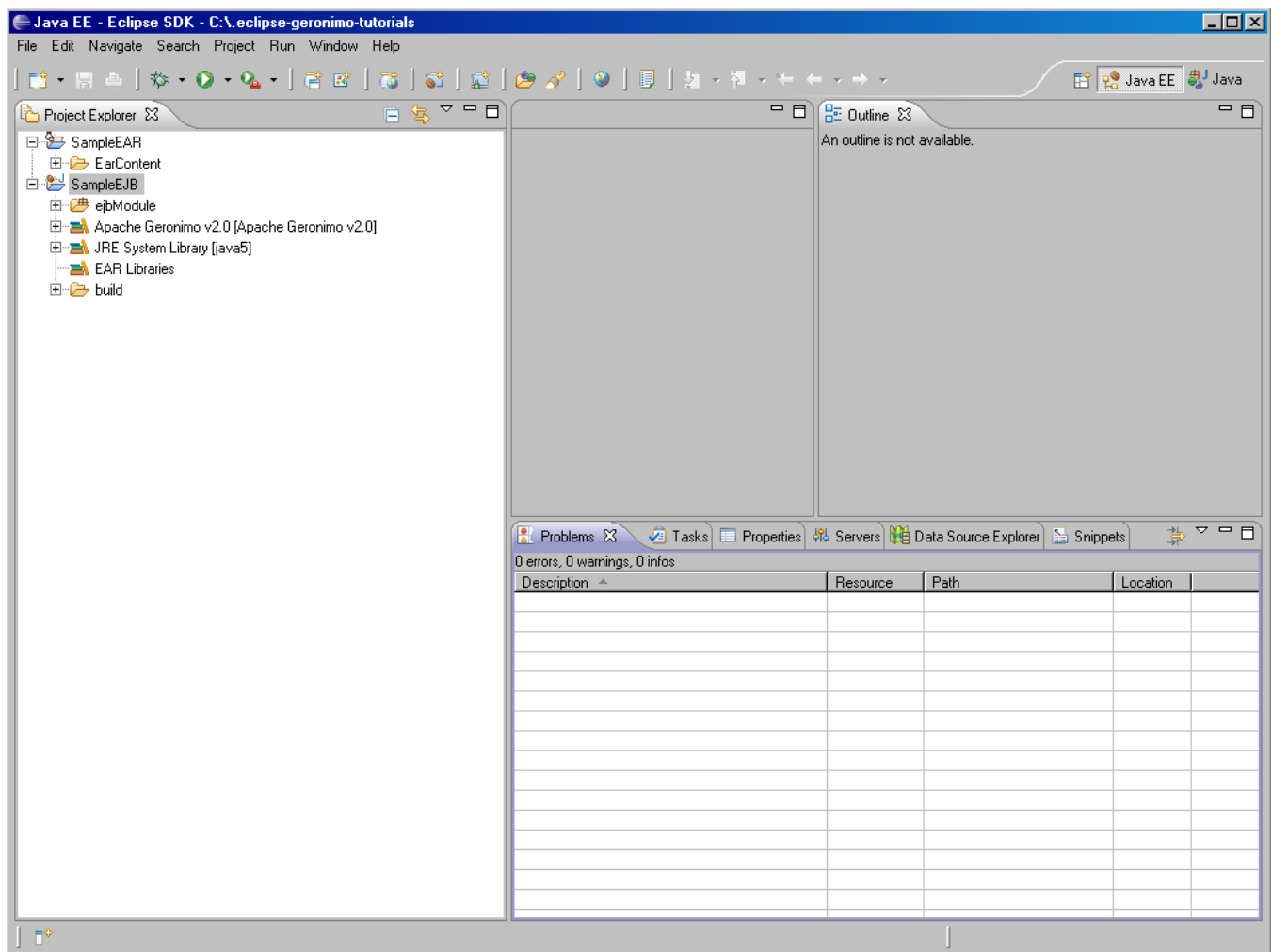
Group Id:
Artifact Id:
Version:
Artifact Type:

☐ Add a runtime dependency to Geronimo's shared library.



Press **Finish**.

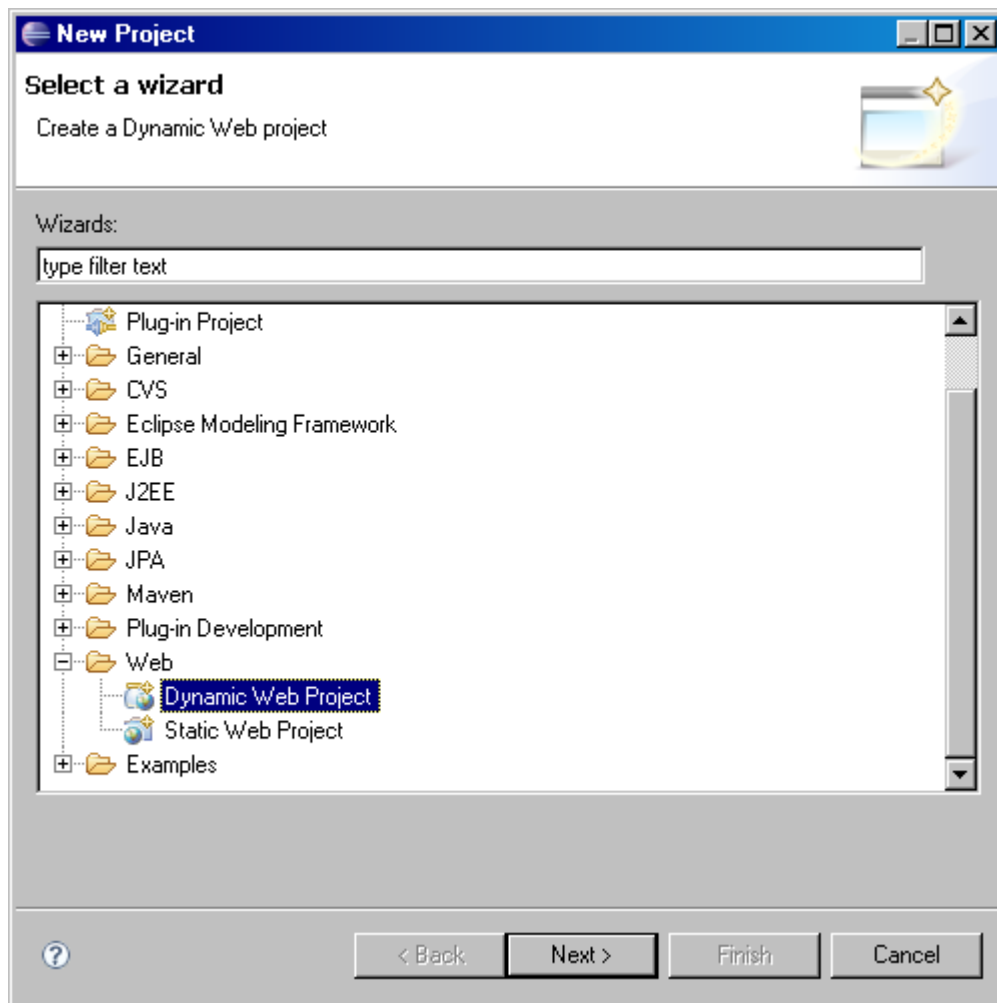
You should now have the following project structure.



Remove `ejbModule/META-INF/openejb-jar.xml` file in the SampleEJB project as it causes deployment issues.

Create Dynamic Web Project

The next step is to create a Dynamic Web project to hold your web application. Press **Ctrl-N** and select **Dynamic Web Project** in **Web** category.



Press **Next**.

In the Dynamic Web Project wizard type in **SampleWAR** as the project name and select **Add project to an EAR** checkbox. Leave the rest as is.

New Dynamic Web Project

Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project contents:
☒ Use default
Directory:

Target Runtime

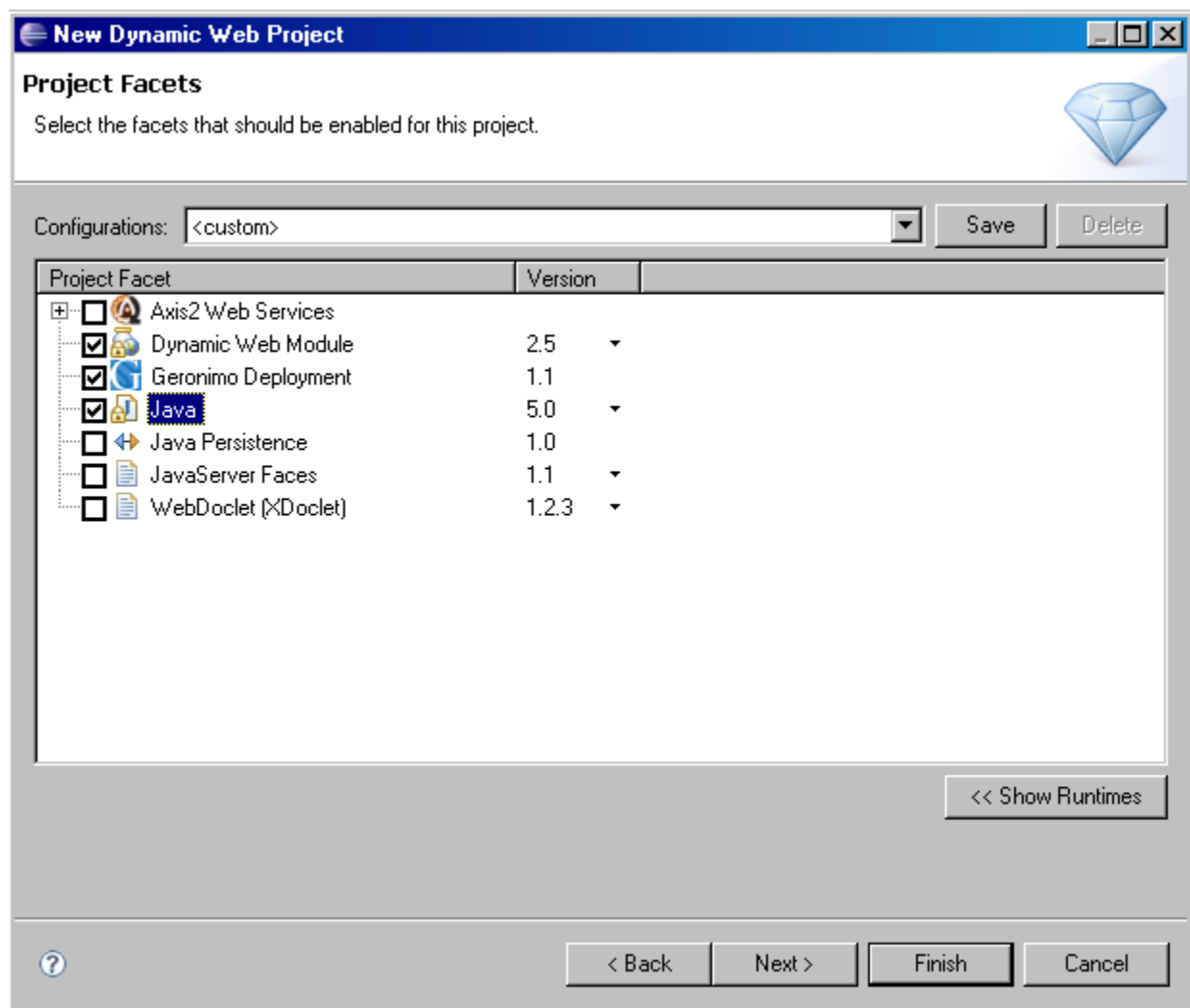
Configurations

A good starting for working with Apache Geronimo v2.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR Membership
☒ Add project to an EAR
EAR Project Name:

Press **Next**.

Make sure that **5.0** for the **Java** facet in the **Project Facets** popup window's selected.



Press **Next** twice.

Fill in the Geronimo Deployment Plan fields with the following values:


- Group Id: **sampleear**
- Artifact Id: **sample-war**
- Artifact Type: **war**

New Dynamic Web Project

Geronimo Deployment Plan
Configure the geronimo deployment plan.

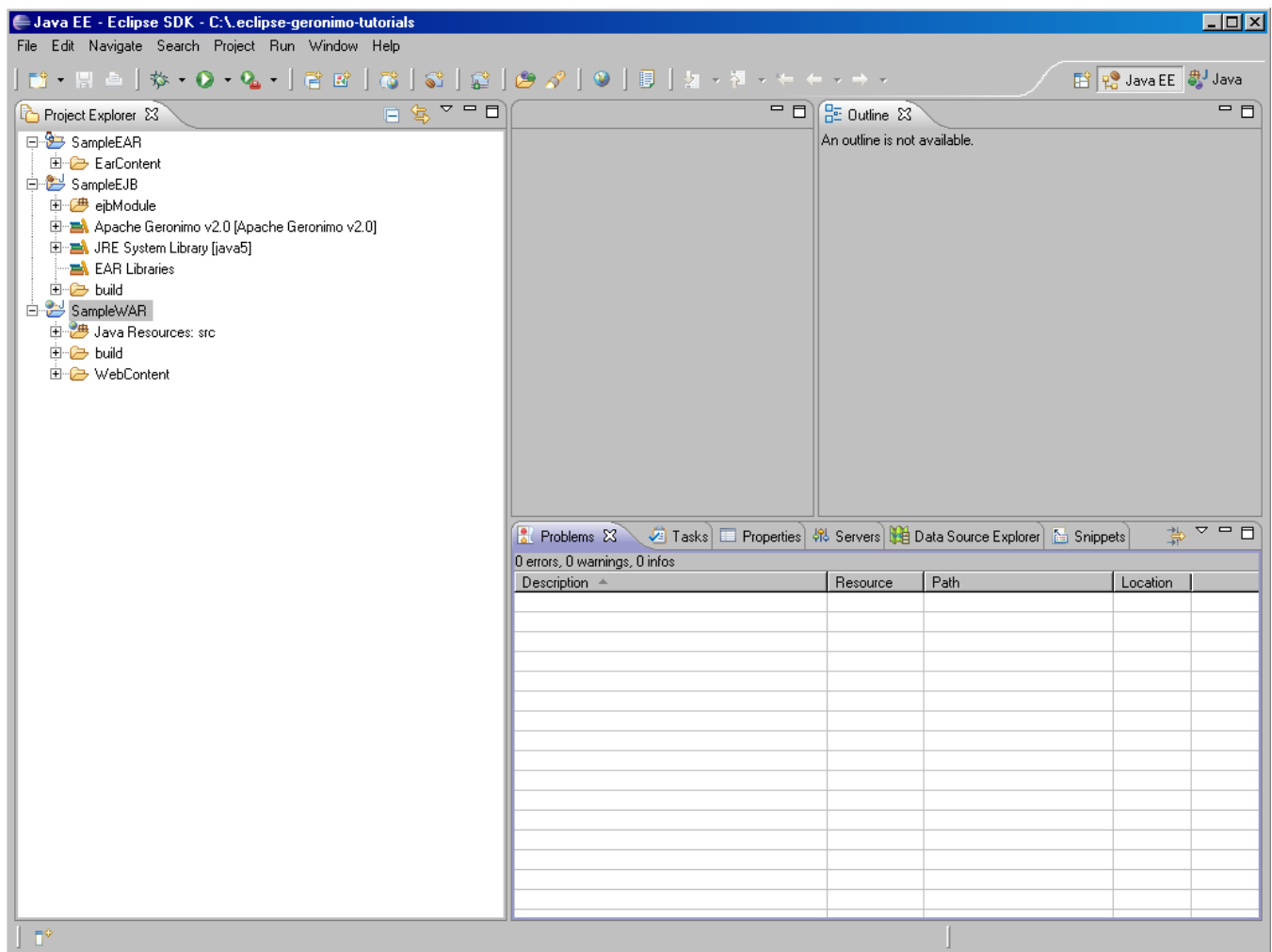
Group Id:
Artifact Id:
Version:
Artifact Type:

☐ Add a runtime dependency to Geronimo's shared library.



Press **Finish**.

You should now have the following project structure.



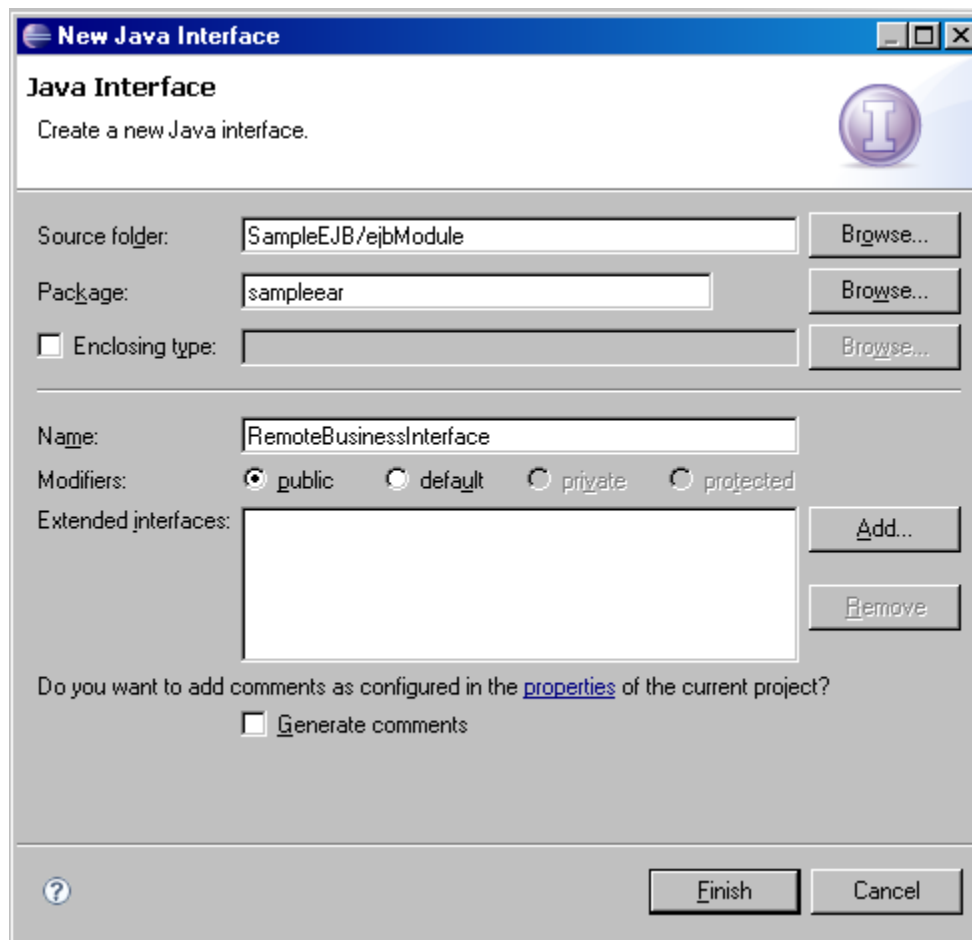
Create Stateless Session EJB

Every stateless session ejb has its own business interface. There're three types of business interfaces - **@Remote**, **@Local** and **@WebService** - and their combinations. Suffice to say that every EJB development starts from defining a business interface and implementing it by a bean implementation class.

Create remote business interface

Right-click on the **SampleEJB** project and select **New > Interface** and fill it in with the following values:

- Package: **sampleear**
- Name: **RemoteBusinessInterface**



Press **Finish**.

Add a business method and mark the interface as a remote one with @Remote annotation.

RemoteBusinessInterface.java

```
package sampleear;

import javax.ejb.Remote;

@Remote
public interface RemoteBusinessInterface {
    public String sayHello(String name);
}
```

Create bean class

Right-click on the **SampleEJB** project and select **New > Class** and fill it in with the following values:

- Package: **sampleear**
- Name: **MyStatelessSessionBean**
- Interfaces: **sampleear.RemoteBusinessInterface**

New Java Class

Java Class
Create a new Java class.

Source folder: SampleEJB/ejbModule Browse...

Package: sampleear Browse...

☐ Enclosing type: Browse...

Name: MyStatelessSessionBean

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: sampleear.RemoteBusinessInterface Add...
Remove

Which method stubs would you like to create?

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

Do you want to add comments as configured in the [properties](#) of the current project?
☐ Generate comments

? Finish Cancel

Press **Finish**.

Implement the business method **sayHello** and mark the class as a stateless session bean with the `@Stateless` annotation.

MyStatelessSessionBean.java

```
package sampleear;

import javax.ejb.Stateless;

@Stateless
public class MyStatelessSessionBean implements RemoteBusinessInterface {

    public String sayHello(String name) {
        return getClass().getName() + " says hello to " + name + ".";
    }
}
```

Web application development

The time has come to use the ejb in the web application. We create a jsp page **index.jsp** that executes a servlet **MyServlet** that in turn executes the ejb **MyStatelessSessionBean**.

Create welcome page - index.jsp

Right-click on the **SampleWAR** project and select **New > JSP**. Name it **index.jsp**. Press **Finish**.

Change it so it executes the servlet upon form submission.

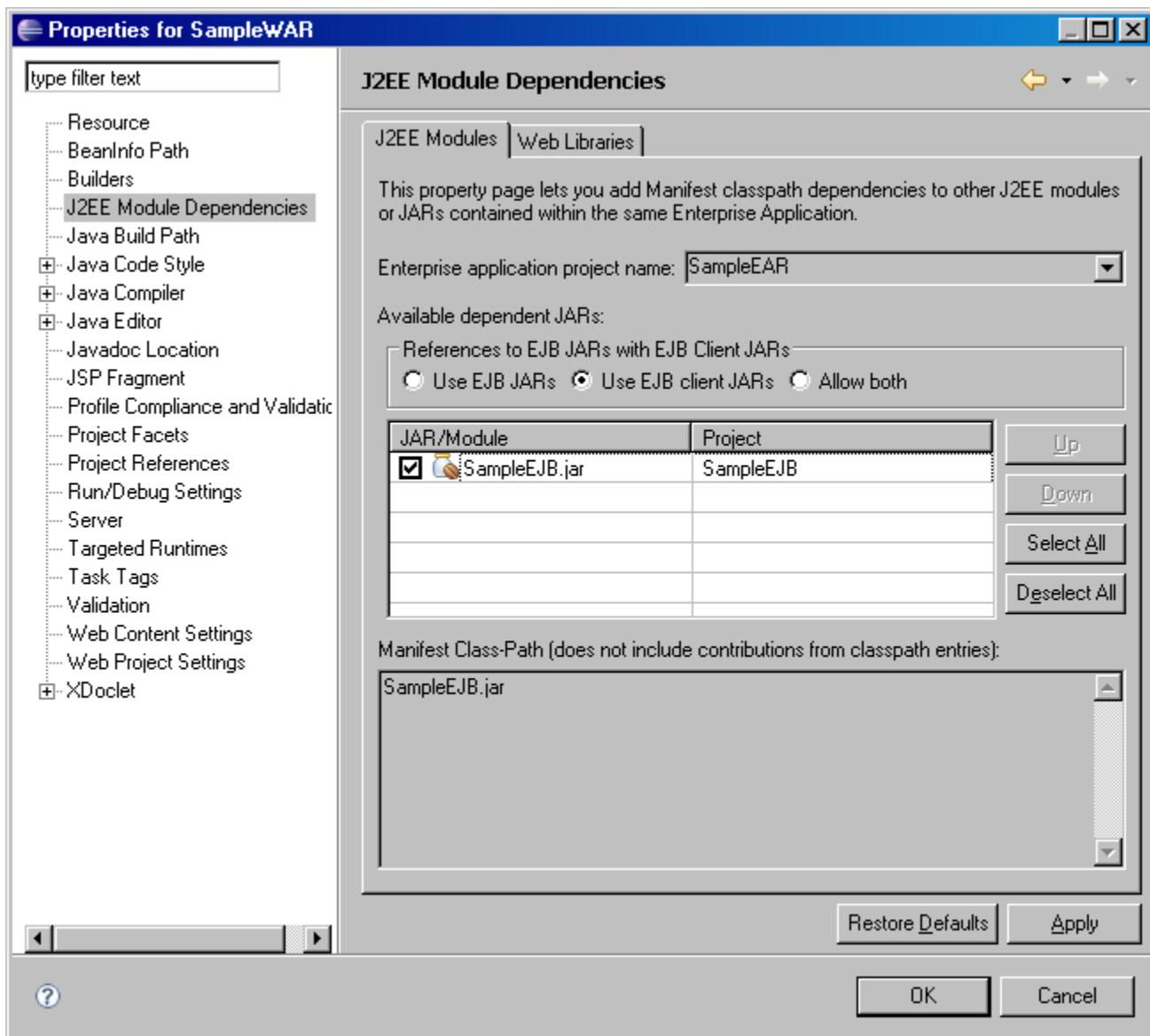
index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>5-minute Tutorial on Enterprise Application Development with Eclipse and Geronimo</title>
  </head>
  <body>
    <form action="${pageContext.request.contextPath}/sayHello">
      <input type="text" name="name" /><input type="submit" value="Press me!" />
    </form>
  </body>
</html>
```

Create servlet - MyServlet

Since the servlet calls the EJB, the web project the servlet is in depends on the EJB project. Let's define the dependency.

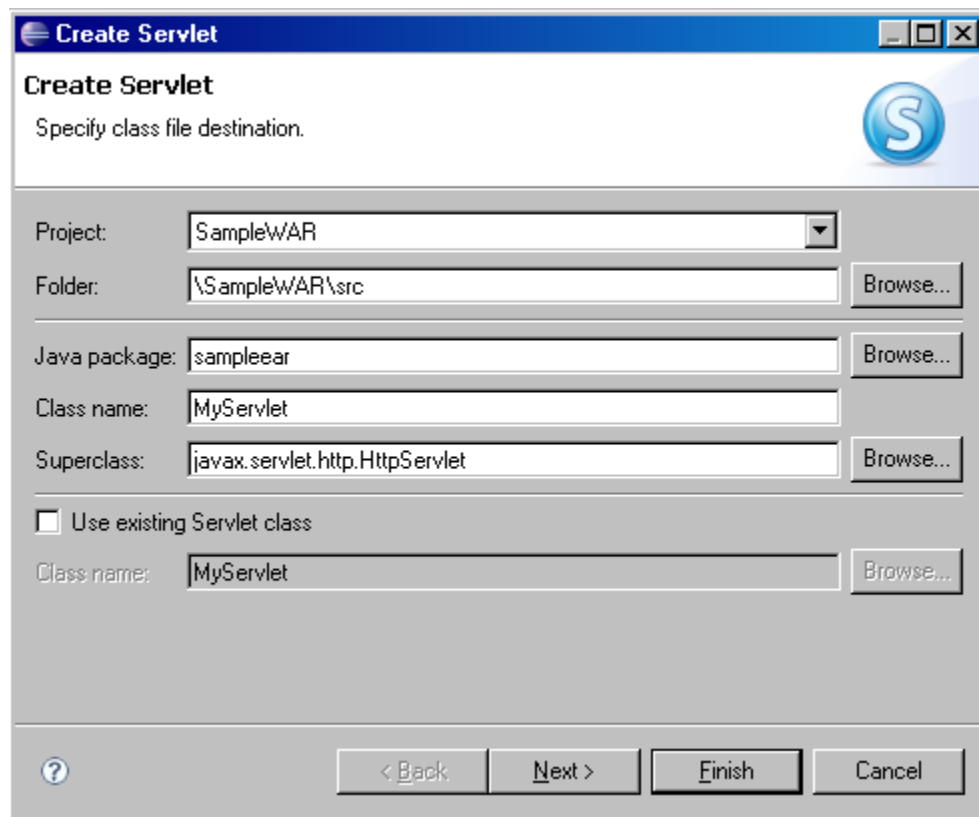
Right-click on the **SampleWAR** project and select **Properties**. Go to **J2EE Module Dependencies** and select the checkbox next to **SampleEJB.jar** (it's in the **J2EE Modules** tab).



Press **OK**.

Right-click on the **SampleWAR** project and select **New > Servlet** and fill it in with the following values:

- Java Package: **sampleear**
- Class name: **MyServlet**



The image shows a 'Create Servlet' dialog box with a blue title bar and a white header area. The header contains the text 'Create Servlet' and 'Specify class file destination.' along with a blue 'S' icon. The main area has several input fields: 'Project' (a dropdown menu showing 'SampleWAR'), 'Folder' (a text field showing '\\SampleWAR\\src' with a 'Browse...' button), 'Java package' (a text field showing 'sampleear' with a 'Browse...' button), 'Class name' (a text field showing 'MyServlet'), and 'Superclass' (a text field showing 'javax.servlet.http.HttpServlet' with a 'Browse...' button). Below these is a checkbox labeled 'Use existing Servlet class' which is unchecked. Under this checkbox is another 'Class name' text field showing 'MyServlet' with a 'Browse...' button. At the bottom, there is a question mark icon, a '< Back' button, a 'Next >' button, a 'Finish' button, and a 'Cancel' button.

Create Servlet

Specify class file destination.

Project: SampleWAR

Folder: \\SampleWAR\\src

Java package: sampleear

Class name: MyServlet

Superclass: javax.servlet.http.HttpServlet

☐ Use existing Servlet class

Class name: MyServlet


Press **Next**.

Change the URL Mapping section so the servlet serves at **/sayHello** url mapping.

Create Servlet

Create Servlet

Enter servlet deployment descriptor specific information.



Name

MyServlet

Description

Initialization Parameters:

Add...

Edit...

Remove


URL Mappings:

/sayHello

Add...

Edit...

Remove



< Back

Next >

Finish

Cancel

Press **Finish**.

Change it to call off the ejb when executed.

MyServlet.java

```
package sampleear;

import java.io.IOException;

import javax.ejb.EJB;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyServlet extends javax.servlet.http.HttpServlet implements javax.servlet.Servlet {
    static final long serialVersionUID = 1L;

    @EJB
    RemoteBusinessInterface remoteBusinessIntf;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,
    IOException {
        String name = request.getParameter("name");
        if (name == null || name.length() == 0) {
            name = "anonymous";
        }
        response.getWriter().write(remoteBusinessIntf.sayHello(name));
    }
}
```

Run it!

After a very hard 4-minute development it's time to give it a shot and see if it works. Not much time left so hurry up!

Right-click on the **SampleEAR** project and select **Run As > Run on Server**. When *Run On Server* popup window comes up, select the **Always use this server when running this project** checkbox. Leave the rest as is.

Run On Server

Define a New Server

Choose the type of server to create

How do you want to select the server?

☐ Choose an existing server

☒ Manually define a new server

Server's host name:

[Download additional server adapters](#)

Select the server type:

- Apache
 - Apache Geronimo v1.0 Server
 - Apache Geronimo v1.1 Server
 - Apache Geronimo v2.0 Server
- +
- IBM
- +
- JBoss
- +
- ObjectWeb
- +
- Oracle

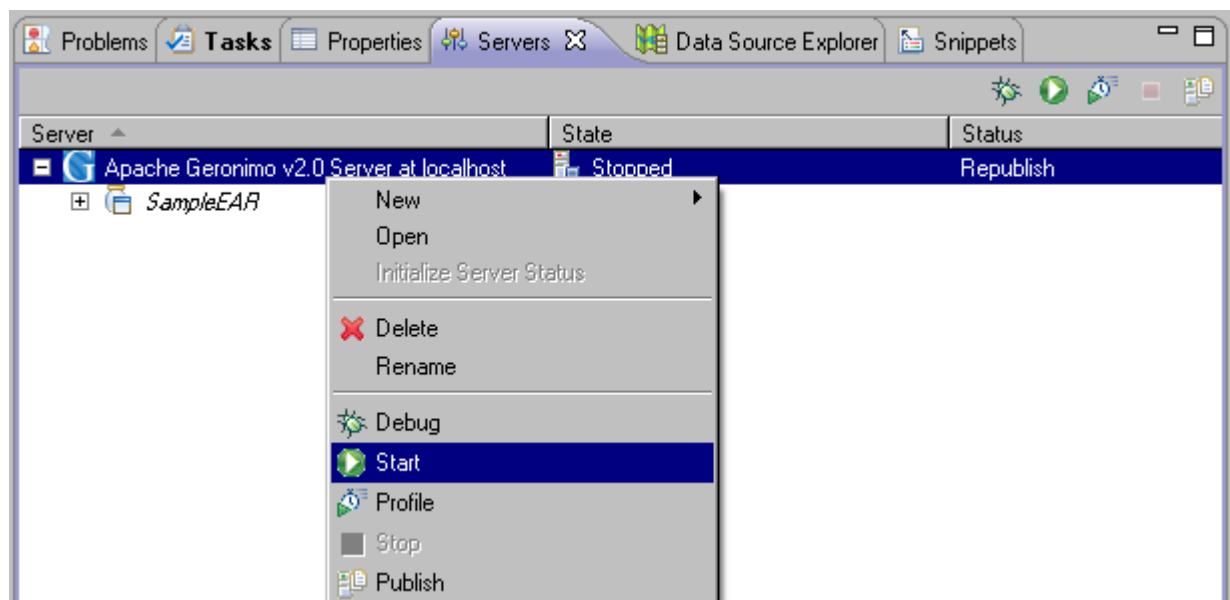
Apache Geronimo v2.0 Server

Server runtime:

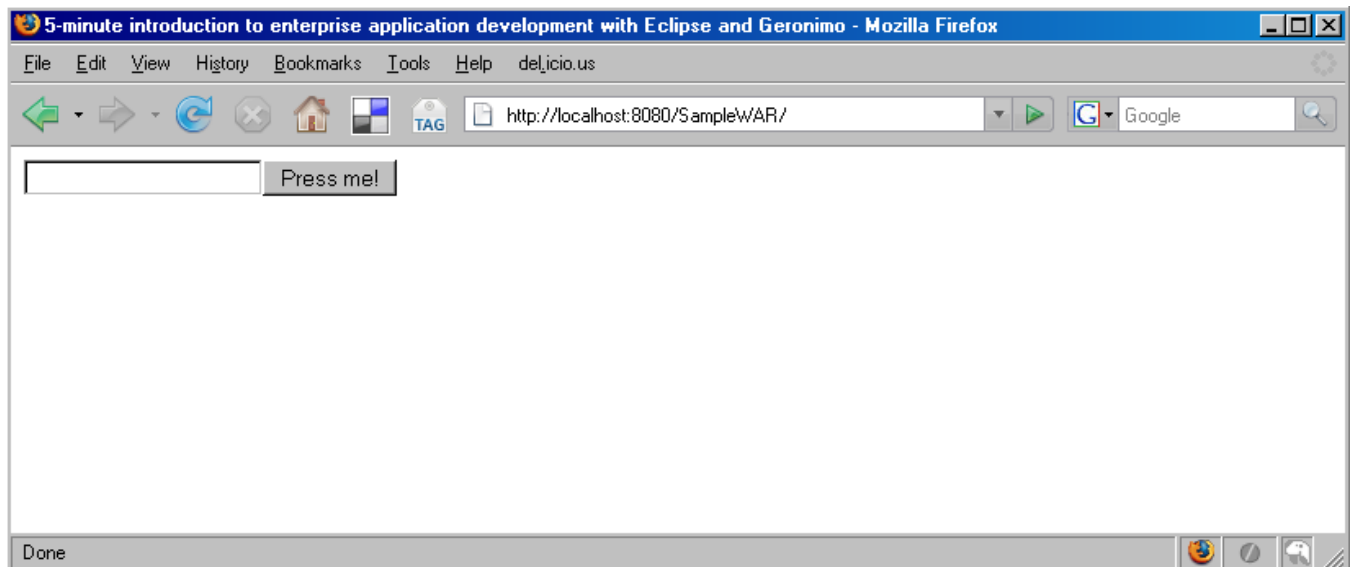
☐ Always use this server when running this project

Press **Finish**.

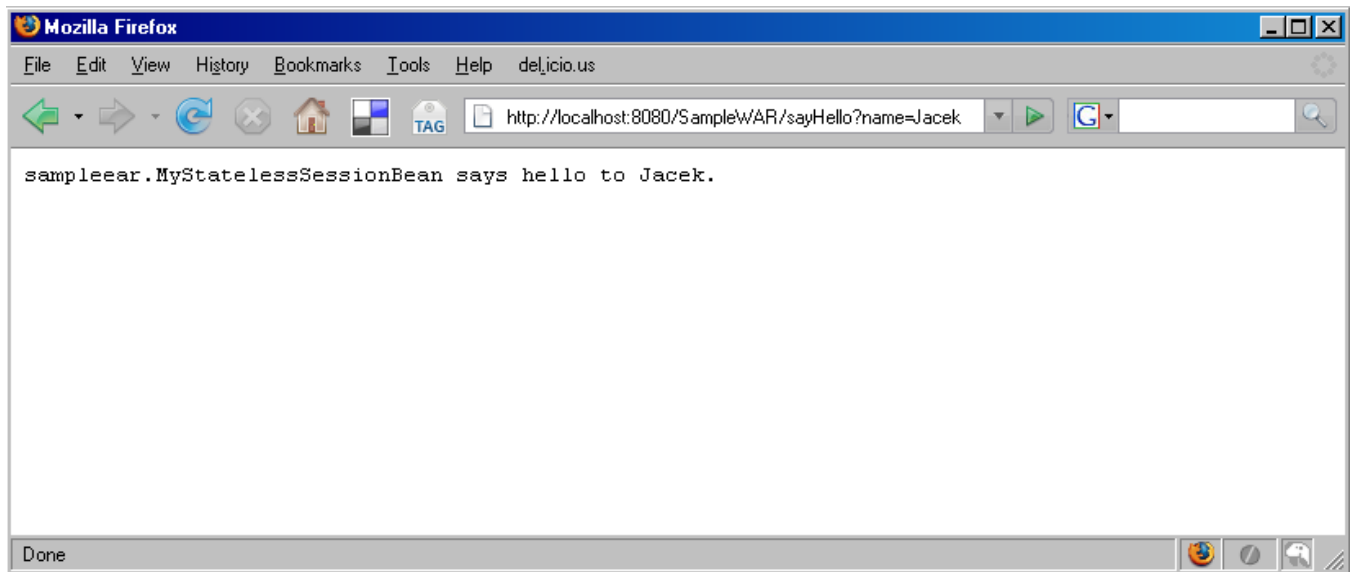
The server's stopped so nothing happens (from a user's perspective at least). Open up the **Servers** tab and right-click on **Apache Geronimo v2.0 Server at localhost** and select **Start**.



After a few seconds, Geronimo should be up and running with the enterprise application published. Open up the browser of your choice and go to <http://localhost:8080/SampleWAR/>.



Type in any name you want, e.g. Jacek and press **Press me!** button.



The tutorial's finished. Let us know how much it actually took and if it worked at all 😊