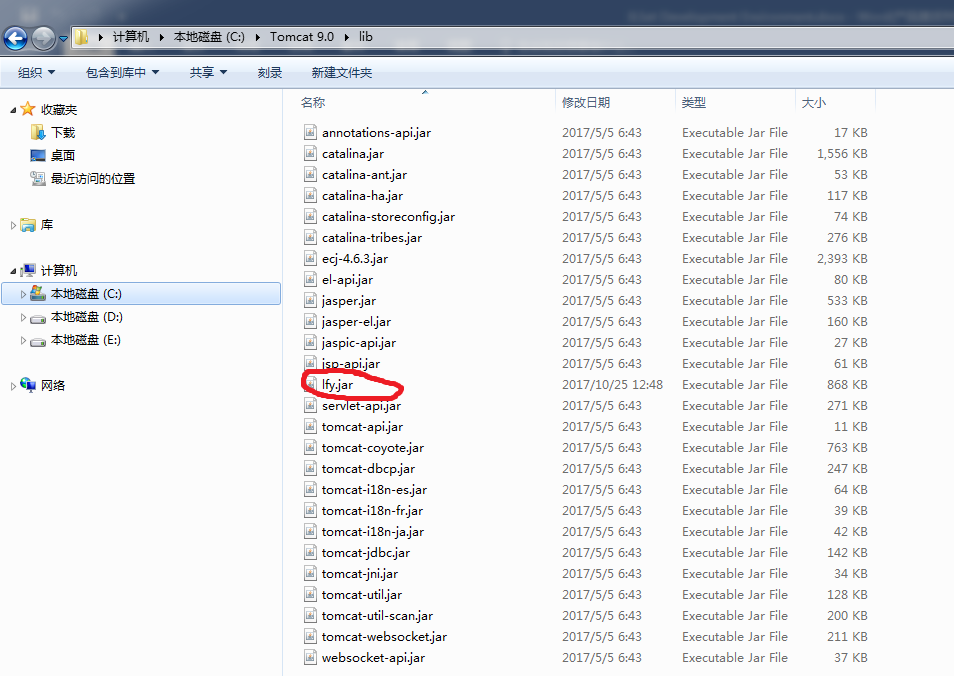
## Put lfy.jar in lib directory of Tomcat

The directory content appears as following.

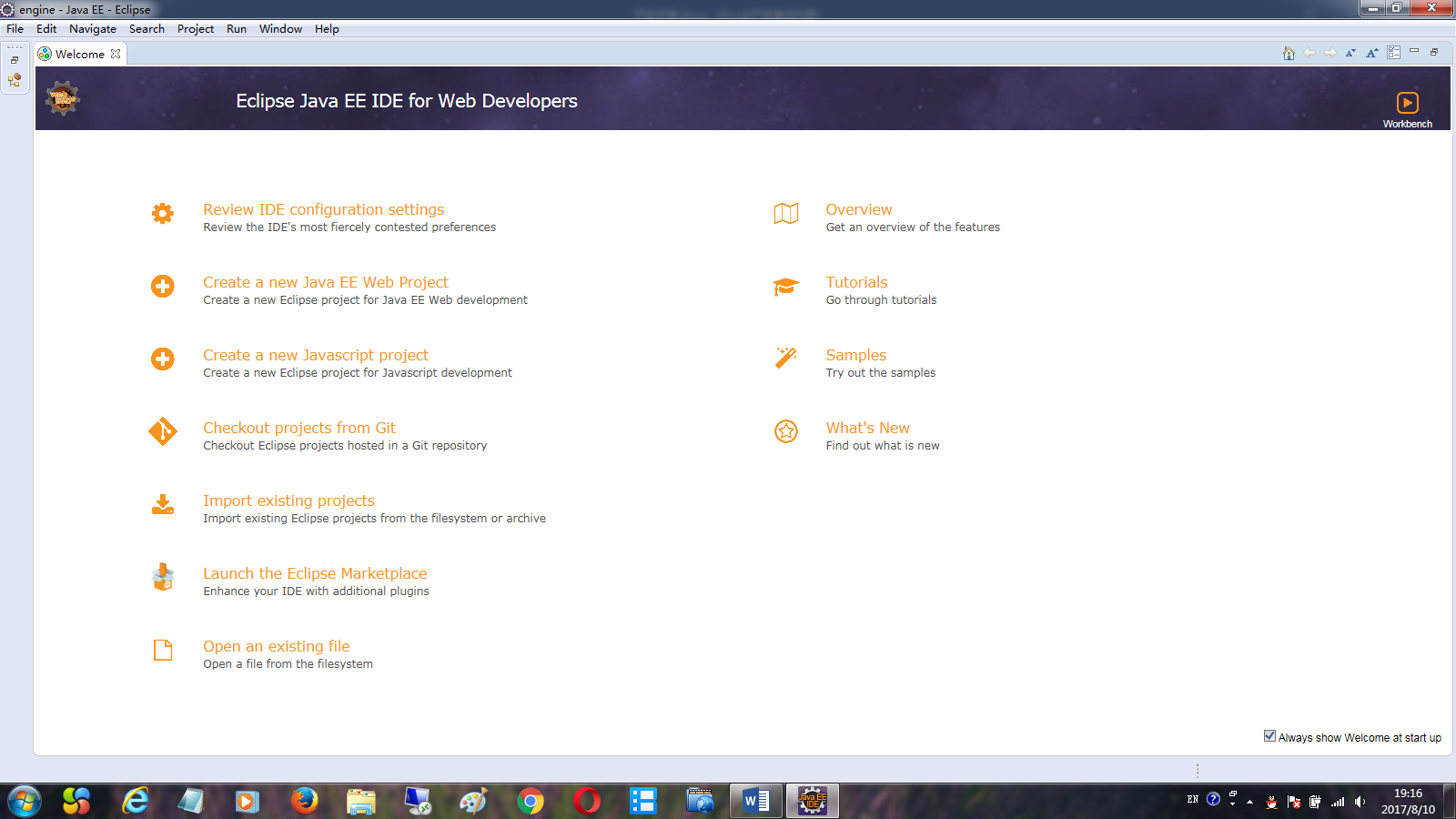


## Copy data in your computer

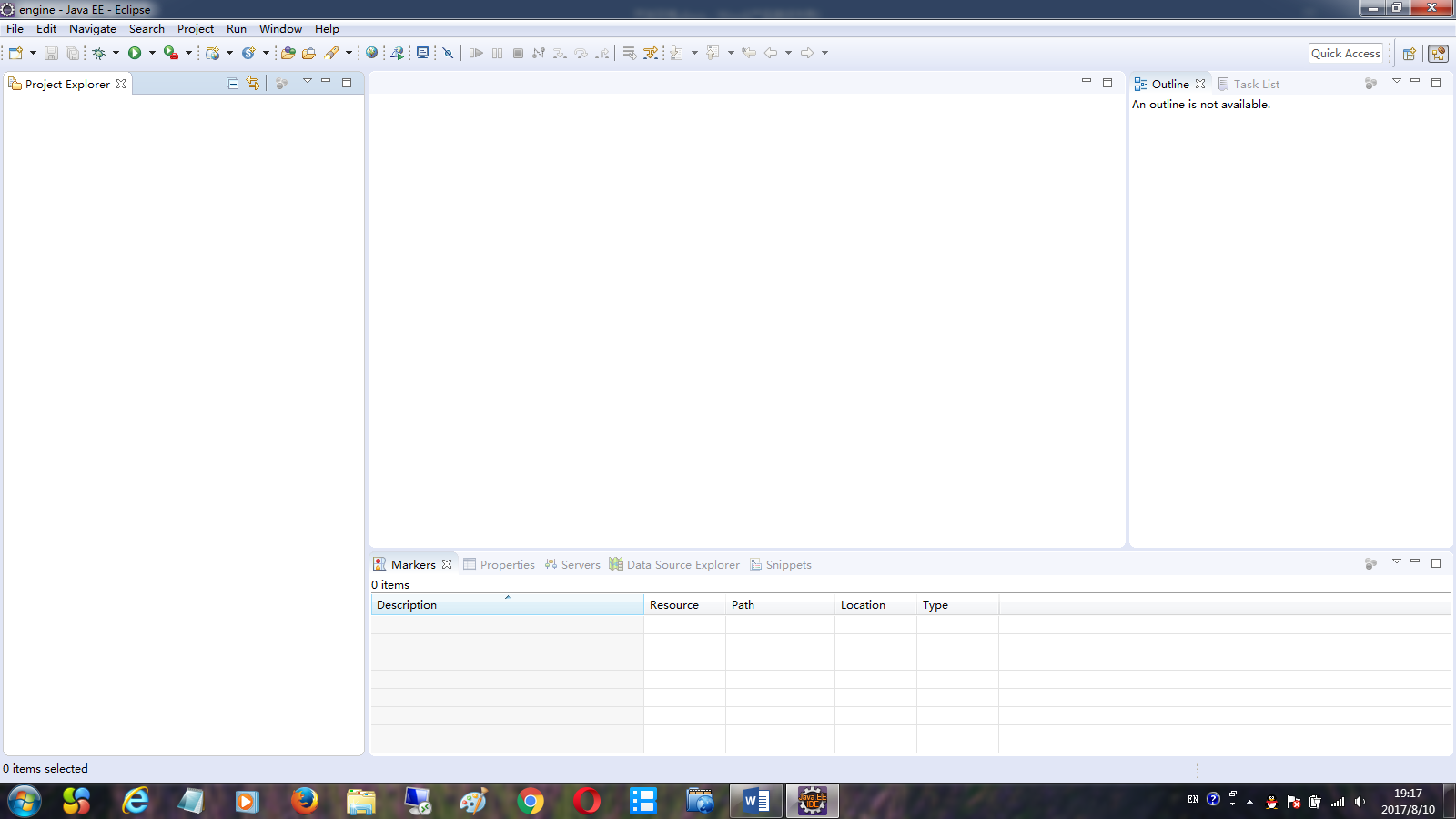
Copy scene data to your computer.

## Start Eclipse

When you first start Eclipse, it looks like following this.



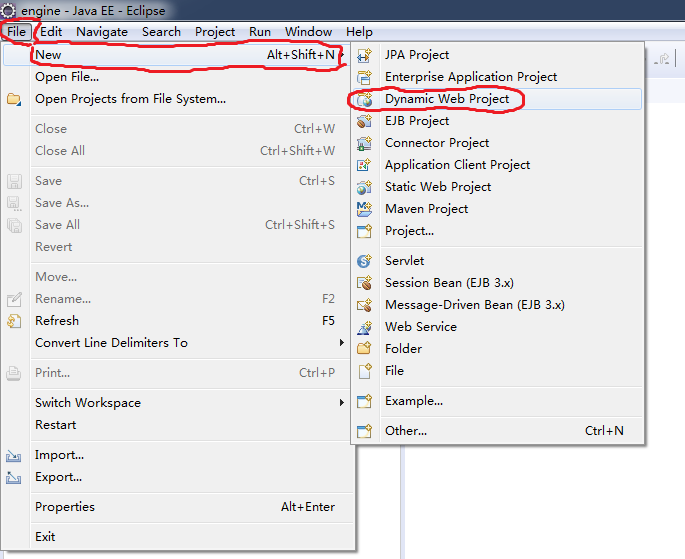
Close the Welcome view, it looks like following this.



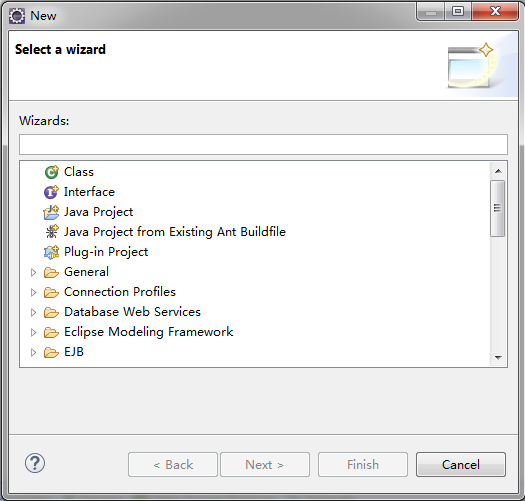
Now eclipse is OK, now you can create dynamic Web Project.

## Create a “dynamic Web Project”

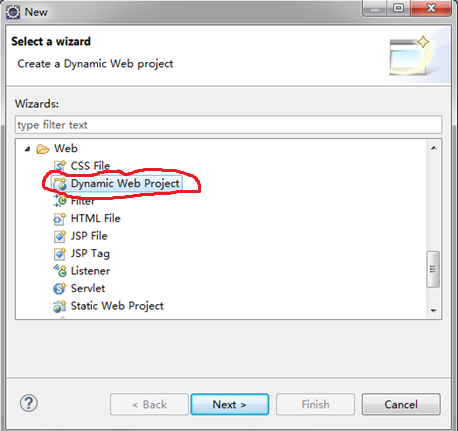
In Eclipse, click File->New->Dynamic Web Project as following menu.



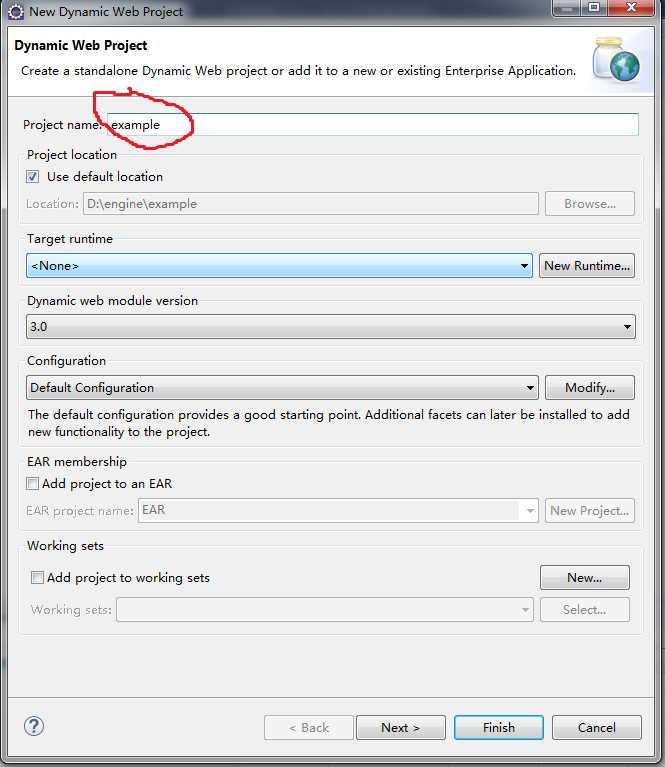
If Dynamic Web Project does not exist in the menu, click File->New->Other at bottom, a dialog as following appears.



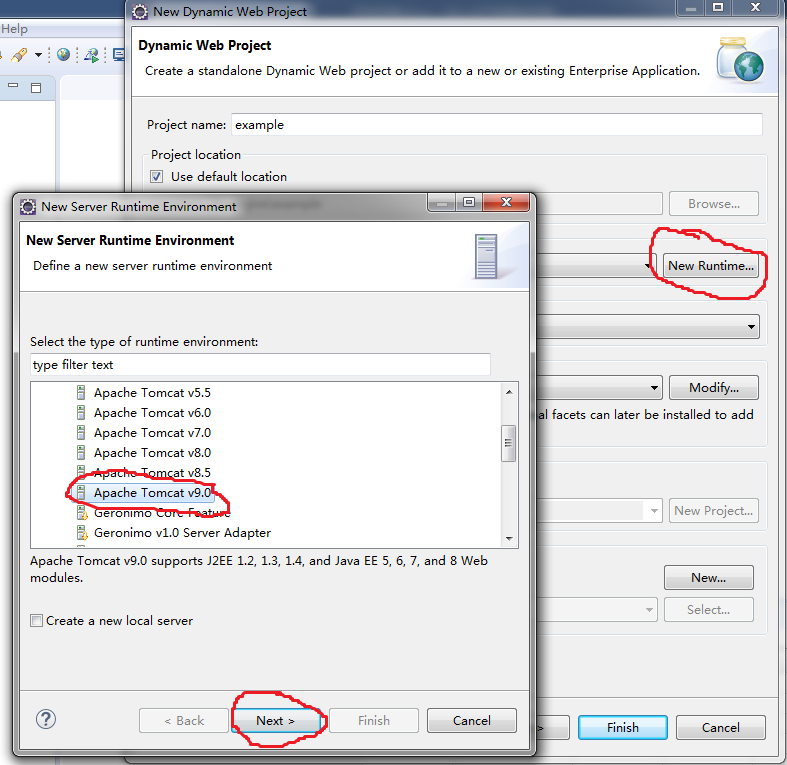
Search the Web item, you will find Dynamic Web Project.



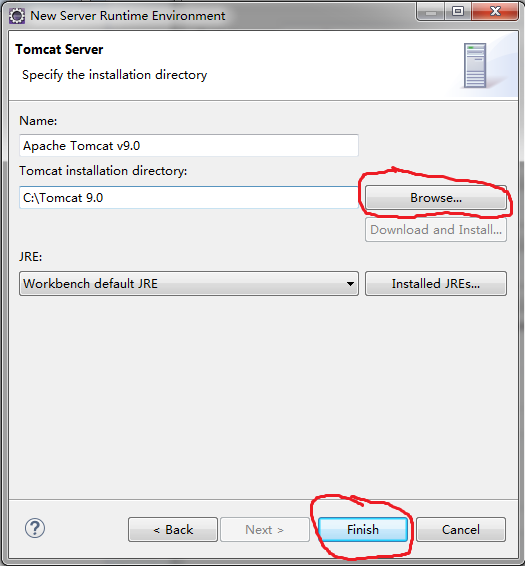
Click Next button, A dialog as following appears. First input your project name as following.



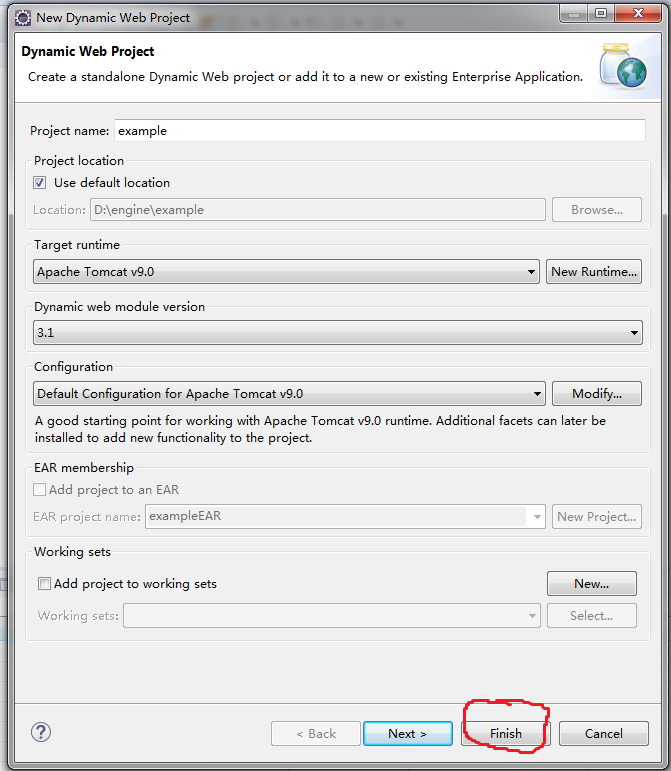
Next, select target runtime



Click Next > to specify the installation directory of Tomcat. Its dialog appears as following. After specifying Tomcat directory, click Finish button.

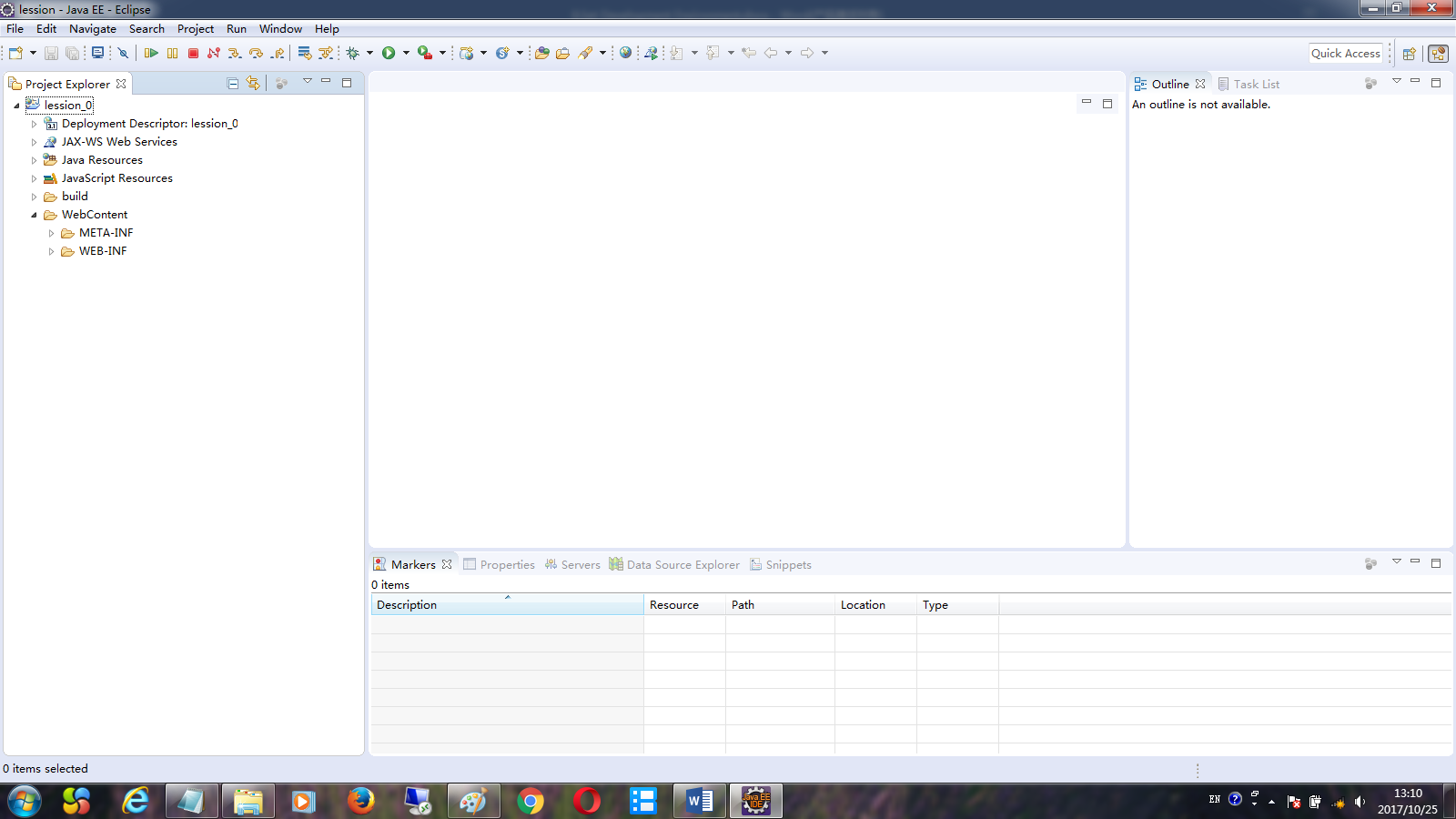


Now the new project dialog is as following.



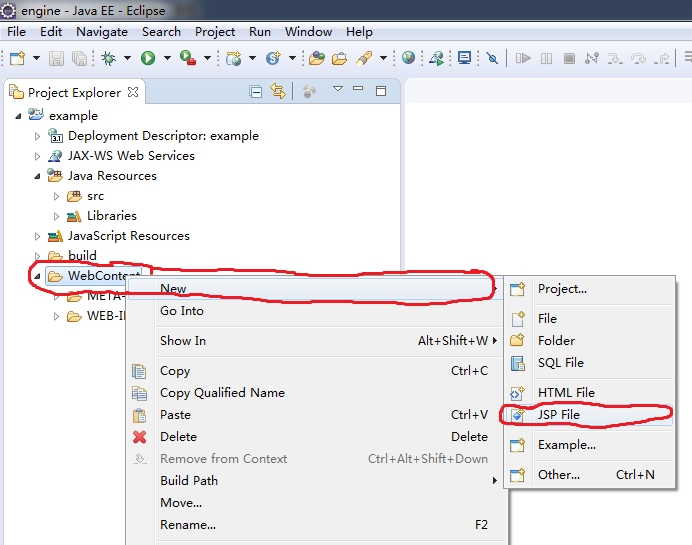
Click Finish button, dynamic Web Project is created successfully.

Now Eclipse looks like following

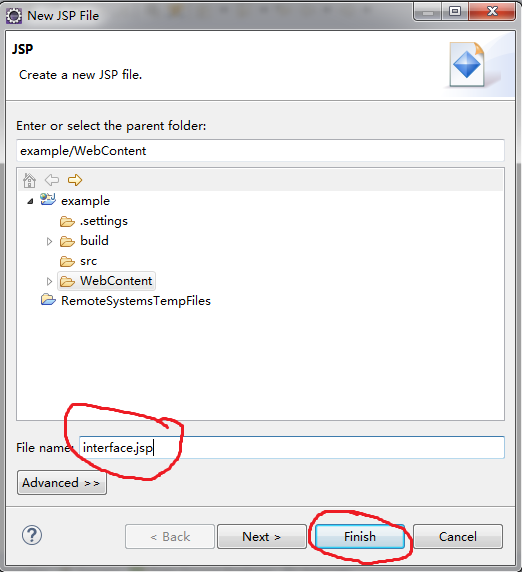


## Create file interface.jsp

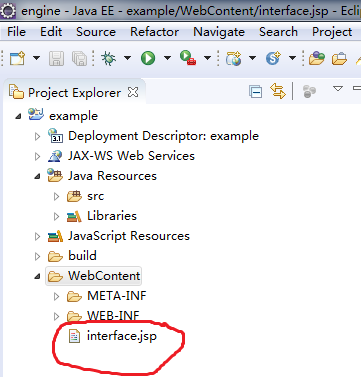
Right-click WebContent directory, New a jsp file.



A dialog as following appears. Specify file name as interface.jsp. Then click Finish.



The file directory looks as following.



Edit file interface.jsp.

First delete all content in the file, put following code in it. It looks as following code.

Then modify data configure file name and temporary directory names to your own data and temporary directory.

<%@ page language="java" %>

<jsp:useBean id="http\_service" class="kernel\_service.http\_service" scope="application"></jsp:useBean>

<%

out=http\_service.service\_call(

"E:/data/configure.txt", /\* data configure file \*/

"F:/temp/proxy.txt", /\* proxy configure file \*/

out,request,response,application,

pageContext,getClass().getClassLoader());

%>

## Create file example.html

The operation is similar with that of interface.jsp. The only difference is file type. example.html is html file,

First delete all content in these file, then put following code in it.

<html>

<head>

<script type="text/javascript"

src=" interface.jsp?function\_name=construct\_render\_object">

</script>

<script type="text/javascript">

var render\_object;

function body\_onload()

{

construct\_render\_object(

document.getElementById("my\_canvas"),

"NoName","NoPassword","chinese",

"sky\_box","","display\_part:sky\_box\_part","",

5,[],

function(my\_render\_object)

{

render\_object=my\_render\_object;

document.title=render\_object.title;

}

);

}

</script>

</head>

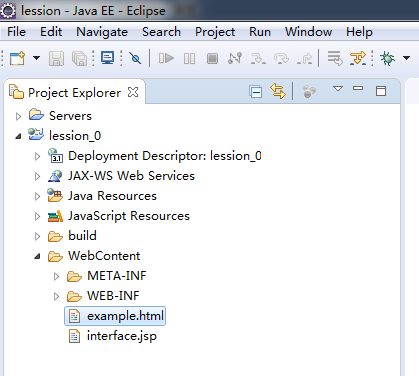
<body onload="body\_onload();">

<canvas id="my\_canvas" tabindex="0" width=1585 height=750></canvas>

</body>

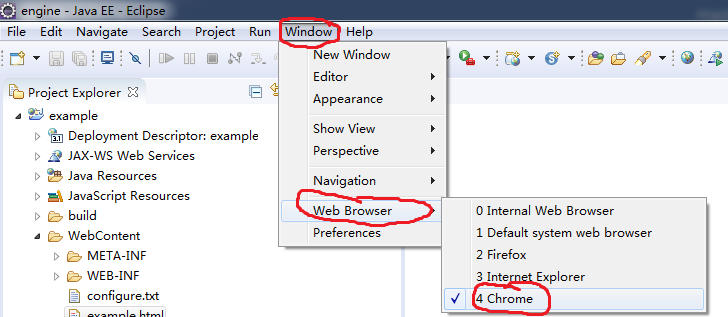
</html>

Now the content directory look as following.



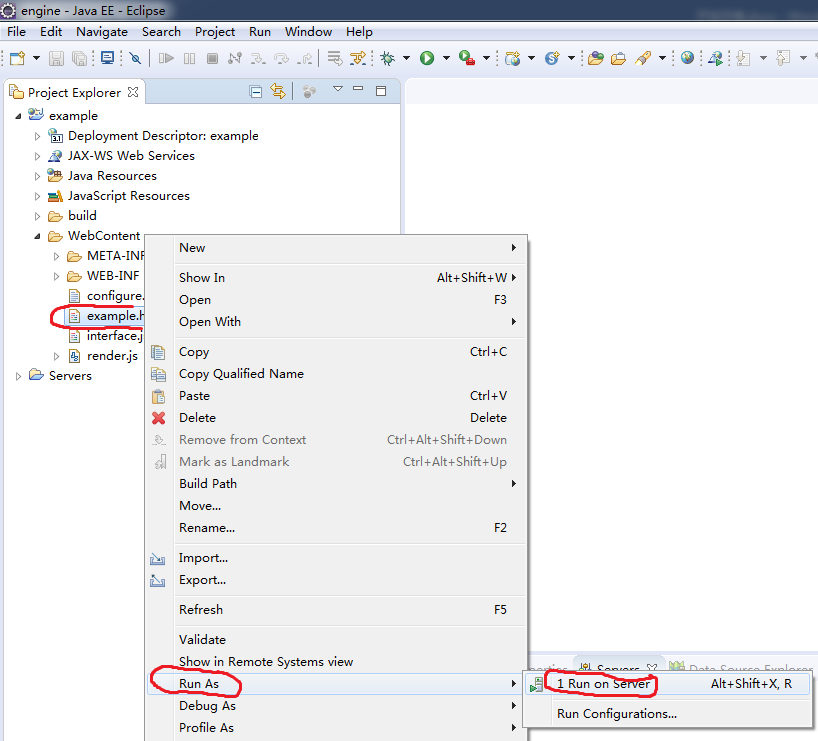
## Specify Web Browser

In Eclipse environment, Click Windows -> Web Browser, specify Web Browser.

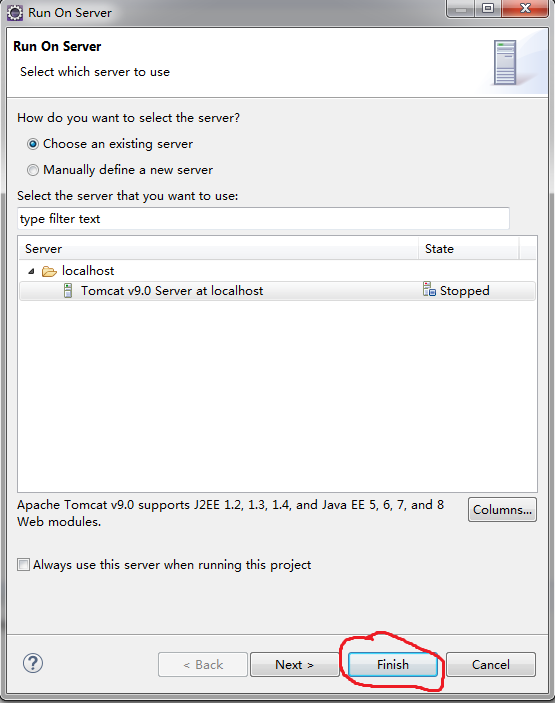
****

## Test example.html

Right-click example.html, appears a menu as following. Click Run on Server to start Web server and browser example.html.

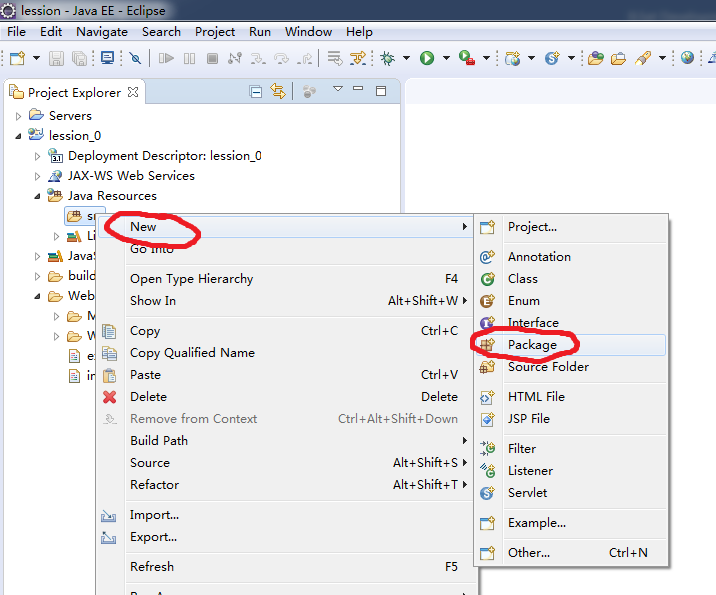
****

The following dialog appears. Click Finish button, Web server will be started, and example.html will display on Browser.

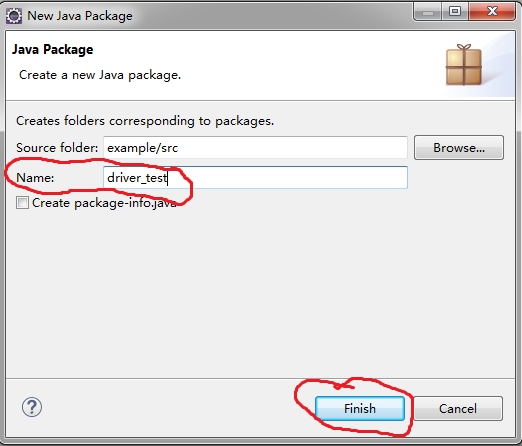
****

## Create a Java package

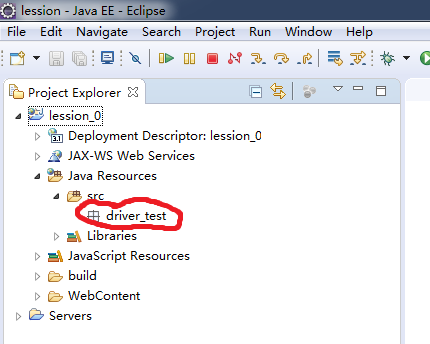
Right-click src directory, New a Java package file as following.



A dialog as following appears. Input package Name, then click Finish button.

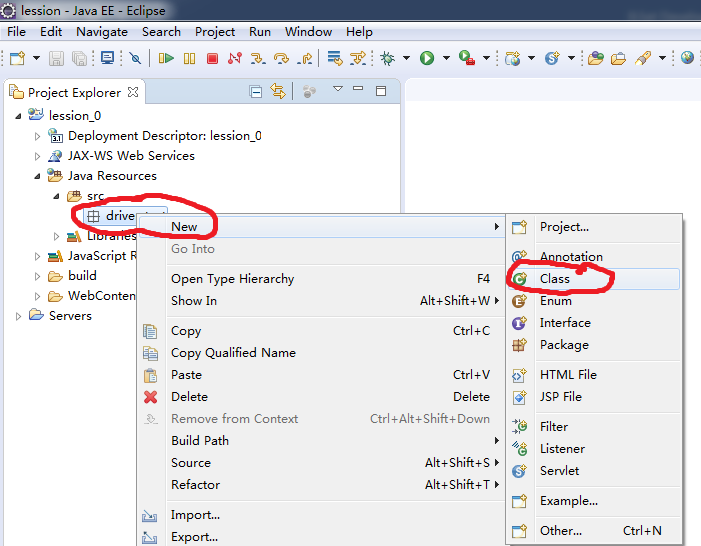


A new Java package will be created, its directory appears as following.

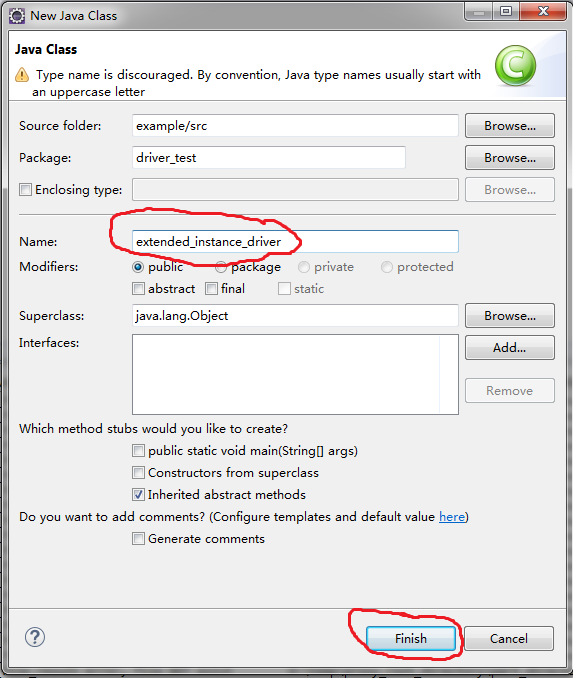


## Create Java classes

Right-click Java package just created, New a Java clas as following.



A dialog as following appears. Input class Name(extended\_instance\_driver), then click Finish button.



A new Java class will be created.

In this way, create four Java class:

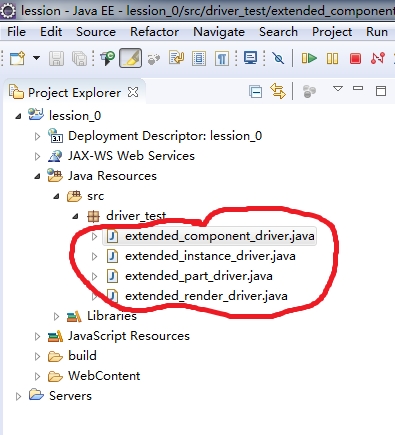
extended\_instance\_driver

extended\_component\_driver

extended\_part\_driver

extended\_render\_driver

Now the directory appears as following.



## Put our Java code in four created Java class

Double-click the four created Java class to edit their code.

Clip our program code, paste them to created Java class.

## Configure our driver and debug it

Edit file

data\ shaders\shaders.txt

or

data\assemble\test\_part\shaders\type\_shaders.txt

or

data\assemble\test\_part\assemble\part\shaders\scene\_shaders.txt

Use our render driver in part configuration file and try it.

The first shader configuration is for the whole system. The second is for the scene of same type. The third is for single scene.

Parts defined in first file are loaded in engine when engine started, parts defined in second and third file are loaded in engine whenever scene is created.

We suggest that you try and debug render driver in the last two shader configuration files.