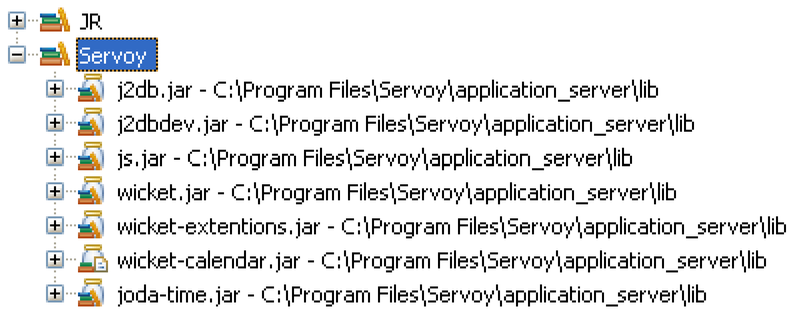
Using the Click framework as a Servoy client

I packaged my little click solution as an Eclipse project to use with Servoy (or any bare Eclipse distribution) so that the sources are enclosed, it's not that complicated to deploy. Follow the instruction closely to get this to work:

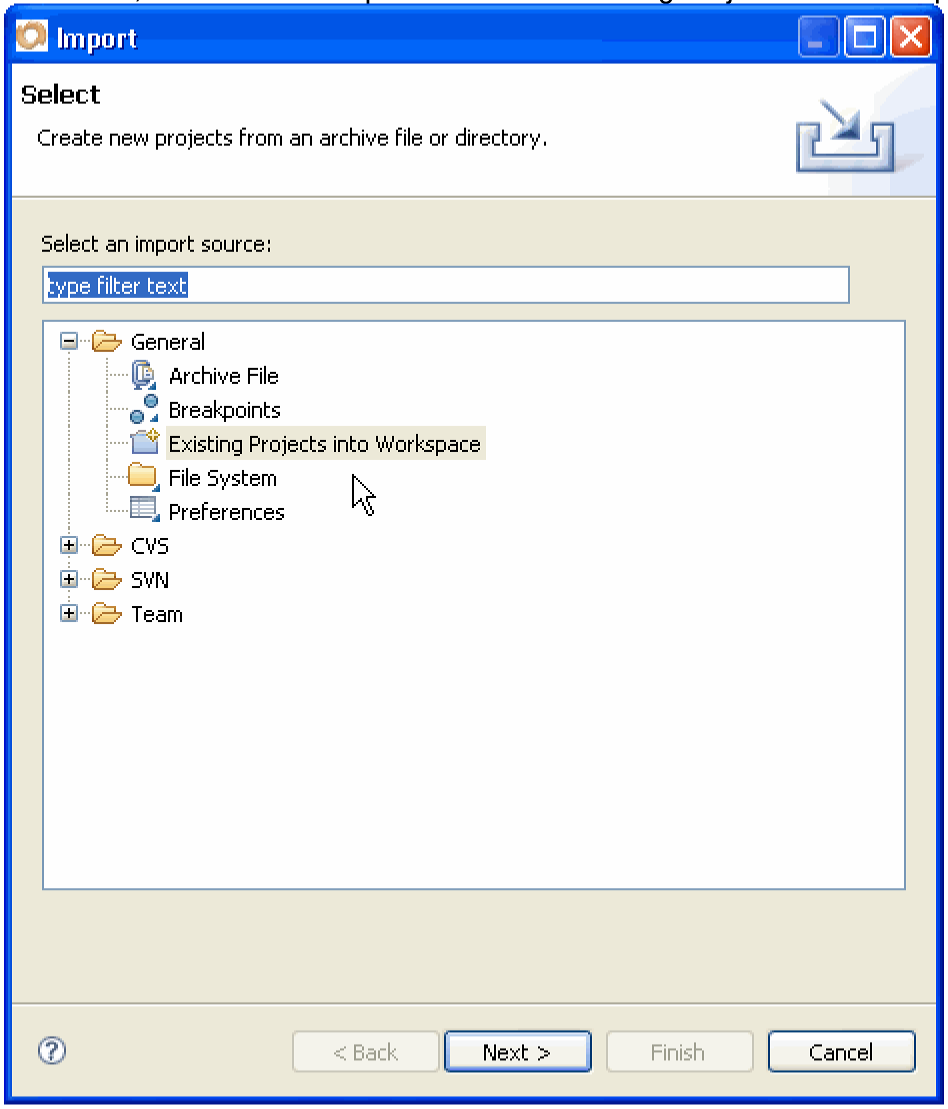
1. *Import the "servoy-click" project.*

First, make sure that you have a "Servoy" User Library configured in you Eclipse distribution, it should contain these jars:

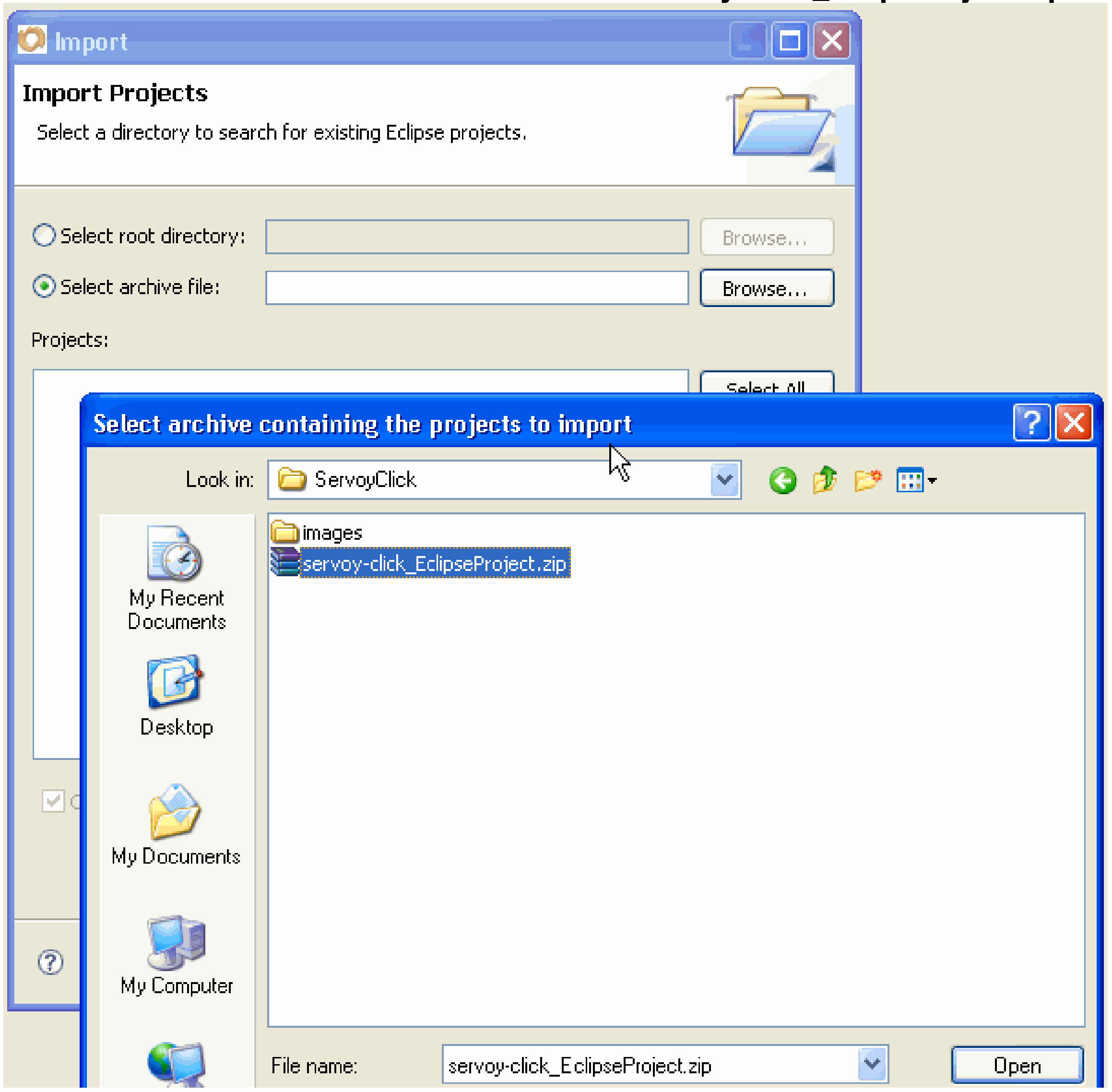


```
JR
Servoy
    j2db.jar - C:\Program Files\Servoy\application_server\lib
    j2dbdev.jar - C:\Program Files\Servoy\application_server\lib
    js.jar - C:\Program Files\Servoy\application_server\lib
    wicket.jar - C:\Program Files\Servoy\application_server\lib
    wicket-extentions.jar - C:\Program Files\Servoy\application_server\lib
    wicket-calendar.jar - C:\Program Files\Servoy\application_server\lib
    joda-time.jar - C:\Program Files\Servoy\application_server\lib
```

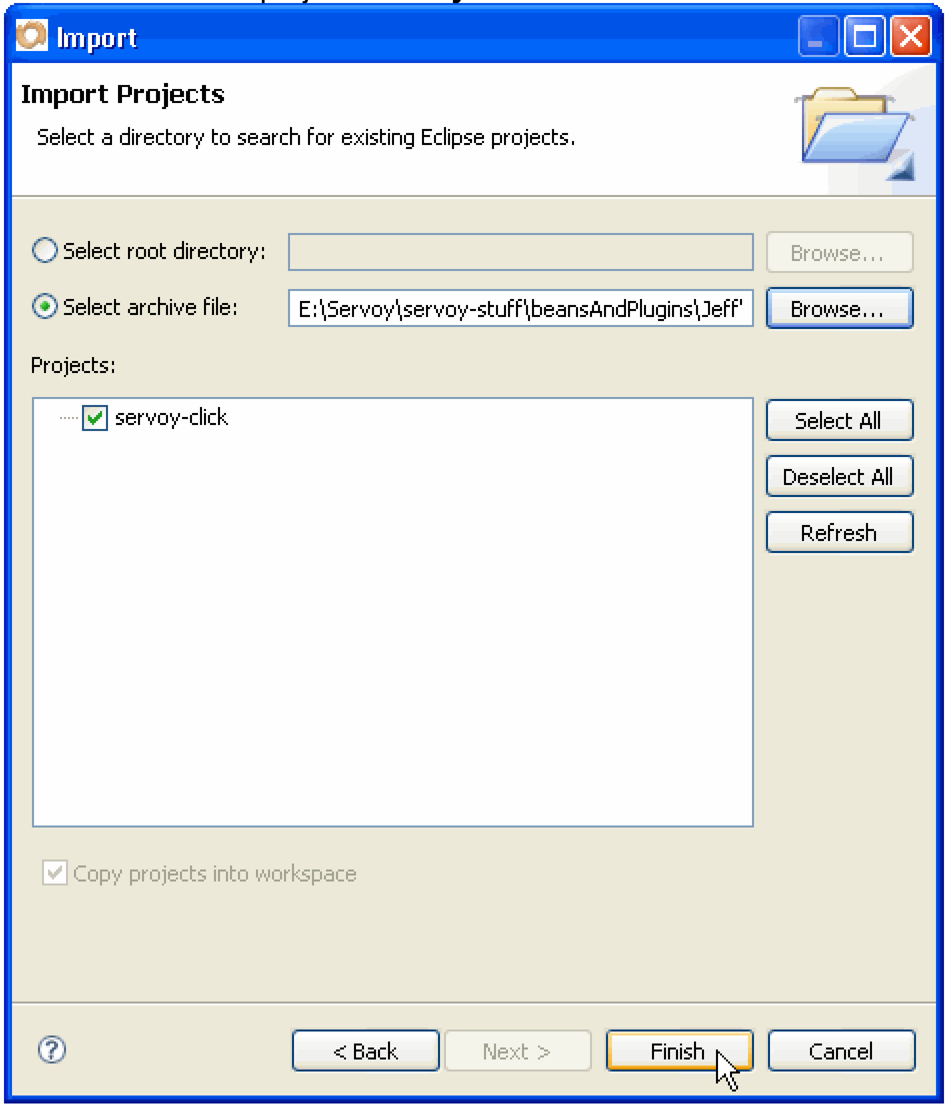
Now import into your Eclipse distribution of choice – I will use Servoy.
To do this, menu “File > Import” – choose “Existing Projects into Workspace”



Choose “Select archive file” and browse to the file “**servoy-click_EclipeProject.zip**”:

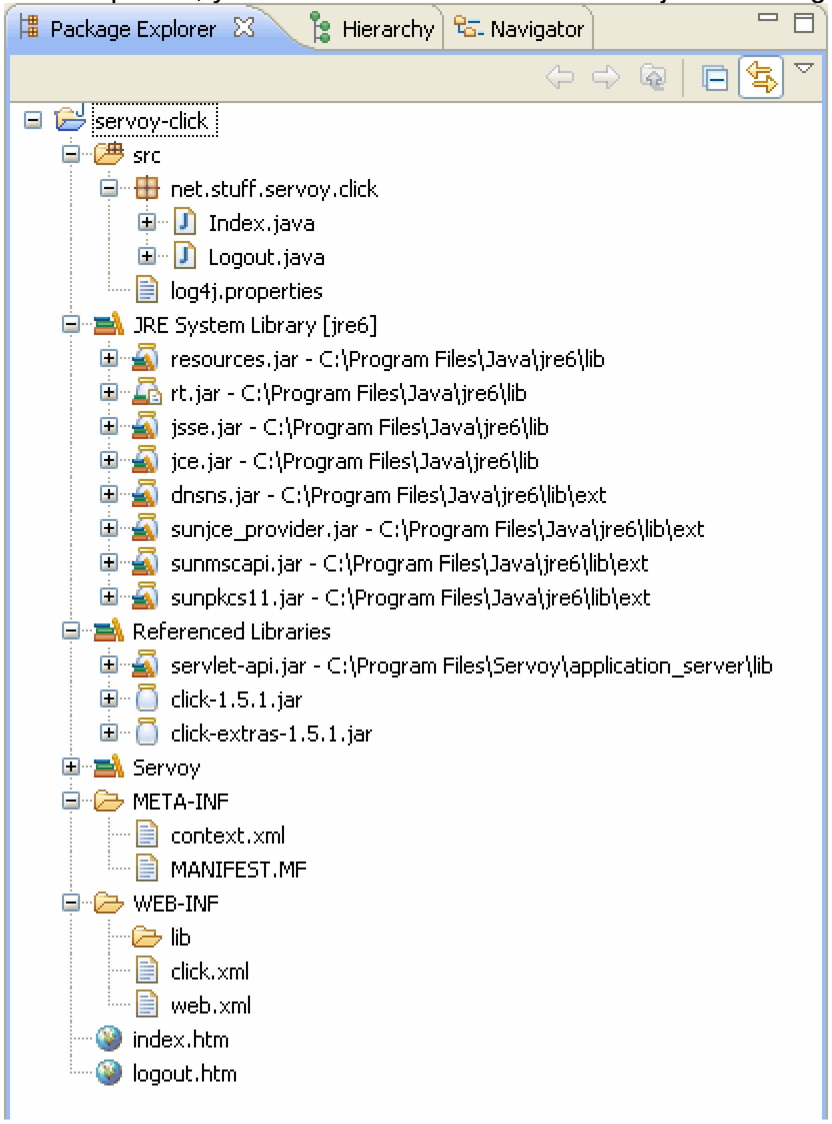

Import
Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Browse...
Select archive file:
Browse...
Projects:
Select All
Select archive containing the projects to import
Look in: ServoyClick
images
servoy-click_EclipseProject.zip
My Recent Documents
Desktop
My Documents
My Computer
File name: servoy-click_EclipseProject.zip
Open

You should see the project “**servoy-click**” contained in the archive like this:



Make sure it is selected and click “Finish”

Once imported, you should see this content in the java “Package” view:



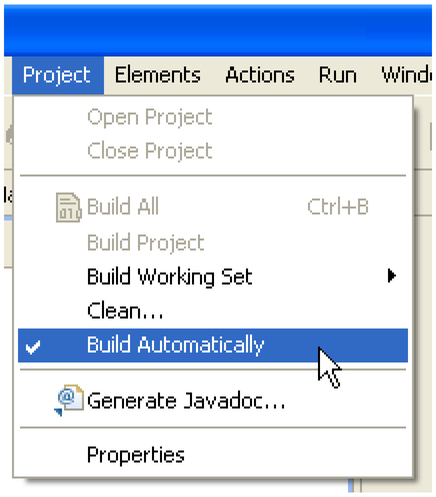
```
servoy-click
  src
    net.stuff.servoy.click
      Index.java
      Logout.java
    log4j.properties
  JRE System Library [jre6]
    resources.jar - C:\Program Files\Java\jre6\lib
    rt.jar - C:\Program Files\Java\jre6\lib
    jsse.jar - C:\Program Files\Java\jre6\lib
    jce.jar - C:\Program Files\Java\jre6\lib
    dnsns.jar - C:\Program Files\Java\jre6\lib\ext
    sunjce_provider.jar - C:\Program Files\Java\jre6\lib\ext
    sunmscapi.jar - C:\Program Files\Java\jre6\lib\ext
    sunpkcs11.jar - C:\Program Files\Java\jre6\lib\ext
  Referenced Libraries
    servlet-api.jar - C:\Program Files\Servoy\application_server\lib
    click-1.5.1.jar
    click-extras-1.5.1.jar
  Servoy
  META-INF
    context.xml
    MANIFEST.MF
  WEB-INF
    lib
    click.xml
    web.xml
  index.htm
  logout.htm
```

If you don’t have JRE System Library [jre6] but another standard JRE selected, that’s fine, although you should have a JRE 1.5+ minimum.

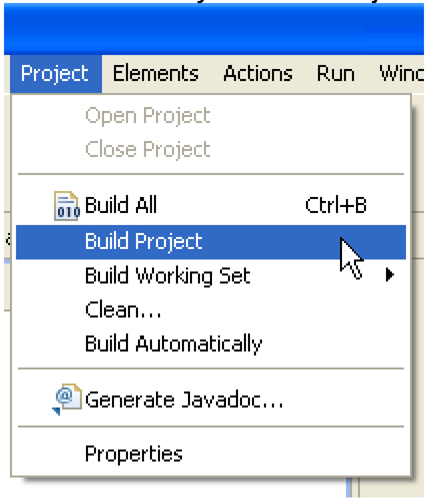
Now once imported the project should have compiled. Make sure you have no “red” markers on the “Index.java” and “Logout.java”

Make sure that the project did compile.

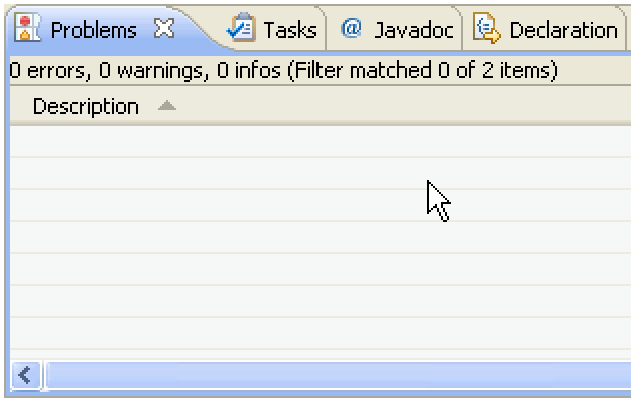
- You should have the option “Build automatically” checked in the Project menu:



- if not try a “Build Project” or “Build All”:



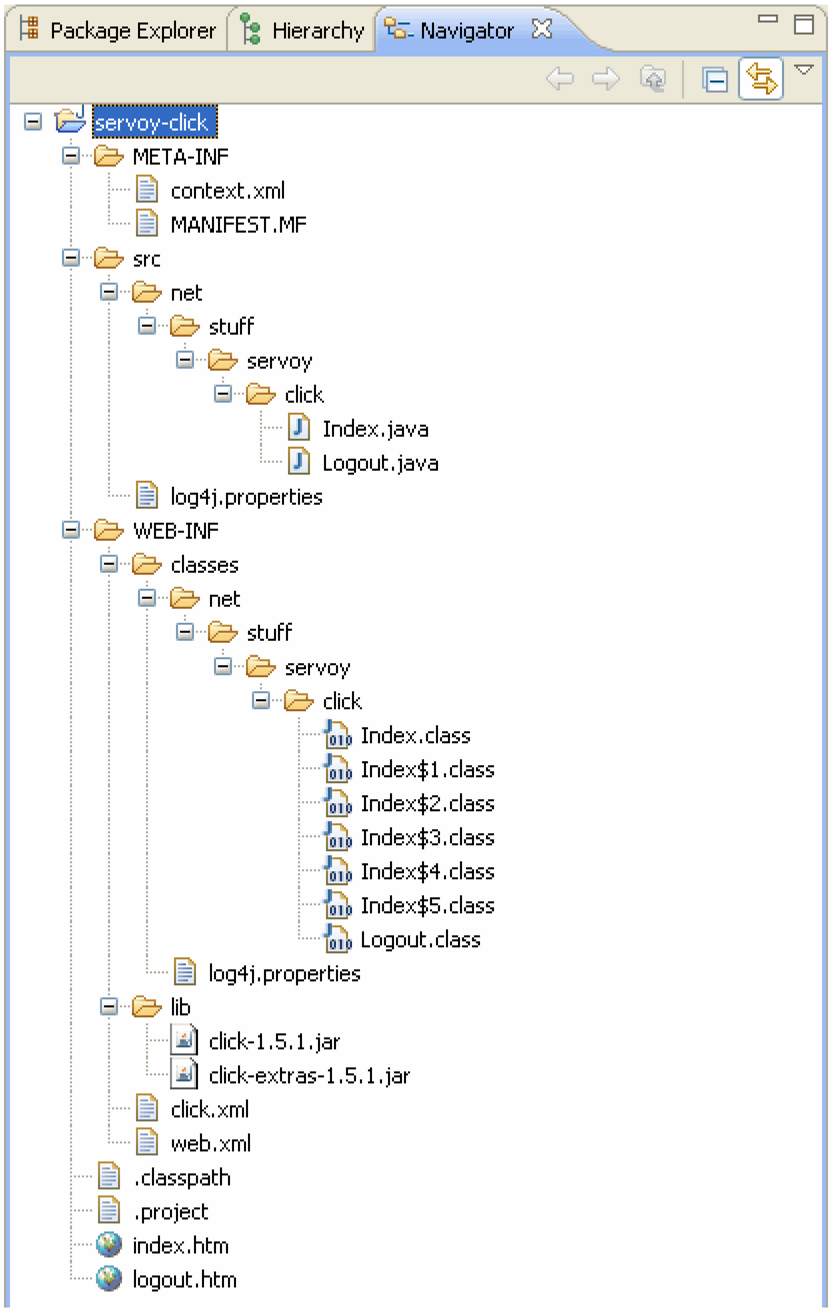
You should also go to the “Problems” view and check that there is no problem with the build:



0 errors, 0 warnings, 0 infos (Filter matched 0 of 2 items)

This is to ensure that the java files were correctly compiled, they should be placed in a /WEB-INF/classes/ - but you cannot see this folder in the Package view.

Alternatively, you could go to the Navigator view to check the structure of the project (or directly check it on your disk); in the Navigator view it should look like this:

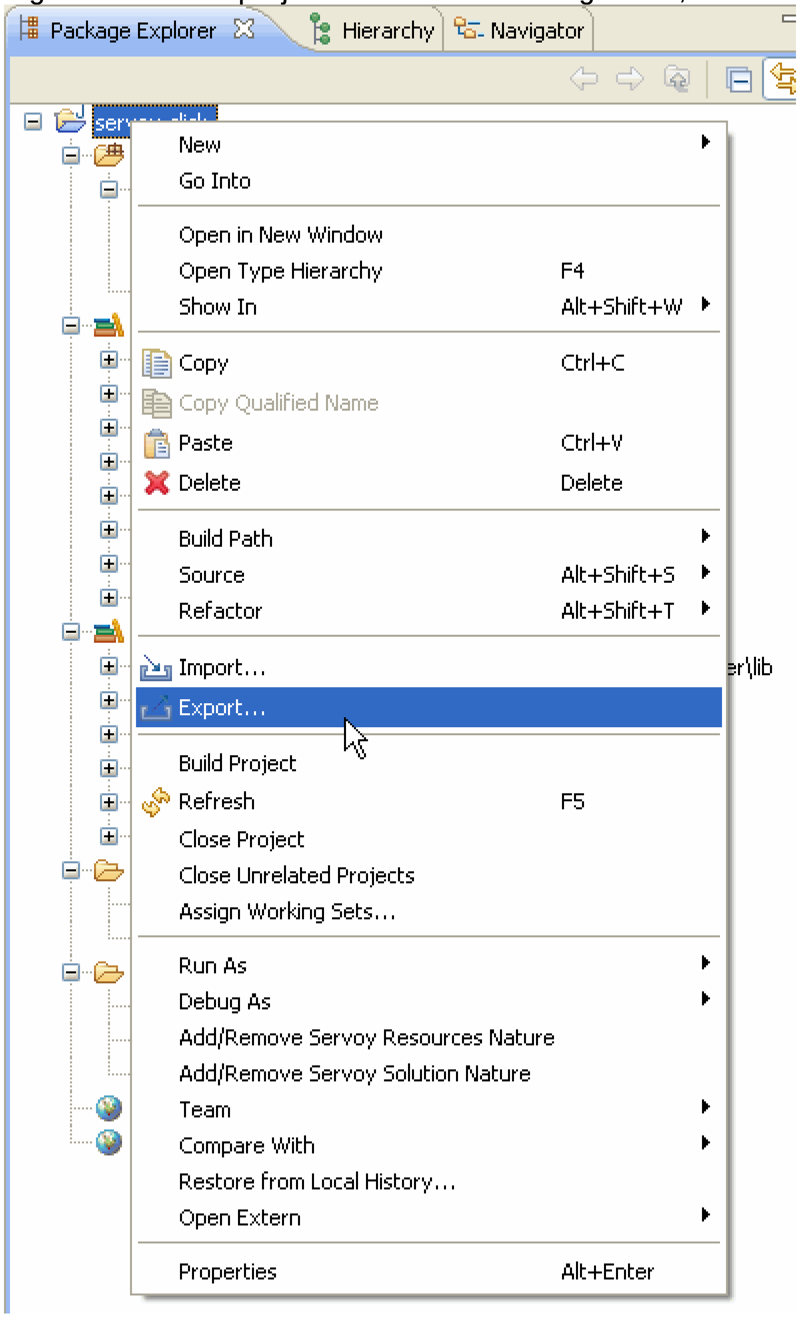


```
servoy-click
  META-INF
    context.xml
    MANIFEST.MF
  src
    net
      stuff
        servoy
          click
            Index.java
            Logout.java
    log4j.properties
  WEB-INF
    classes
      net
        stuff
          servoy
            click
              Index.class
              Index$1.class
              Index$2.class
              Index$3.class
              Index$4.class
              Index$5.class
              Logout.class
      log4j.properties
    lib
      click-1.5.1.jar
      click-extras-1.5.1.jar
    click.xml
    web.xml
  .classpath
  .project
  index.htm
  logout.htm
```

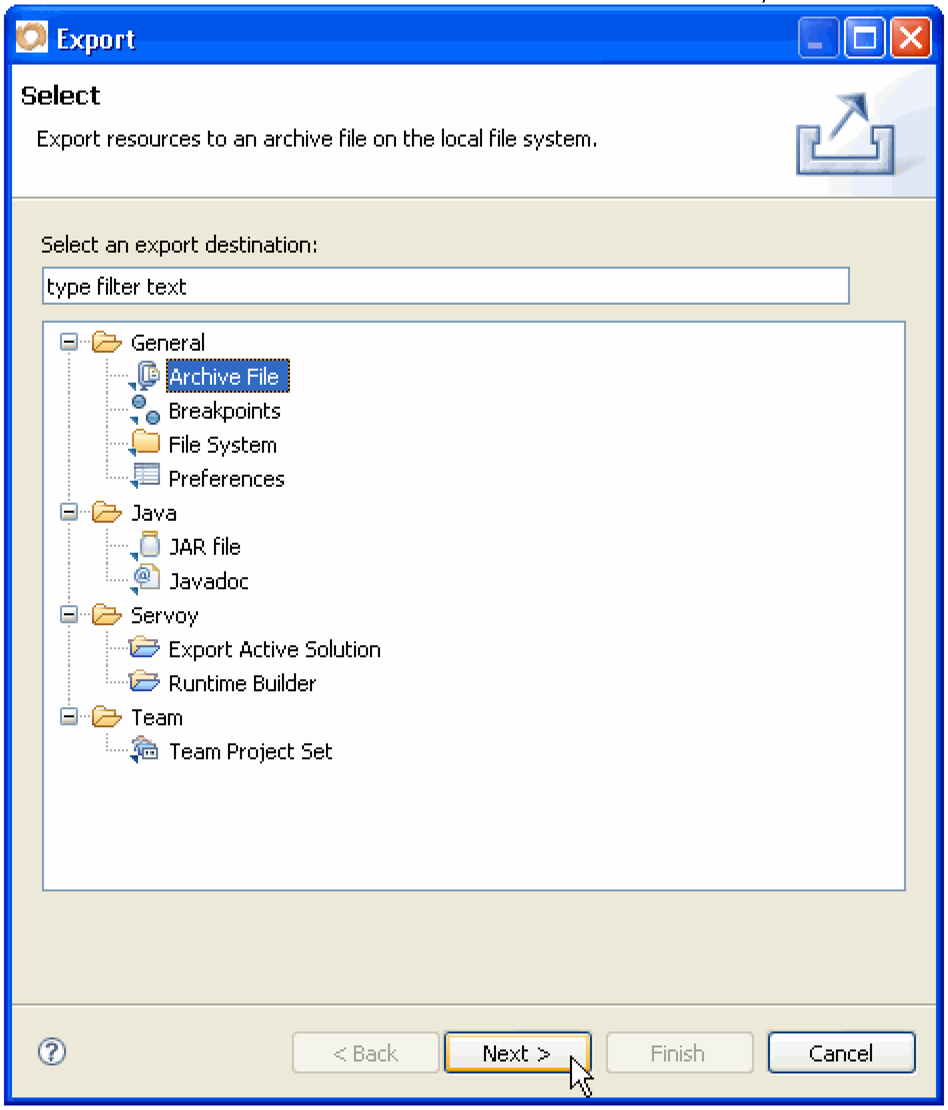
Now that we are sure that the project built correctly and that the structure is right, let’s create a “war” file (Web Archive).

2. *Create the war file for deployment:*

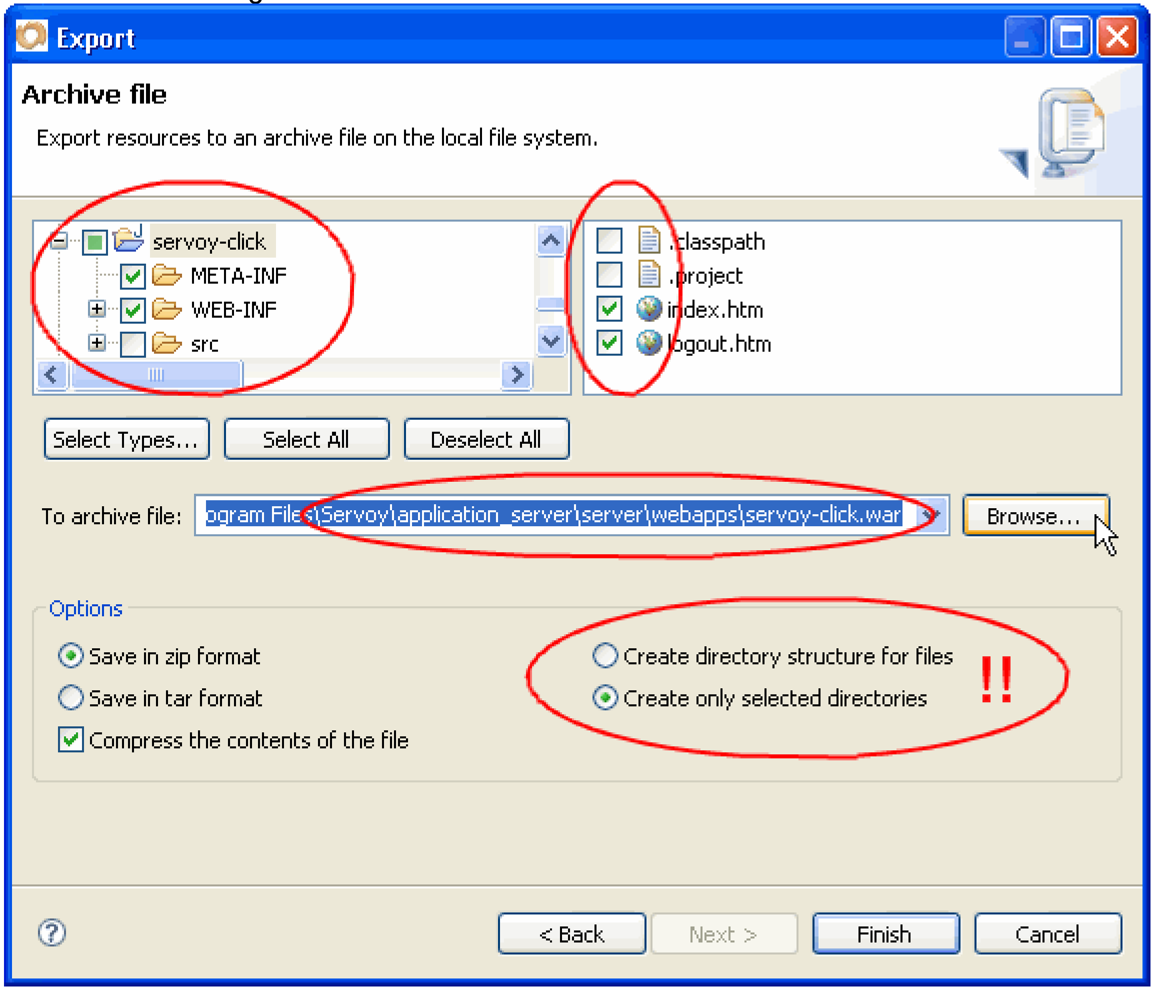
Right-click on the project folder in the Package view, and choose “Export…”



Choose “Archive file” under the “General” node of the tree, and click “Next >”



Create the following structure:



Export

Archive file

Export resources to an archive file on the local file system.

- [■] servoy-click
  - [✓] META-INF
  - [✓] WEB-INF
  - [ ] src

- [ ] .classpath
- [ ] .project
- [✓] index.htm
- [✓] logout.htm

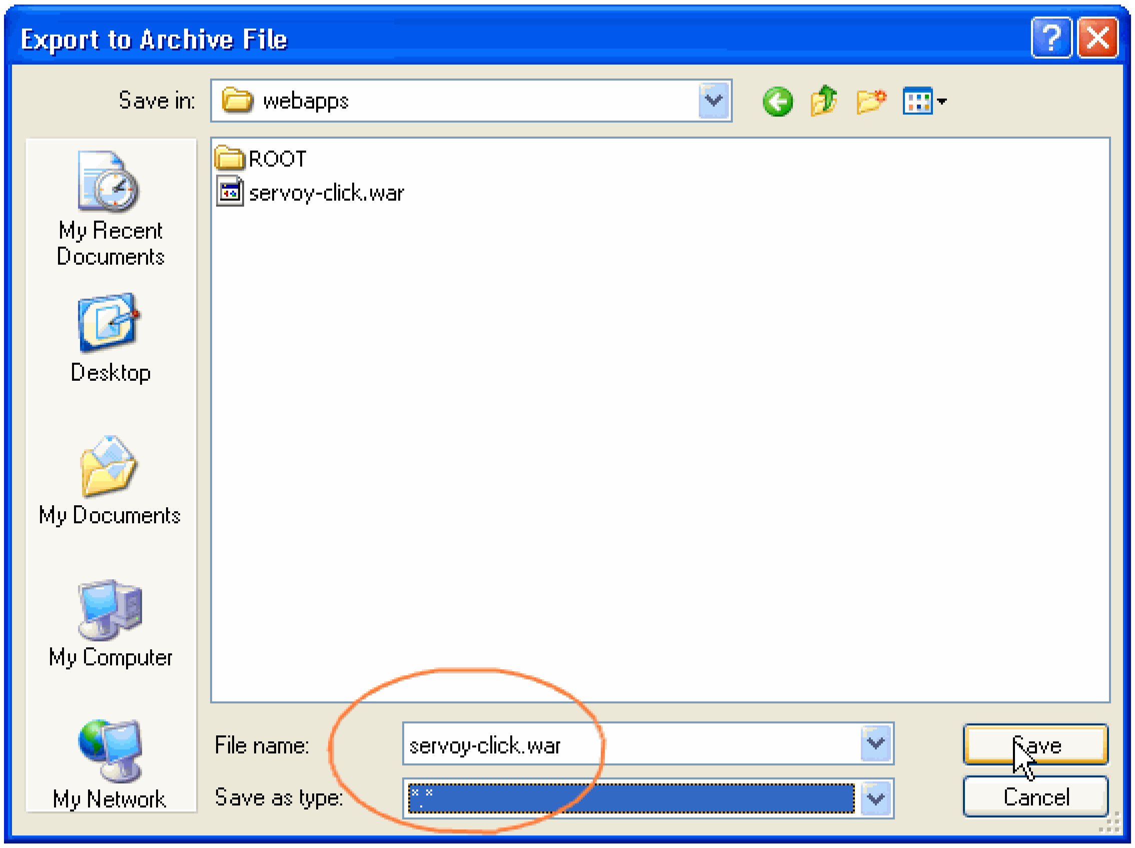
To archive file: ogram Files\Servoy\application_server\server\webapps\servoy-click.war

Options

- [● Save in zip format]
- ○ Save in tar format
- [✓] Compress the contents of the file
- ○ Create directory structure for files
- [● Create only selected directories]

- uncheck ".classpath" and ".project" in the "servoy-click" root folder
- uncheck the "src" folder
- check "Save in zip format" and "Compress the contents of the file"
- ensure that you have the option "Create only selected directories" checked!

Now browse to your “/$servoyInstallationDir$/application_server/server/webapps/” folder (where the ROOT folder is located too) and give your archive the name “**servoy-click.war**”:



Export to Archive File

Save in: webapps

ROOT
servoy-click.war

File name: servoy-click.war
Save as type: *.*

That’s it!

3. Import the sample solution and test the "Click client"

Now launch the "servoy-admin" page and import the "**headless_client_demo.servoy**" solution - it's just the regular headless demo solution (based on the example_data.products table), with 2 buttons added to launch the JSP and the Click client.

Now for some unknown reason yet - (that might have to do with cross-context in Tomcat and the Developer special case as a Server, because if I install the click app in the Root folder it works, but then I have to modify the "web.xml" file of Servoy's ROOT context), the click version doesn't work in Developer.

But it works fine in the server, so quit Developer, and launch the server from the .bat (or .sh) file or as a service (or daemon).

The .bat (or .sh) is located in the "/$ServoyInstallDir$/application_server/" folder.

Once launched, you can use the smart client, the web client, the headless client (if you have installed the headless JSP files) and the Click client on the same form:

To navigate to the Click client, you can launch the smart client first and click on the "show web page Click" button located at the bottom right of the form – it will open the solution at the address of your Servoy server, for me it is:
http://localhost/servoy-click/

The demo Click is meant to replicate the headless client interface, except that it's not build in JSP, it's build in click.

This is why I enclosed the Eclipse project so that you can view how easy and clean it is to develop pages with the Click framework, whatever the back-end is (JDBC, ORM like Cayenne or Hibernate, Spring, or… Servoy).

Send me your comments about this one.

Patrick Talbot
Servoy Stuff
2009-06-15