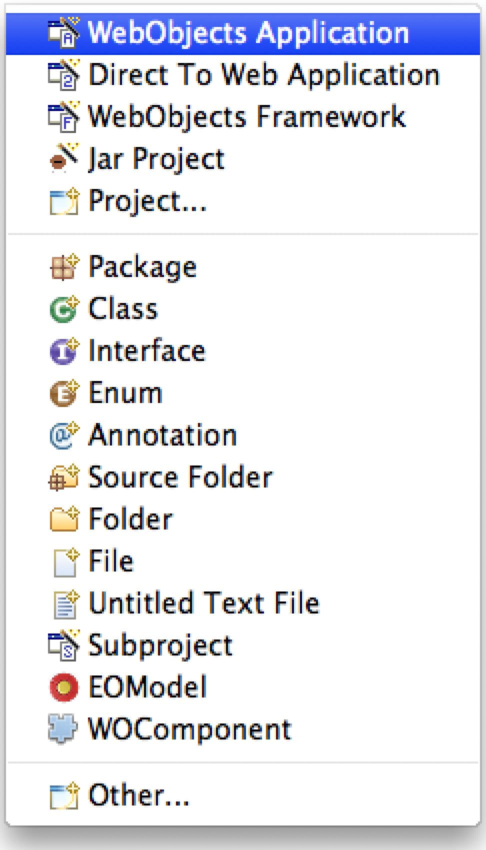
Java Client and Direct To Java Client Example Project using WebStart

This tutorial will help you create a single Eclipse/WOLips project that contains both the client- and server-side classes of a WebObjects Java Client project, build both applications and then launch them both.

You can also create a Java Client application using two separate Eclipse projects. One for the server (a WOLips project), and one for the client (most likely a plain Java project). This will make the separation between client and server much more obvious, but then you will have two Eclipse projects to maintain. The approach described in this tutorial is most appropriate for WebStart based client app distribution.

1. Create a new WebObjects Application



2. Name it "WebStartJNLPTest" in Eclipse with a package of "your.app"
   Note!
   If you change the Project name or packages, then you'll need to make additional changes in the subsequent steps.
3. Modify the Main.html by adding:

   Main.html Changes

   ```
   <WEBOBJECT NAME=JavaClientLink>click here</WEBOBJECT>
   ```

4. Modify the Main.wod file by adding:

   Main.wod Changes

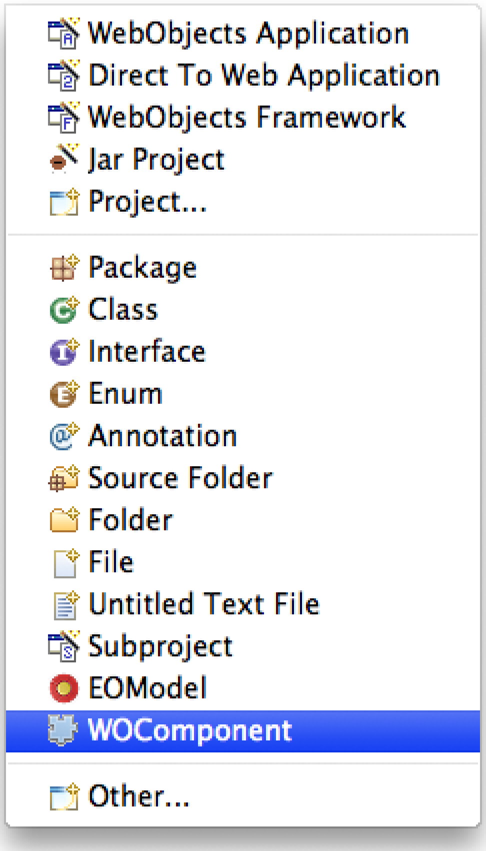
   ```
   JavaClientLink: WOHyperlink {
       href = javaClientLink;
   }
   ```

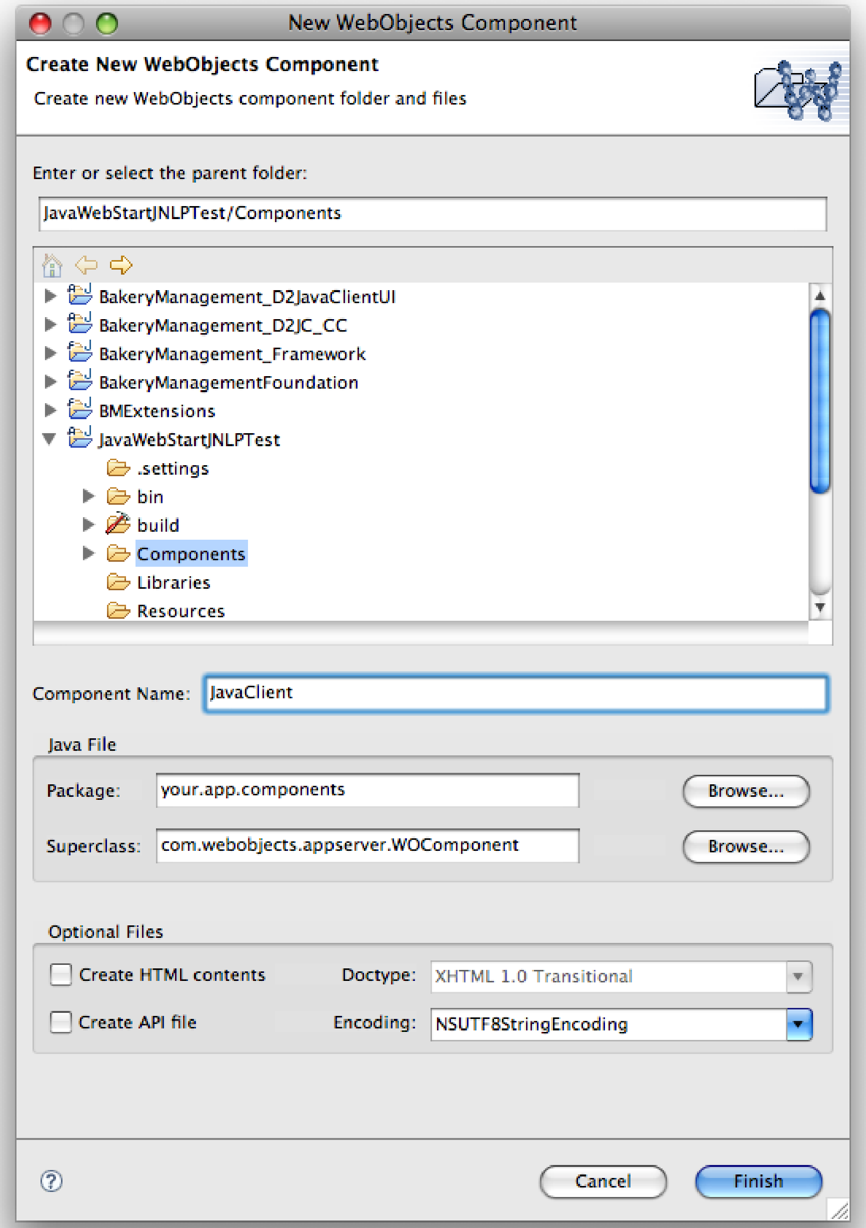
5. Modify the Main.java file by adding:

Main.java Changes

```
public String javaClientLink() {
        return WOJavaClientComponent.webStartActionURL(context(), "JavaClient");
}
```

6. Create a WOComponent named "JavaClient".




New WebObjects Component

Create New WebObjects Component

Create new WebObjects component folder and files

Enter or select the parent folder:

JavaWebStartJNLPTest/Components

- BakeryManagement_D2JavaClientUI
- BakeryManagement_D2JC_CC
- BakeryManagement_Framework
- BakeryManagementFoundation
- BMExtensions
- JavaWebStartJNLPTest
  - .settings
  - bin
  - build
  - Components
  - Libraries
  - Resources

Component Name: JavaClient

Java File

Package: your.app.components

Superclass: com.webobjects.appserver.WOComponent

Optional Files

Create HTML contents

Doctype: XHTML 1.0 Transitional

Create API file

Encoding: NSUTF8StringEncoding

This WOComponent will be what passes the specifics about your client-side application to WebObjects.

7. Modify the JavaClient.html replacing the contents with:

JavaClient.html Contents

```
<WEBOBJECT NAME=JavaClientComponent></WEBOBJECT>
```

8. Paste the following as the contents of the JavaClient.wod file. Feel free to change the applicationName, applicationDescription and vendor parameters to anything you wish.

JavaClient.wod Contents

```
JavaClientComponent: WOJavaClientComponent {
    applicationClassName = "com.webobjects.eogeneration.EODynamicApplication";
    applicationName = "D2JC WebStart Test";
    applicationDescription = "D2JC WebStart Test";
    downloadClientClasses = "noDownloadClientClasses";
    vendor = "My Company";
}
```

Note!

If you are not creating a Direct To Java Client (D2JC) application, then you will need to replace the applicationClassName parameter (com. webobjects.eogeneration.EODynamicApplication) with the name of your client-side application class. Your non-D2JC client-side Application class **must** subclass EOApplication.

9. If you didn't add an EOModel file to the project when you created it, do it now. Drop any existing EOModel in the Resources directory, or create one from scratch.
   If you don't specify a client class name for each entity, then WebObjects will automatically use EOGenericRecord as it's class. Once you want to start specifying client-side logic including default values and such, you will need to add the Client Class Name to the Entity in the EOModel, and then setup a new .eogen file in eclipse to generate the client-side classes.
10. Add the following frameworks to the project build path:
    JavaDirectToWeb
    JavaDTWGeneration
    JavaEOApplication
    JavaEOControl
    JavaEODistribution
    JavaEOGeneration
    JavaEOInterface
    JavaEOInterfaceSwing
    JavaEOProject
    JavaEORuleSystem
    JavaFoundation
    JavaJDBCAdaptor
    JavaWebObjects
    JavaWOExtensions
    Note!

    If you are not creating a D2JC application, then you won't need all of these, but they won't hurt anything either.
11. Create a ClasspathClient.txt file in the root of the project and paste the following as contents:
    Note!

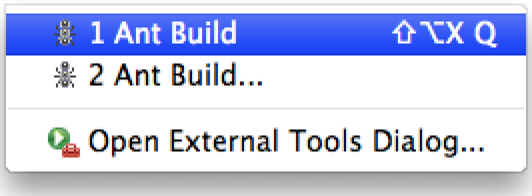
    If you changed the applicaton name in Step 2 above, make sure you change the name of the Client-Side .jar file in the first classpath line below (WebStartJNLPTest.jar). Also be sure to match the case of the project. Client-Side .jar filenames are CamelCase instead of all lowercase as server-side .jar filenames are.

ClasspathClient.txt File

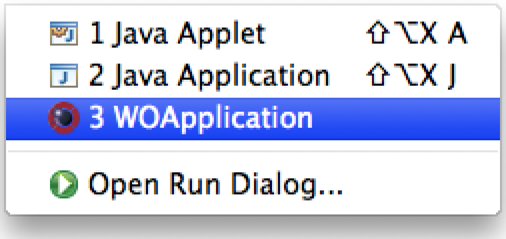
```
# ApplicationClass == com.webobjects.eoapplication.client.EOClientApplicationSupport
# JDB              == jdb
# JDBOptions       ==
# JVM              == java
# JVMOptions       ==
APPROOT/WebServerResources/Java/WebStartJNLPTest.jar
/System/Library/Frameworks/JavaFoundation.framework/WebServerResources/Java/JavaFoundation.jar
WOROOT/Library/Frameworks/JavaEOControl.framework/WebServerResources/Java/JavaEOControl.jar
WOROOT/Library/Frameworks/JavaXML.framework/WebServerResources/Java/JavaXML.jar
WOROOT/Library/Frameworks/JavaDirectToWeb.framework/WebServerResources/Java/JavaDirectToWeb.jar
WOROOT/Library/Frameworks/JavaEOApplication.framework/WebServerResources/Java/JavaEOApplication.jar
WOROOT/Library/Frameworks/JavaEODistribution.framework/WebServerResources/Java/JavaEODistribution.jar
WOROOT/Library/Frameworks/JavaEOGeneration.framework/WebServerResources/Java/JavaEOGeneration.jar
WOROOT/Library/Frameworks/JavaEOInterface.framework/WebServerResources/Java/JavaEOInterface.jar
WOROOT/Library/Frameworks/JavaEOInterfaceSwing.framework/WebServerResources/Java/JavaEOInterfaceSwing.jar
WOROOT/Library/Frameworks/JavaEORuleSystem.framework/WebServerResources/Java/JavaEORuleSystem.jar
WOROOT/Library/Frameworks/JavaWOJSPServlet.framework/WebServerResources/Java/JavaWOJSPServlet_client.jar
HOMEROOT/Library/Java
LOCALROOT/Library/Java
WOROOT/Library/Java
/Network/Library/Java
WOROOT/Library/Frameworks/JavaVM.framework/Classes/classes.jar
WOROOT/Library/Frameworks/JavaVM.framework/Classes/ui.jar
```

This file is only required by the application when it is being launched from within Eclipse.

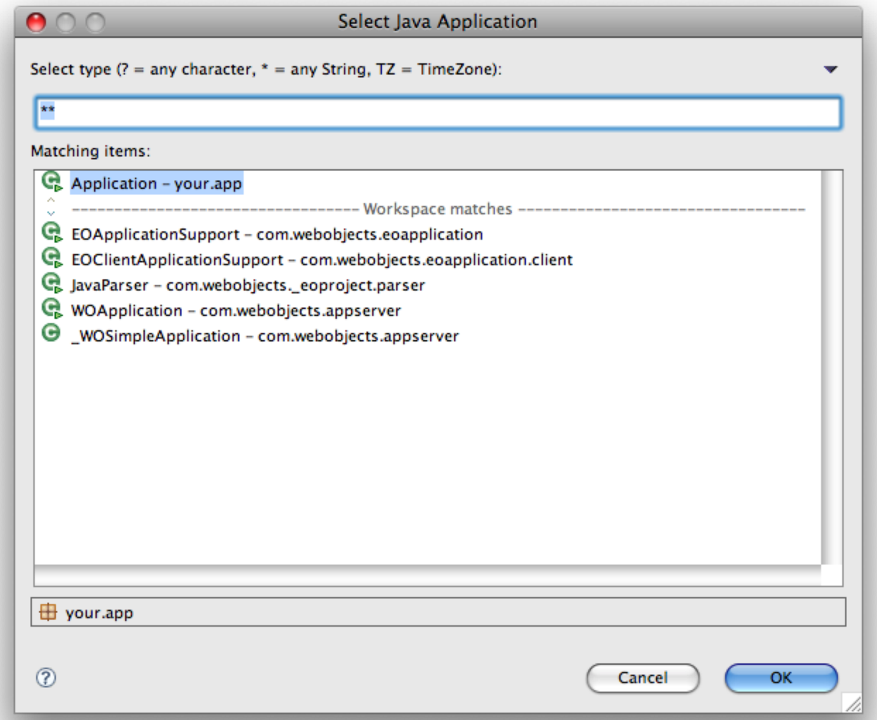
12. Create a Java Client Incremental Builder by following the instructions. This file is needed to make up for the lack of Java Client application building by the WOLips Incremental Builder. It also copies the client launch scripts into the project so that the application will auto-launch when you run it from within Eclipse.
13. If you did not setup the Java Client Incremental Builder, then right-click on the javaclientbuild.xml file and select "Run As -> Ant Build"



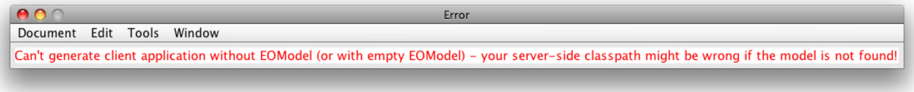
14. Right-click on the project and select "Run As -> WebObjects Application"



15. Select your.app.Application as the Application class


Select Java Application
Select type (? = any character, * = any String, TZ = TimeZone):
**
Matching items:
Application – your.app
Workspace matches
EOApplicationSupport – com.webobjects.eoapplication
EOClientApplicationSupport – com.webobjects.eoapplication.client
JavaParser – com.webobjects._eoproject.parser
WOApplication – com.webobjects.appserver
_WOSimpleApplication – com.webobjects.appserver
your.app
Cancel
OK

16. If you want to actually deploy this application, then you will also need to modify the build.xml file.
    This is it. WebStart should automatically download your client application and launch it. If you are creating a D2JC application and did not put an EOModel file in the Resources directory of your project, then you'll see something like this:



```
Can't generate client application without EOModel (or with empty EOModel) – your server-side classpath might be wrong if the model is not found!
```

    If you don't want a D2JC application, then you are on your own for creating the user interface for it. Your client-side application can be a command-line or GUI application. If you want a Swing GUI, then I **highly** recommend Florijan Stamenkovic's JBND Framework