

Riatrix4Applet Quickstart Guide

for Eclipse

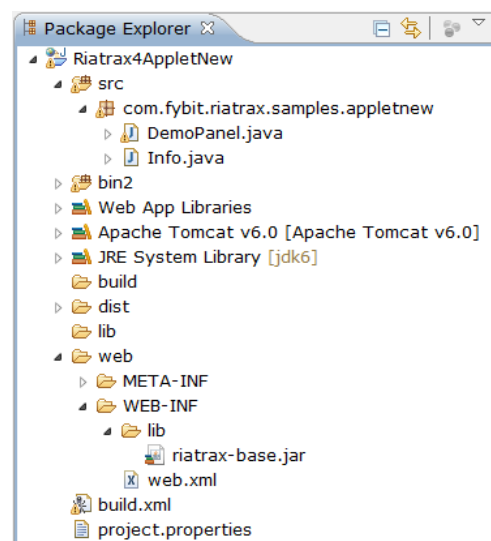
The Fybit Quickstart guide explains you how to work with Fybit to develop powerful web applications. The guide uses the sample and template application contained in *Riatrix4AppletNew.zip* for the explanation and assumes that you completed the IDE setup.

Project Layout, Compile, Launch

Project Layout

The Riatrix4AppletNew project uses a standard Java project layout. The src folder contains the Java sources. All files in the web directory are available in the web application, e.g. index.html. The web/WEB-INF/lib directory contains the Fybit JAR libraries.

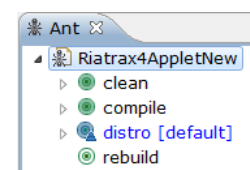
The directory bin2 is created by the Fybit engine and contains classes for remoting as well as the generated Java code. Normally, you don't have to look at or edit those files.



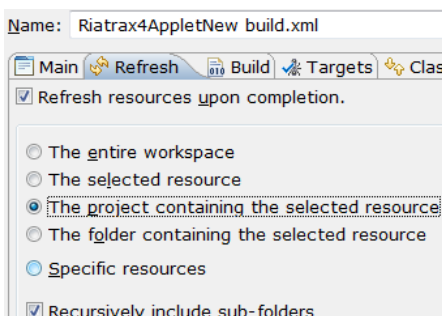
Building the application

To compile the application, you need to open the Ant view: Window → Show View → Other → Ant → Ant. Then, drag the build.xml file to the Ant view.

- clean: removes all generated files from bin2 and regenerates classes and other files
- compile: does the regular Java compile
- distro: creates a WAR file to be deployed outside the IDE



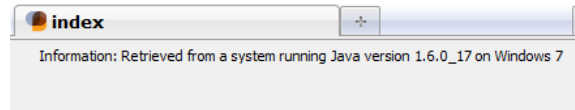
There is no need to execute ant targets inside the IDE unless you change the interface between client and server or add code to the client, in that case, you should use compile or distro. We recommend to automatically refresh the project upon building with Ant, which can be configured as follows: In the Ant view, right click on compile → Run As → External Tools Configuration → Refresh Tab. Then check “Refresh resources upon completion” and select “The project containing the selected resource”.



Launching the application

You can launch the application in the browser: Right click on the project → Run As → Run on server → Tomcat v6 → Finish.

The server starts and displays the application in the browser.



First Steps with Fybit

Defining the Program Entry Point

The JavaScript compiler needs to find the entry point to the application. This entry point is specified by the

```
@MainPanel(name="index")
public class DemoPanel extends JPanel {
```

`@MainPanel` annotation, e.g. in class `DemoPanel.java`. The `name` attribute uniquely identifies that entry point. The class must be an instance of `java.awt.Component` and will by default fill the whole browser window. You are free to use any Swing-compatible widgets, including third party libraries like JGoodies or JFreeChart.

Remote Invocations

In order to make remote invocations to the server, you can use the `@Services` annotation to denote instances that call the server:

```
@Services(implementation=MyInfo.class)
static Info info;
```

Some rules about fields annotated with `@Services`:

- The field has to be `public static` or `protected static`.
- Use the interface as type (here `MyInfo`) and specify the concrete server class with the `implementation` attribute.
- If asynchronous calls to the server are needed, use the proxy class consisting of the interface's name and postfix "Proxy", e.g. `MyInfoProxy`, and specify the interface with the `iface` attribute.