Get Involved

java-net Project
Request a Project
Project Help Wanted Ads
Publicize your Project
Submit Content

Get Informed

About java.net
Articles
Weblogs
News
Events
Also in Java Today
java.net Online Books
java.net Archives

Get Connected

java.net Forums
Wiki and Javapedia
People, Partners, and Jobs
Java User Groups
RSS Feeds

Search

Web and Projects: »
Online Books: »
Advanced Search

Project Wonderland v0.5: Working with Modules

by Jordan Slott <jslott@dev.java.net>

Purpose

In this tutorial, you will learn how to compile a module using a sample module project and the install that module into an instance of the Project Wonderland server. A sample module project is included with this tutorial.

Expected Duration: 30 minutes

Prerequisites

Before completing this tutorial, make sure you have downloaded and successfully compiled the Project Wonderland source, as described by Project Wonderland v0.5: Download, Configure, Build and Run from the Source Code. It is also a good idea to be familiar with Project Wonderland's web-based administration detailed here: Project Wonderland v0.5: Web-Based Administration.

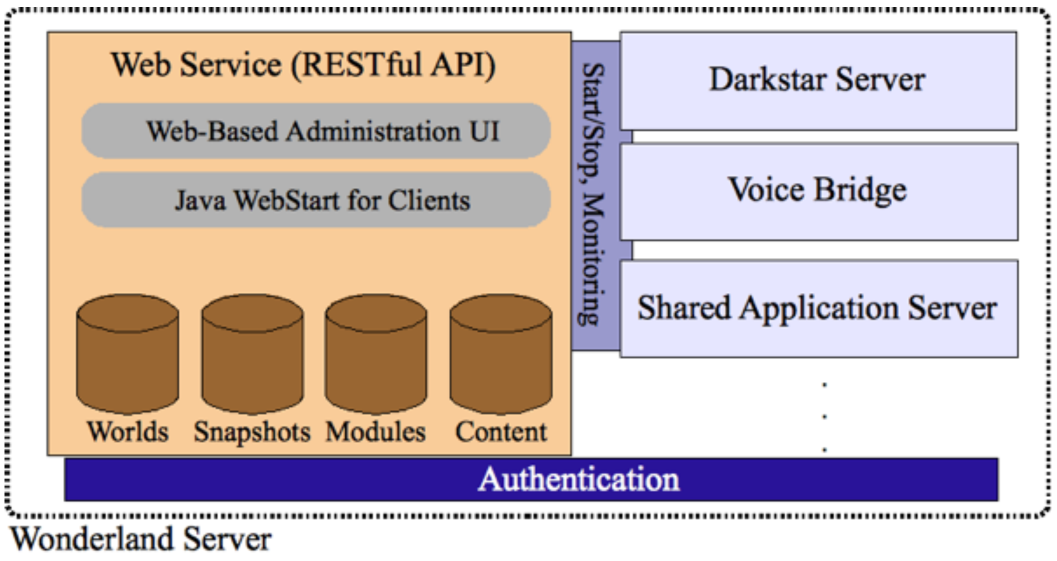
Project Wonderland Modular Architecture

Beginning in v0.5 of Project Wonderland, **modules** are the chief way developers extend the functionality of the system. The module system is fairly central to the new software architecture being introduced in this version (Figure 1). Not only can developers extend the functionality of the Project Wonderland platform with modules, much of the functionality shipped with the "core" software are actually stored in modules and downloaded by the Project Wonderland client on-demand: this modular architecture allows pieces of the platform to be updated at any time.

Project Wonderland modules consists of a collection of code, artwork and other resources. Examples of code contained within a module include new Cell types for Project Wonderland and other "services" that execute in either/both the client or server. (Examples of services include, for example, custom art loaders). Project Wonderland modules may also contain 3D models, textures and no code whatsoever. Or they may contain world definitions as a collection of XML files (called a Wonderland File System).

Modules are installed via a web-based administration user interface that you will see shortly. Once installed, the system decomposes modules into their constituent parts and distributes them to the different parts of the server and client as necessary.

Figure 1. Project Wonderland v0.5 Server Architecture (Click on the image for a full-screen version)



In previous versions of Project Wonderland, extending its functionality consisted almost entirely of writing new Cell types -- while this may be the chief activity when developing modules in v0.5, a module is a much more powerful concept -- not only may it contain code that other modules may use, but it also may contain artwork and WFSs. Developers may create new Cell types and also distribute artwork to go along with their custom Cell types. They may also assemble individual Cells into pre-configured spaces (as defined by a WFS) and include them in the module too. A module may also just contain artwork, for example, a set of re-usable useful components that developers may wish to share with others.

Artwork contained within a module is now served directly by a web server now embedded in the Project Wonderland server, there is no longer a need to setup a separate web server to serve art assets. This greatly simplifies the deployment of Project

Wonderland.

Download sample module project

First, download the sample module project that contains an example new Cell type for Project Wonderland v0.5 and the build system necessary to create the module. This module contains the complete source code for the "shape" module that is the subject of future developer tutorials.

1. Download the "shape" module: .tar.gz (Linux/UNIX/Mac OSX) or .zip (Windows)
2. Expand it into a directory of your ch0osing, for example in **$HOME/src/**, where **$HOME** is your home directory.

The **shape-tutorial-module/** directory should contain the following files and directories:

```
art      build.xml      my.module.properties   nbproject/      src/
```

The **build.xml**, **my.module.properties**, and **nbproject/** files and directory are part of the example build system. You will use this same build system for your future module development. The **src/** directory contains the code for our example: a new Cell type for Project Wonderland that displays a shape in the world.

Set the Project Wonderland.dir property

In order to compile the module project, you need to first tell it where you placed your compiled Project Wonderland source installation. The module project needs the Project Wonderland source tree; it relies upon important utilities and the compiled Project Wonderland APIs and libraries contained there.

1. Edit the **my.module.properties** in your **shape-tutorial-module/** directory.
2. Edit the **Project Wonderland.dir** property so that it points at your Project Wonderland source tree.

In the example snippet from **my.module.properties** below, the Project Wonderland source tree is located in the user's home directory under the directory named **src/Project Wonderland/**.

```
#
# Property: Project Wonderland.dir (required)
# The location of the Wonderland source
#
wonderland.dir=/Users/jordan/src/wonderland
```

Compiling the shape module project

To compile this module, in the **shape-tutorial-module/** directory:

```
% ant
```

In the course of compiling your module, the build system creates the **build/** and **dist/** directories. The entire module itself is packaged into a Java archive (JAR) file, named **shape-tutorial-module.jar**. While you do not need to know the contents of this JAR file, it is worthwhile to spend a few moments to peak inside and take a look around.

Listed below are the contents of **shape-tutorial-module.jar**. (You may obtain a similar listing by invoking the **jar tvf dist/shape-tutorial-module.jar** command).

```
     0 Tue Sep 01 13:03:54 EDT 2009 META-INF/
    94 Tue Sep 01 13:03:52 EDT 2009 META-INF/MANIFEST.MF
     0 Mon Aug 31 13:08:02 EDT 2009 art/
198029 Mon Aug 31 13:06:38 EDT 2009 art/MountainPicture.png
   215 Tue Sep 01 12:58:28 EDT 2009 module.xml
    75 Tue Sep 01 12:58:28 EDT 2009 requires.xml
     0 Tue Sep 01 13:03:54 EDT 2009 client/
 15550 Tue Sep 01 13:03:54 EDT 2009 client/shape-tutorial-module-client.jar
     0 Tue Sep 01 13:03:54 EDT 2009 server/
  5703 Tue Sep 01 12:58:28 EDT 2009 server/shape-tutorial-module-server.jar
```

The **module.xml** file describes the module: its unique name, version, and a textual description. The **requires.xml** file lists all other modules upon which this module depends. The **client/** directory contains all of the code (packaged as a JAR file) to be included in the Wonderland client, and the **server/** directory contains all of the code (packaged as a JAR file) to be included in the Wonderland server.

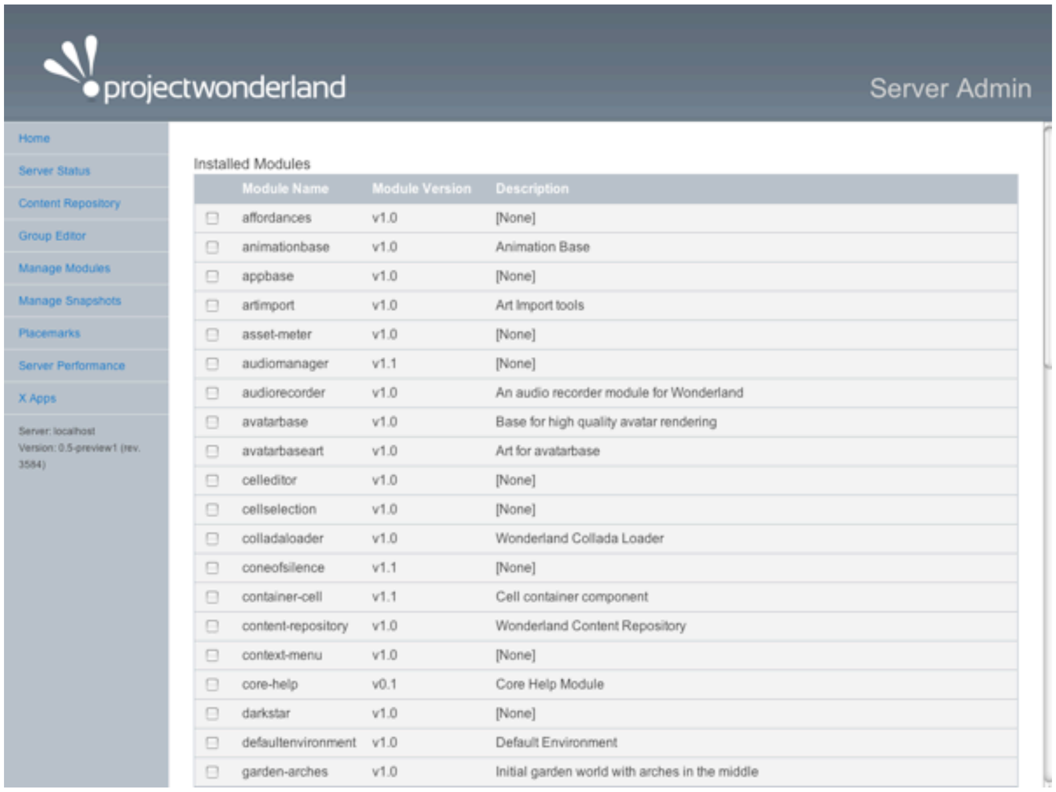
Installing the shape module in Wonderland

Version 0.5 of Project Wonderland provides a web-based interface to administer modules.

1. Run the Wonderland server by an "ant run-server" in your **wonderland/** directory.
2. In your favorite web browser, visit **http://<host>:<port>/wonderland-web-front/admin** where <host> is machine of your server and <port> is typically 8080.
3. Click on the **Manage Modules** link on the left-hand side bar.

You should see a screen that resembles Figure 2. Listed are the modules that are currently installed, and the modules that are waiting to be installed ("Pending Modules") and removed ("Removed Modules"). Modules are listed as "pending", for example, if they depend upon other modules that must be installed too. Module are listed as "removed", for example, if other installed modules still depend upon them.

Figure 2. Module management (Click on the image for a full-screen version)

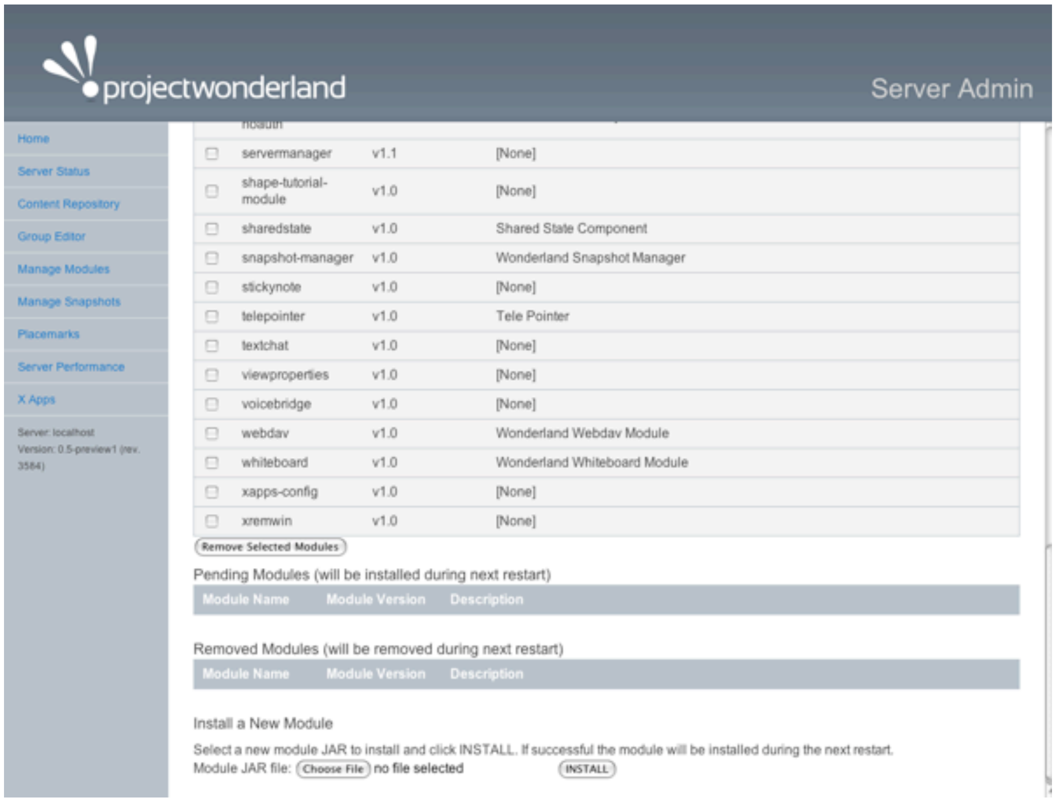


Installed Modules

	Module Name	Module Version	Description
☐	affordances	v1.0	[None]
☐	animationbase	v1.0	Animation Base
☐	appbase	v1.0	[None]
☐	artimport	v1.0	Art Import tools
☐	asset-meter	v1.0	[None]
☐	audiomanager	v1.1	[None]
☐	audiorecorder	v1.0	An audio recorder module for Wonderland
☐	avatarbase	v1.0	Base for high quality avatar rendering
☐	avatarbaseart	v1.0	Art for avatarbase
☐	celleditor	v1.0	[None]
☐	cellselection	v1.0	[None]
☐	colladaloader	v1.0	Wonderland Collada Loader
☐	coneofsilence	v1.1	[None]
☐	container-cell	v1.1	Cell container component
☐	content-repository	v1.0	Wonderland Content Repository
☐	context-menu	v1.0	[None]
☐	core-help	v0.1	Core Help Module
☐	darkstar	v1.0	[None]
☐	defaultenvironment	v1.0	Default Environment
☐	garden-arches	v1.0	Initial garden world with arches in the middle

To install a module, use the "Choose File" button to select the **dist/shape-tutorial-module.jar** file in your module project directory. Then click the INSTALL button. If successful, you should be brought to a page that indicates success. At this point, click the **Manage Modules** link on the left-hand side bar to refresh the page. You should see your module listed as "Installed" (Figure 3). Because the example module contains code that gets installed in the Wonderland server, the server components must be restarted. This server restart is done automatically when you install the module.

Figure 3. Shape module is installed (Click on the image for a full-screen version)



	noauth		
☐	servermanager	v1.1	[None]
☐	shape-tutorial-module	v1.0	[None]
☐	sharedstate	v1.0	Shared State Component
☐	snapshot-manager	v1.0	Wonderland Snapshot Manager
☐	stickynote	v1.0	[None]
☐	telepointer	v1.0	Tele Pointer
☐	textchat	v1.0	[None]
☐	viewproperties	v1.0	[None]
☐	voicebridge	v1.0	[None]
☐	webdav	v1.0	Wonderland Webdav Module
☐	whiteboard	v1.0	Wonderland Whiteboard Module
☐	xapps-config	v1.0	[None]
☐	xremwin	v1.0	[None]

Pending Modules (will be installed during next restart)

Module Name	Module Version	Description

Removed Modules (will be removed during next restart)

Module Name	Module Version	Description

Install a New Module

Select a new module JAR to install and click INSTALL. If successful the module will be installed during the next restart.

Module JAR file: Choose File no file selected INSTALL

In addition to the Web-based Administration UI, you may also install the module by executing **ant deploy** in your **shape-tutorial-module/** directory.

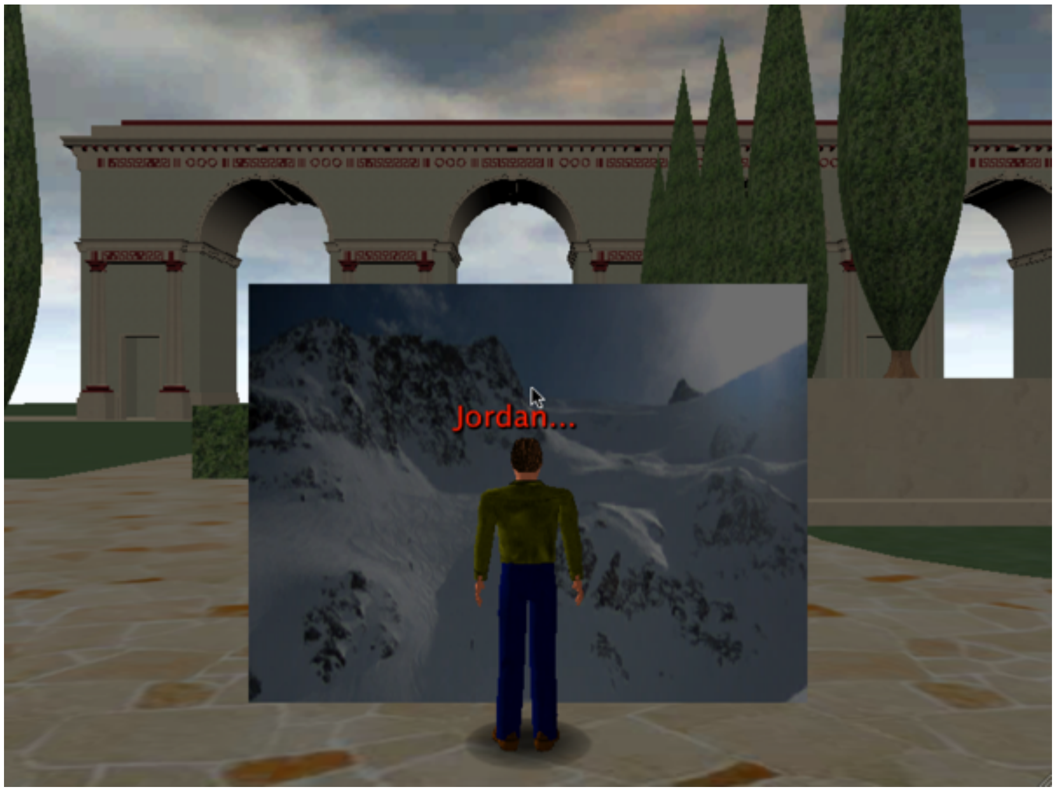
Starting the Wonderland Client

To run the client, change directories to **wonderland/core** and do an "ant run" (or you may start your client via Java Web Start). You should see the initial, default world after your client starts up. You may then use the Object Palette to create a new instance of the example Cell in your world.

1. Enter **ant run** to start your client.
2. Select Insert -> Object from the main menu.
3. Select **Shape Tutorial** from the list of Cells in the palette.
4. Click Create.

You should see a small box textured with mountain scenery. Your client window should appear as follows:

Figure 4. Shape module created in-world (Click on the image for a full-screen version)



Next Steps

Now that you have learned how to compile and install a new module into a Wonderland server and create a Cell in-world using the Object palette, you can learn how to develop your own custom modules at Project Wonderland v0.5: Developing a New Cell.

Topic **ProjectWonderlandWorkingWithModules05** . { Edit | Ref-By | Printable | Diffs r1 | More }