



Wowza® IDE 2

User's Guide

Wowza IDE 2: User's Guide



Copyright © 2006 - 2010 Wowza Media Systems. All rights reserved.

Third-Party Information

This document contains links to third-party websites that are not under the control of Wowza Media Systems, Inc. (“Wowza”) and Wowza is not responsible for the content on any linked site. If you access a third-party website mentioned in this document, then you do so at your own risk. Wowza provides these links only as a convenience, and the inclusion of any link does not imply that Wowza endorses or accepts any responsibility for the content on third-party sites.

This document refers to third party software that is not licensed, sold, distributed or otherwise endorsed by Wowza. Please ensure that any and all use of Wowza software and third party software is properly licensed.

Trademarks

Wowza, Wowza Media Systems, Wowza Media Server and related logos are trademarks of Wowza Media Systems, Inc., and may be registered in the United States or in other jurisdictions including internationally.

Adobe and Flash are registered trademarks of Adobe Systems Incorporated, and may be registered in the United States or in other jurisdictions including internationally.

Silverlight is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

QuickTime, iPhone and iPod are either registered trademarks or trademarks of Apple, Inc. in the United States and/or other countries.

Other product names, logos, designs, titles, words, or phrases mentioned may be trademarks, service marks or trade names of other entities and may be registered in certain jurisdictions including internationally.

Third Party Copyright Notices

Eclipse 3.5 and JDT: Copyright 2000, 2006 Eclipse contributors and others. All rights reserved.

Table of Contents

Introduction	5
Before Installation	5
Installing the Wowza IDE	6
Extending Wowza Media Server Using Java	7
Project and Module Class Creation	7
Import ModuleServerSide Example	10
Application.xml Configuration	10
Server Startup	11
Module Testing	12
Server Extension Wizards	13

Introduction

What is the Wowza IDE?

Wowza IDE is an integrated development environment for creating extensions, configuring and managing the Wowza Media Systems product line. It is an application built on top of the popular Eclipse application framework from the Eclipse Foundation. You can learn more about Eclipse at <http://www.eclipse.org>. The Wowza IDE is available for the Windows and Mac OS X platforms.

This version of the Wowza IDE includes the Eclipse version 3.5.1 Java development tools as well as custom IDE features to enhance the development of Wowza Media Server modules and server extensions. The remainder of this document covers the process for developing Java server extensions.

Before Installation

To run the Wowza IDE and build Wowza Pro extensions, a Java Development Kit (JDK) version 6 (aka 1.6) or greater is required. The Mac OS X operating system ships with all the tools needed to run the Wowza IDE. On the Windows platform, installation of the JDK is required. To develop Wowza Media Server modules and server extensions, installation of Wowza Server is required.

Note

The support section of the Wowza Media Systems website contains additional information and links to help with obtaining the correct Java environment and tools for your platform. You can visit this site at: <http://www.wowzamedia.com>.

Installing the Wowza IDE

The Wowza IDE is installed using an installer. Follow the instructions below for your platform.

Windows

To install the Wowza IDE on Windows, double-click the installer file and follow the instructions on the screen.

To uninstall, choose **Uninstall Wowza IDE 2** from the **Start>Programs>Wowza IDE 2** menu.

Mac OS X

To install the Wowza IDE on Mac OS X, mount the disk image (double-click .dmg) file, double-click the installer package (.pkg) file and follow the instructions on the screen. Files will be installed to the following locations.

```
/Applications/Wowza IDE 2.0.0
```

To uninstall, throw the following folders and files into the trash.

```
folder:      /Applications/Wowza IDE 2.0.0
```

Extending Wowza Media Server Using Java

How do I extend Wowza Media Server module using the Wowza IDE?

This chapter covers the process of creating a Wowza Media Server module using the Wowza IDE. The document does not include details about the Wowza Media Server API itself.

Please consult the **Wowza Media Server 2 User's Guide**, **Wowza Media Server 2 Server Side API** and **Flash Media Server to Wowza Server 2 API Mapping** documents that ship with the Wowza Media Server 2 for API details. This document serves as an extension, not as a replacement to the Eclipse **Workbench User Guide** and **Java Development User Guide**. The full Eclipse documentation can be found at <http://help.eclipse.org> or by selecting the **Help Contents** menu item in the **Help** menu of the Wowza IDE.

This chapter covers the process for coding, building and deploying a custom server side module for Wowza Media Server 2. A server side module is a Java Archive (jar) file that encapsulates a set of functionality that is dynamically linked into the server at runtime. All modules must be compiled and built using a Java Development Kit (JDK) version 6 (aka 1.6) or greater. See the **Extending Wowza Server Using Java** chapter of the **Wowza Media Server 2 User's Guide** for more details.

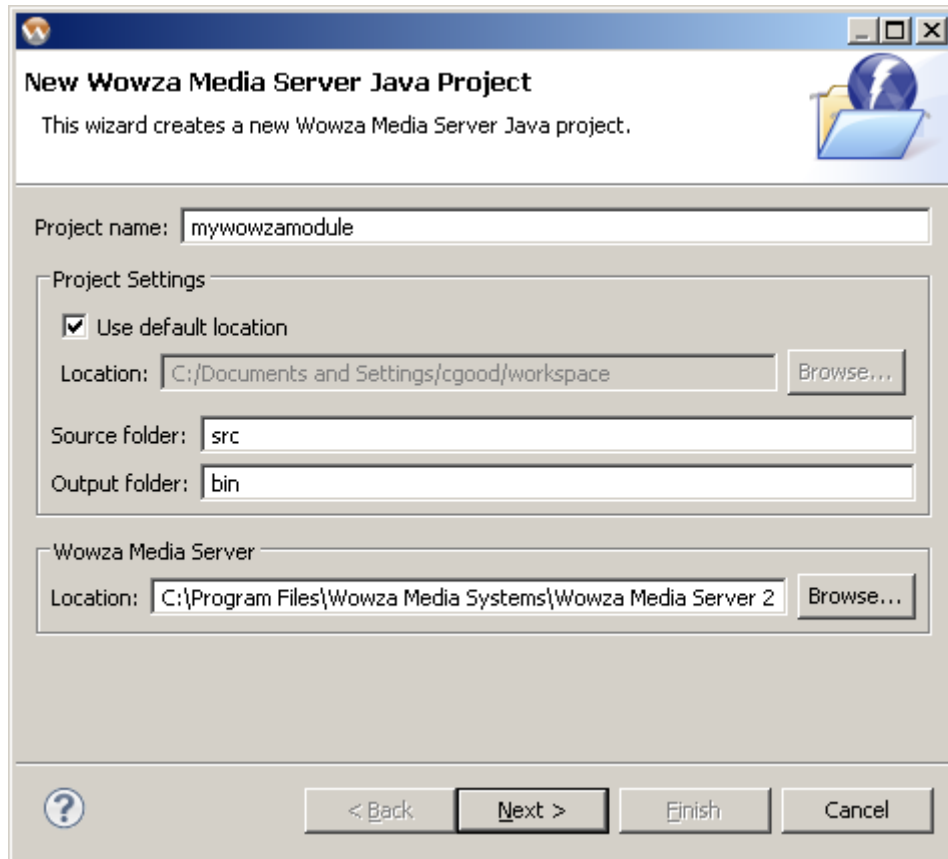
Note

It is important that you use the same version of the Java environment in your development environment as well as in the production environment. Your custom modules will not function properly in your production environment if the version of Java is less than the development environment.

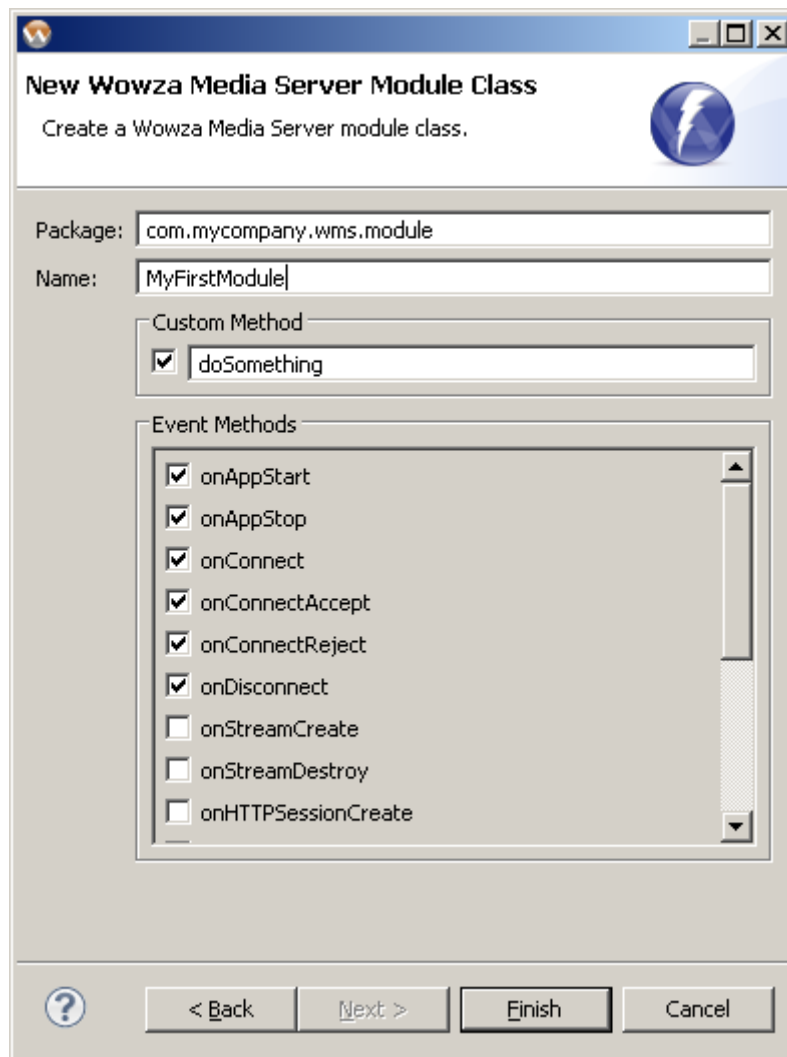
Project and Module Class Creation

Let's jump right in and launch the Wowza IDE. The first time the Wowza IDE is started, it will ask you to select a location for the workspace folder. This folder will be used to store all project related files and folders.

To invoke the new project wizard, select **New>Wowza Media Server Project...** from the **File** menu. You will see the following dialog box:



Enter a project name for your project. For this example, we will use the name **mywowzmodules**. This name will also serve as the name of your .jar file that is deployed in the Wowza Server **[install-dir]/lib** folder. Keep the default **Project Settings**. The **Wowza Media Server /Location** field should contain the full path to Wowza Media Server installation folder. Click the **Next >** button to proceed to the next step. You will see the following dialog:



This dialog is used to configure a new module class. We will be choosing a package name, class name and configuring the methods for the new module class. Java stores classes in packages. These packages are organized on the file system as folders. The convention for package naming is very similar to the processes for choosing a domain name for your company. If the web site address for your company is **www.mycompany.com**, then your company's Java code might be organized in a package that starts with **com.mycompany**. Since you are creating code for Wowza Media Server you might choose to organize this code in a sub-package named **wms**. Next, you might want to organize all your module code in a sub-package named **module**. So the full package name for your modules would be **com.mycompany.wms.module**. For this example we will be using the package name **com.mycompany.wms.module**. Enter this package name into the **Package** field.

Next, enter a class name for your module class. For this example we will use the class name **MyFirstModule**. Enter this module name into the **Name** field.

The bottom section of this dialog is used to configure the methods and method events that this module will expose. The **Custom Method** section is used to create a single custom method that is directly callable from your client side code (`NetConnection.call("doSomething", null);`). Once the class file is created, you can easily add more custom methods by creating more methods that follow this same call signature. This process is discussed in more detail in the **Extending Wowza Server Using Java** chapter of the **Wowza Media Server 2 User's Guide**.

The **Event Methods** section is used to create method events that will be called during server event processing. This process is also described in the **Extending Wowza Server Using Java** chapter of the **Wowza Media Server 2 User's Guide**. For this example, keep the default settings for the **Custom Method** and Event Methods sections. Next, click the **Finish** button.

The Wowza IDE will create the Wowza Media Server module project, create the module class, create a run command and compile and bind the module class into a jar file. The jar file will automatically be deployed to the Wowza Media Server **[install-dir]/lib** folder.

Import ModuleServerSide Example

Next, we are going to import another module class into our project. In the **Package Explorer** right-click on the **com.mycompany.wms.module** package in the **src** folder and select **Import...** In the import dialog, open the **General** folder, select the **File System** item and click the **Next >** button. Click the **Browse...** button and browse to the **[install-dir]/examples/ServerSideModules/server** folder that is part of the Wowza Server installation. Put a check mark in front of the file **ModuleServerSide.java** and click the **Finish** button. We have just imported a new module class into our project. Double click on the newly imported class in the **Package Explorer** to view it.

Application.xml Configuration

Now that we have the Wowza IDE building our .jar file, we need to instruct Wowza Server to load the new modules. We are now going to leave the Wowza IDE and configure an **Application.xml** configuration file to invoke our new module.

1. Create a new folder in the Wowza Media Server **[install-dir]/applications** folder named **mymodules**.
2. Create a new folder in Wowza Media Server **[install-dir]/conf** folder named **mymodules**.
3. Copy **[install-dir]/conf/Application.xml** into the newly created conf folder.

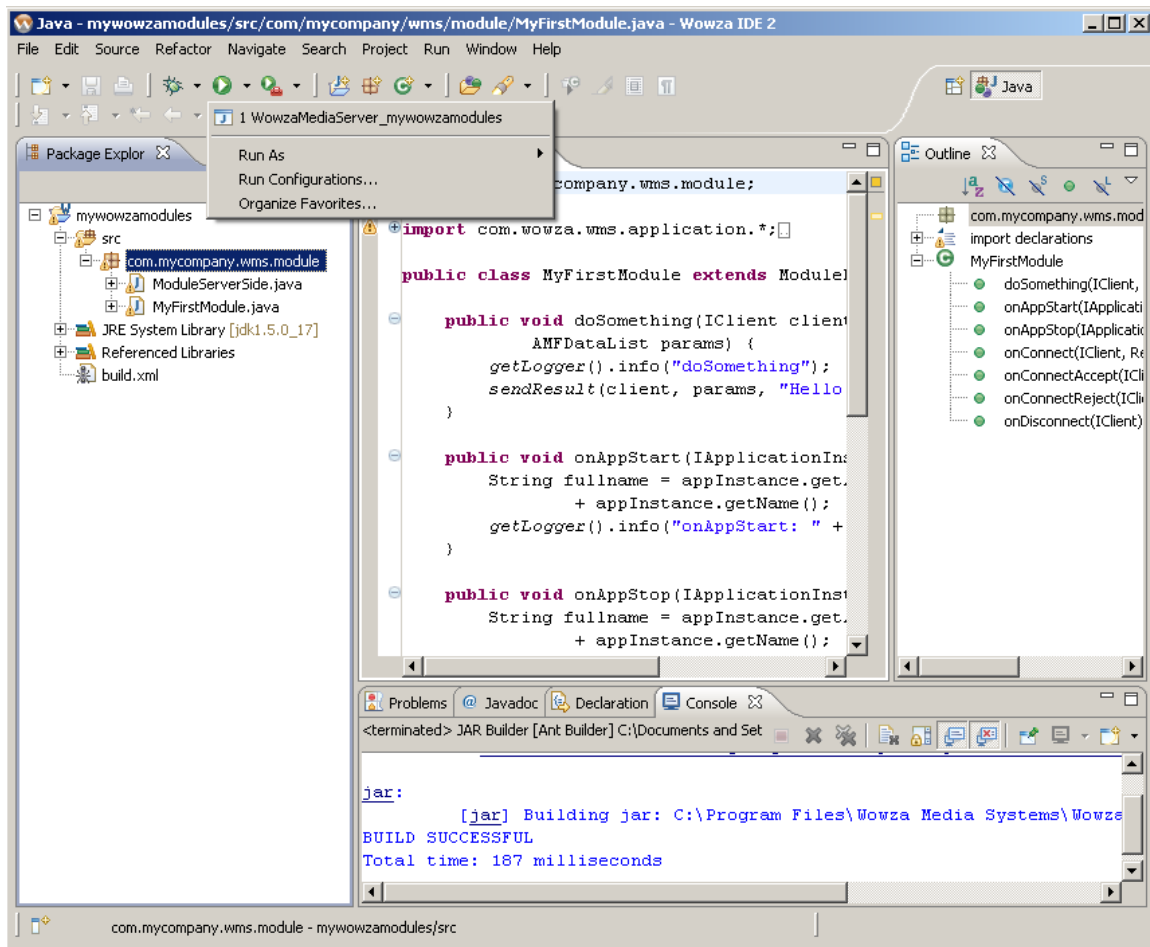
4. Edit the newly copied **Application.xml** file and add the following module definitions to the end of the <Modules> section:

```
<Module>
  <Name>MyFirstModule</Name>
  <Description>MyCompany MyFirstModule</Description>
  <Class>com.mycompany.wms.module.MyFirstModule</Class>
</Module>
<Module>
  <Name>ModuleServerSide</Name>
  <Description>MyCompany ModuleServerSide</Description>
  <Class>com.mycompany.wms.module.ModuleServerSide</Class>
</Module>
```

The new module classes are now available to the **mymodules** application.

Server Startup

We can now run Wowza Media Server from within the Wowza IDE. If you already have Wowza Server running either as a service or standalone, stop it now. To run Wowza Server from within the Wowza IDE, select **WowzaMediaServer_mywowzamodules** from the run menu in the toolbar. You can also run Wowza Server in debug mode, by selecting the same menu item from the debug toolbar menu. This will start Wowza Server and all console log output will go to the **Console** tab window at the bottom of the Wowza IDE. You can terminate the server by selecting the red **Terminate** icon in the console window. Consult the Eclipse IDE documentation for the detailed instructions on using the integrated Java debugger.



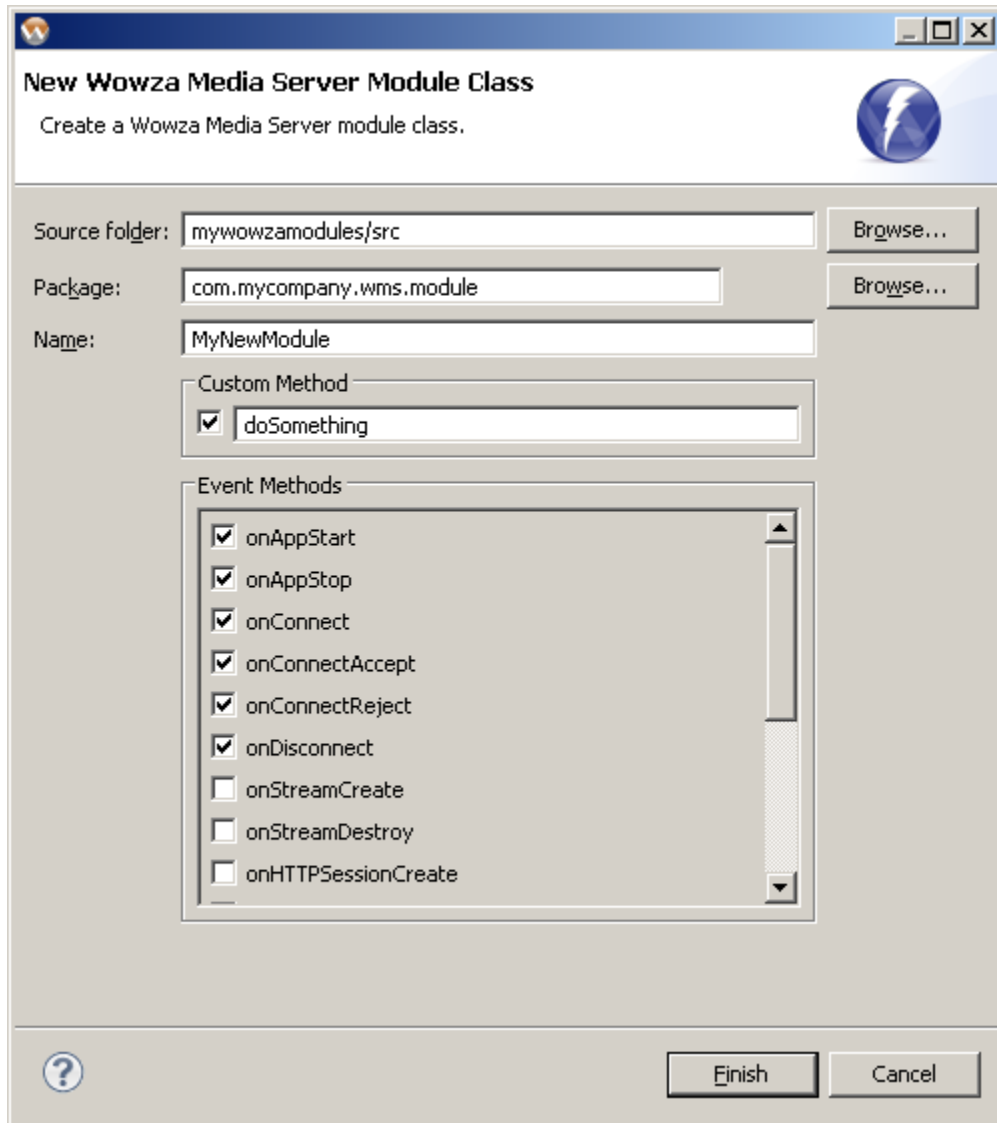
Module Testing

Finally, we get to test the new modules. Launch the Flash editor and open the file `[install-dir]/examples/ServerSideModules/client/MyFirstModule.fla`. From the **Control** menu select **Test Movie**. This will connect to Wowza Media Server using the application name **mymodules**. It will load the **MyFirstModule** class and invoke the **doSomething** method. You should see the trace window open in the Flash editor and display the text **doSomething: Hello World**.

You can also test the second module that we imported into the project. This module contains some great examples of how to send rich data between the Flash client and the server. Take some time to study both the server and client side code to get a better understanding of how this works. This module can be exercised using the `[install-dir]/examples/ServerSideModules/client/ServerSideModules.fla` Flash file.

Server Extension Wizards

The Wowza IDE includes three easy to use wizards for creating module classes, HTTPProviders and ServerListeners. For this example we will use the **Wowza Media Server Module Class** wizard to add an additional module class to the project. First, in the **Package Explorer** right-click the **com.mycompany.wms.module** package in the **src** folder and select **New>Wowza Media Server Module Class**. You will see the following dialog box:



Enter a new name for your class and configure the **Custom Method** and **Event Methods** sections and click the **Finish** button. This new module class will be added to your project's jar file and can be made available to a Wowza Media Server application by adding a new **<Module>** definition to the application's **Application.xml** file.

To add an HTTPProvider, in the **Package Explorer** right-click the **com.mycompany.wms.module** package in the **src** folder and select **New>Wowza Media Server HTTPProvider Class**. Enter a new name for your class and click the **Finish** button. To add a reference to your HTTPProvider, edit **[install-dir]/conf/VHost.xml** and add a new **<HTTPProvider>** entry to the **<HTTPProviders>** list just before the entry for **HTTPServerVersion**. For example, if you class name is **MyHTTPProvider** the **<HTTPProvider>** entry should look like this:

```
<HTTPProvider>
  <BaseClass>com.mycompany.wms.module.MyHTTPProvider</BaseClass>
  <RequestFilters>*myhttpprovider</RequestFilters>
  <AuthenticationMethod>none</AuthenticationMethod>
</HTTPProvider>
```

To invoke your HTTPProvider, start Wowza Media Server, open a web browser and enter the URL: **http://[wowza-ip-address]:1935/myhttpprovider**.

To add a ServerListener, in the **Package Explorer** right-click the **com.mycompany.wms.module** package in the **src** folder and select **New>Wowza Media Server ServerListener Class**. Enter a new name for your class and click the **Finish** button. To add a reference to your ServerListener, edit **[install-dir]/conf/Server.xml** and add a new **<ServerListener>** entry to the **<ServerListeners>** list. For example, if you class name is **MyServerListener** the **<ServerListener>** entry should look like this:

```
<ServerListener>
  <BaseClass>com.mycompany.wms.module.MyServerListener</BaseClass>
</ServerListener>
```

From here on out adding functionality to your module is a snap. The Wowza IDE provides a rich set a capabilities like: syntax highlighting, code completion, code templates, live code parsing/compiling, error highlighting and integrated documentation (Javadocs). These capabilities will speed the module development process. To get more information on the Wowza IDE's features and functionality, select the **Help Contents** link in the **Help** menu. Consult the **Extending Wowza Server Using Java** chapter of the **Wowza Media Server 2 User's Guide** for more detailed information on module creation specifics and the server side API.