



THE  
POWER  
TO KNOW.

# **SAS<sup>®</sup> AppDev Studio<sup>™</sup> 4.4**

## **Eclipse Plug-ins**

### **User's Guide**

The correct bibliographic citation for this manual is as follows: SAS Institute Inc 2013. *SAS® AppDev Studio™ 4.4 Eclipse Plug-ins: User's Guide*. Cary, NC: SAS Institute Inc.

**SAS® AppDev Studio™ 4.4 Eclipse Plug-ins: User's Guide**

Copyright © 2013, SAS Institute Inc., Cary, NC, USA

All rights reserved. Produced in the United States of America.

**For a hardcopy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

**U.S. Government License Rights; Restricted Rights:** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227–19 Commercial Computer Software–Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

Electronic book 1, December 2013

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/publishing](http://support.sas.com/publishing) or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.



# Gain Greater Insight into Your SAS® Software with SAS Books.

Discover all that you need on your journey to knowledge and empowerment.

 [support.sas.com/bookstore](http://support.sas.com/bookstore)  
for additional books and resources.

  
THE POWER TO KNOW.

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies. © 2013 SAS Institute Inc. All rights reserved. S107969US.0613



---

# Contents

<i>What's New</i> .....	vii
<b>Chapter 1 • Installing AppDev Studio</b> .....	<b>1</b>
Installation Prerequisites .....	1
Installation and Post-Installation of SAS AppDev Studio .....	2
Accessibility Features of AppDev Studio .....	4
<b>Chapter 2 • Overview of AppDev Studio 4.4</b> .....	<b>5</b>
The SAS AppDev Studio 4.4 Eclipse Plug-ins .....	5
New Features .....	5
Migrating Applications to AppDev Studio 4.4 .....	6
<b>Chapter 3 • Projects, Profiles, and Templates</b> .....	<b>7</b>
SAS Web Application Projects .....	7
Minimize the Number of Open Projects .....	8
Server Profiles .....	8
Templates .....	9
Template Descriptions .....	10
<b>Chapter 4 • Walk-Through for Web Infrastructure Platform Templates</b> .....	<b>13</b>
Introduction .....	13
Part I: Create the Project and the Application Metadata .....	14
Part II: Add the JDBC TableView Template .....	18
Part III: Add a Welcome Page and Run the Application .....	21
Add a ReportViewer Servlet Template to the Project .....	23
Add a SAS Stored Process Servlet Template to the Project .....	24
Input and Output Parameters .....	28
<b>Chapter 5 • Walk-Through for Data-Driven Project Creation</b> .....	<b>31</b>
Create an Information Map Fixed Portlet from Data .....	31
The Portlet Editor .....	36
<b>Chapter 6 • Template and Testing Details</b> .....	<b>37</b>
Files Added by the Metadata Creation Template .....	37
Copying the Application Metadata .....	39
Files Added by the Stored Process Java Client Template .....	39
Files Added by the Stored Process Servlet Template .....	41
Tomcat Configuration Details .....	42
Deployment and Authentication .....	43
<b>Chapter 7 • Exporting Projects</b> .....	<b>45</b>
Exporting Java and SAS Java Projects as a Set of JAR Files .....	45
Exporting a SAS Web Application Project as a WAR File .....	46
Exporting a Project Using a Deployment Descriptor File .....	47
<b>Chapter 8 • Managing the JAR Files in a Project</b> .....	<b>49</b>
The SAS Repository .....	49
Opening the SAS Repository Properties Editor .....	50
Identifying Dependent JAR Files .....	51

Removing JAR Files from the Classpath .....	52
Changing the Order of JAR Files in the Classpath .....	53
Adding New JAR Files to the Classpath .....	54
Adding Dependent JAR Files .....	55
Specifying the Current Versions of JAR Files .....	55
Specifying Other JAR File Versions .....	55
Removing Version Restrictions .....	56
Reporting the Classpath JAR Files .....	56
Reporting JAR File Relationships .....	57
Finding a Class in a JAR File .....	58
Changing Default Classes for SAS Java Projects .....	59
<b>Chapter 9 • Using the SAS Editor Extensions .....</b>	<b>61</b>
Introduction to SAS Editor Extensions .....	61
Accessing SAS Component API Documentation .....	62
Adding Missing Import Statements .....	63
Attaching a SAS Model to a Viewer .....	64
SAS Snippets .....	67
<b>Appendix 1 • Creating a SAS Web Application That Does Not Use the Web     Infrastructure Platform .....</b>	<b>71</b>
<b>Index .....</b>	<b>77</b>

# What's New

---

## About This Book

This book has been updated to cover the first maintenance release for AppDev Studio 4.4. Significant differences between the initial release of AppDev Studio 4.4 and the first maintenance release for AppDev Studio 4.4 are noted.





## Chapter 1

# Installing AppDev Studio

---

<b>Installation Prerequisites</b> .....	<b>1</b>
Supported Versions of SAS Software .....	1
Install Eclipse 4.2.2 .....	1
Java Platform Requirements .....	2
<b>Installation and Post-Installation of SAS AppDev Studio</b> .....	<b>2</b>
Installation Instructions .....	2
Post-Installation Configuration .....	3
Eclipse Memory Settings .....	3
<b>Accessibility Features of AppDev Studio</b> .....	<b>4</b>

---

## Installation Prerequisites

### *Supported Versions of SAS Software*

To work correctly, a SAS AppDev Studio 4.4 project must match the maintenance level of the SAS 9.4 installation on which it is run. You need to apply a maintenance update to SAS AppDev Studio only if you are developing SAS projects for SAS 9.4 installations that have also had maintenance applied. The first maintenance release for SAS AppDev Studio 4.4 requires the first maintenance release for SAS 9.4.

### *Install Eclipse 4.2.2*

You must have Eclipse 4.2.2 installed before installing AppDev Studio 4.4. The AppDev Studio installation updates the Eclipse environment, and without a compatible version of Eclipse installed, you cannot install AppDev Studio.

Install Eclipse by following these steps:

1. Create a working directory for Eclipse 4.2.2 (for example, `C:\Eclipse422`).
2. Download Eclipse 4.2.2. The supported release of Eclipse can be found on the SAS Third-party Downloads page located at <http://support.sas.com/resources/thirdpartysupport/v94/other.sw.html#eclipse>.
3. Extract the Eclipse archive to `C:\Eclipse422`. You should now have an eclipse directory inside your working directory (for example, `C:\Eclipse422\eclipse`).

## Java Platform Requirements

The SAS AppDev Studio 4.4 development bundle requires Java 2 Standard Edition (J2SE) 1.7.0\_15 or higher for application and web application development, and execution of applications at run time. Note that AppDev Studio 4.4 has only been tested with the Oracle JRE.

For supported web browsers and application servers, see the Third-Party Software for SAS 9.4 information at <http://support.sas.com/resources/thirdpartysupport/v94/index.html>.

Although the Eclipse IDE for Java EE Developers version 4.2.2 requires Java 6 at a minimum, SAS AppDev Studio 4.4 was tested with Java 7, and that version is recommended. See also <http://www.eclipse.org/downloads/moreinfo/jre.php>.

---

# Installation and Post-Installation of SAS AppDev Studio

## Installation Instructions

The SAS Deployment Wizard provides two ways to install AppDev Studio.

The first is to install all the software that you need for development on a single machine. This plan installs and runs all three tiers on the same machine. This includes a scaled-down Business Intelligence (BI) server, a middle tier, and the AppDev Studio Eclipse Plug-ins. This type of installation is needed when the necessary servers, such as an Enterprise Business Intelligence (EBI) environment, are not available elsewhere. Running all three tiers on one machine is computationally intensive, so plan accordingly. With this plan you can also interact with existing remote tiers, which AppDev Studio makes easy using profiles. To select this type of installation, use the installation plan **AppDev Studio, one machine**.

The second way is to install only the AppDev Studio Eclipse Plug-ins. This installation assumes that the necessary servers already exist. To select this type of installation, use the plan **AppDev Studio, three machine**, and install only the AppDev Studio Eclipse Plug-ins client.

Follow these steps to begin a guided installation of AppDev Studio:

1. Start the SAS Deployment Wizard, select **Install SAS Software**, and click **Next**.
2. Select **Perform a Planned Deployment** and **Install SAS Software**.
  - a. If you are installing all three tiers using the one machine plan, also select **Configure SAS Software**.
  - b. If you are installing only the AppDev Studio Eclipse Plug-ins using the three machine plan, clear the **Configure SAS Software** check box.
3. Click **Next**.
4. Follow the on-screen instructions for installing the software.

## Post-Installation Configuration

Before you start AppDev Studio, you must connect it to an Eclipse installation and then configure it.

1. Connect AppDev Studio to an Eclipse installation by following these steps:
  - a. Start the SAS AppDev Studio Eclipse Configuration Tool (**Start** ⇒ **All Programs** ⇒ **SAS** ⇒ **SAS AppDev Studio 4.4 Eclipse Configuration Tool**).
  - b. In the Configuration Tool, select **Eclipse** ⇒ **Search** and specify the top level of an Eclipse installation that you want to connect to AppDev Studio.  
  
The search is recursive, and a search of **C:\Eclipse422\** will find the Eclipse installed at **C:\Eclipse422\eclipse**. If you have multiple Eclipse installations under one directory, point to the containing directory to find all the Eclipse installations under that directory.
  - c. When the search is complete, select from the main Configuration Tool window the Eclipse installation that you want to connect to AppDev Studio, and then select **Eclipse** ⇒ **Connect**. The connection process modifies the Eclipse installation to run AppDev Studio and can take several minutes.
  - d. Exit the AppDev Studio Eclipse Configuration Tool.
2. Perform the New Workspace setup for each Eclipse workspace that you want to use.

When you initially launch AppDev Studio within an empty Eclipse workspace, the New Workspace Setup cheat sheet starts and guides you through the following processes:

- creating compatible Java runtimes
- specifying Eclipse compiler options
- setting the correct server run time for web application development
- creating a BI Server Profile
- creating a connection profile for the BI server
- configuring a Tomcat server for testing

To start this cheat sheet, launch the Eclipse attached to AppDev Studio. If the New Workspace Setup cheat sheet is not automatically displayed, select **Help** ⇒ **Cheat Sheets**, and then expand **SAS AppDev Studio** and choose **New Workspace Setup**.

## Eclipse Memory Settings

Because of a memory intensive Java EE task in the Eclipse Web Tools Platform, if you like to have several SAS Web Application projects open in your workspace, you should specify the maximum heap size to be at least 768 MB. On rare occasions this task can be triggered simultaneously on multiple worker threads in Eclipse. When this happens, Eclipse can run out of memory if the heap is not large enough, causing Eclipse to become unstable.

To change the maximum heap size, modify the **-Xmx** setting in the **eclipse.ini** file for the Eclipse installation to which SAS AppDev Studio is connected. For example:

```
-Xmx768m
```

**See Also**

“Minimize the Number of Open Projects” on page 8

---

## Accessibility Features of AppDev Studio

SAS AppDev Studio Eclipse 4.4 Plug-ins includes accessibility and compatibility features that improve the usability of the product for users with disabilities. These features are related to accessibility standards for electronic information technology that were adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973, as amended. However, portions of the AppDev Studio interface are implemented using Java controls, and do not always comply with Section 508 guidelines.

Notable exceptions include:

- In some cases, screen-reading technology is unable to read text, read field labels in the correct order, or read only the text that is currently visible.
- The color of the text in the Help window of the AppDev Studio Configuration Tool cannot be changed.
- The portlet editor does not scroll vertically via keyboard controls.

If you have questions or concerns about the accessibility of SAS products, send e-mail to [accessibility@sas.com](mailto:accessibility@sas.com).

## Chapter 2

# Overview of AppDev Studio 4.4

---

<b>The SAS AppDev Studio 4.4 Eclipse Plug-ins</b> .....	<b>5</b>
<b>New Features</b> .....	<b>5</b>
<b>Migrating Applications to AppDev Studio 4.4</b> .....	<b>6</b>

---

## The SAS AppDev Studio 4.4 Eclipse Plug-ins

The SAS AppDev Studio 4.4 Eclipse Plug-ins support SAS application developers who use the open-source Eclipse IDE or a third-party IDE based on the Eclipse platform. Templates are provided that assist with the development of applications and web applications, including portlets, SAS Stored Processes, web-based reporting, and OLAP solutions. AppDev Studio 4.4 also enhances the standard Eclipse Java editor by integrating the SAS component API documentation into the Eclipse Help system.

AppDev Studio 4.4 creates SAS web applications only for SAS 9.4, and those projects are now automatically configured to work with SAS 9.4 and the SAS Web Infrastructure Platform.

---

## New Features

The first maintenance for SAS AppDev Studio 4.4 contains the following new features:

- a new command for configuring a Tomcat server to use for testing SAS web applications.
- new versions of the SAS Java Components and SAS Web Infrastructure Platform facets.

## Migrating Applications to AppDev Studio 4.4

AppDev Studio 4.4 cannot automatically migrate projects from previous versions. There are a few manual steps that you must perform. See the *SAS AppDev Studio 4.4 Eclipse Plug-ins Migration Guide, Second Edition* on the AppDev Studio Developer's Site ([support.sas.com/rnd/appdev/](http://support.sas.com/rnd/appdev/)) for information.

## Chapter 3

# Projects, Profiles, and Templates

---

<b>SAS Web Application Projects</b> .....	<b>7</b>
<b>Minimize the Number of Open Projects</b> .....	<b>8</b>
<b>Server Profiles</b> .....	<b>8</b>
Introduction .....	8
BI Server Profiles .....	9
Metadata Server Connection Profiles .....	9
<b>Templates</b> .....	<b>9</b>
<b>Template Descriptions</b> .....	<b>10</b>
SAS DataBean .....	10
SAS Information Delivery Portal Portlets .....	10
SAS Foundation Services Support .....	10
SAS Web Application Examples .....	11
SAS Web Infrastructure Platform Support .....	11
SAS Stored Process .....	12

---

## SAS Web Application Projects

SAS AppDev Studio web application development is as flexible as you need it to be. You can add to a project your code and third-party classes or tag libraries such as Apache Struts or JavaServer Faces. You can also add a JAR file of Java utility classes to a web application's `\WEB-INF\lib` directory.

SAS Web Application Projects also support the SAS Java Components and the SAS Web Infrastructure Platform, which includes features such as the Logon Manager, themes, and SAS Platform Services. This support is achieved by adding to the project static content, Eclipse Web Tools Platform facets ([www.eclipse.org/webtools/](http://www.eclipse.org/webtools/)), and JAR files from the SAS Versioned Jar Repository.

The static content can include configuration information, such as declarations in `web.xml`, other configuration files, and also file resources that are served by the web application.

Facets are a feature of the Web Tools Platform, and define functionality that can be added to a project. Two facets are added to every SAS Web Application Project: the “SAS Java Components” facet and the “SAS Web Infrastructure Platform” facet. (The “SAS Java Components” facet is the SAS 9.4 equivalent of the “SAS Web Module with WIK” facet.) Facets are versioned, and the 9.4.0.0000 version of the SAS Java

Components and SAS Web Infrastructure Platform facets corresponds to the initial release of SAS 9.4. AppDev Studio 4.4 for the first maintenance of SAS 9.4 uses version 9.4.1.0000 facets. You can view the facets of a SAS Web Application project by opening the project's Properties and selecting **Project Facets**.

In AppDev Studio 4.4, the SAS Web Application Project wizard always adds both facets to a new SAS Web Application project. Neither facet can be removed from the project. This means the SAS Web Application Project wizard cannot be used to create a SAS Web Application project if you do not want to include the "SAS Web Infrastructure Platform" facet. For a process to create a SAS Web Application project with only the "SAS Java Components" facet see ["Creating a SAS Web Application That Does Not Use the Web Infrastructure Platform"](#) on page 71.

The SAS Versioned Jar Repository is attached to the project and adds the JAR files needed to support the two included facets. The JAR files added to the project by the SAS Versioned Jar Repository are also included in the application's `\WEB-INF\lib\` directory. You can view these included JAR files by opening the project's SAS Repository. See ["Opening the SAS Repository Properties Editor"](#) on page 50.

In addition to the SAS Repository, there is also a SAS Tooling library included in the build path of the project. This library includes a JAR file that provides classes that are needed at only build time, such as the annotation classes. Because these classes are not needed at run time, this JAR file is not included when the project is deployed or exported to a WAR file.

---

## Minimize the Number of Open Projects

Because of the resources devoted to each open project, you should minimize the number of projects that you have open at one time. The fewer projects that you have open, the more responsive the development environment will be.

To open an existing project, right-click the project in the Project Explorer, and select **Open Project**. Open projects are indicated with an open folder icon.

To close an existing project, right-click the project and select **Close Project**.

---

## Server Profiles

### *Introduction*

To avoid repeatedly entering host names, port numbers, and other settings for the numerous servers in a SAS BI Server installation, AppDev Studio 4.4 provides server profiles that you can define and then use to simplify development against a specific BI installation. There are two types of server profiles used by AppDev Studio 4.4: the BI Server Profile, and the Metadata Server Connection Profile.

A BI Server Profile contains only enough information to uniquely identify a BI Server installation. The profile enables AppDev Studio 4.4 to communicate with an installation, and acquire additional information from it. Some tasks in AppDev Studio 4.4, such as adding template content to a project, let you select the BI Server Profile for the BI Server



installation that you want to target. When you do this the settings needed to complete the task are obtained from the BI Server Profile, avoiding the need for manual entry.

A Metadata Server Connection Profile is used to log in to the SAS Metadata Server for a particular BI Server installation. The Connection Profile contains credentials for the login and which BI Server Profile to use to connect to the associated Metadata Server. Because both profiles provide information and connection to required servers, you should create these profiles before starting a project.

### **BI Server Profiles**

A BI Server Profile contains information about a particular BI Server installation.

To create a new BI Server Profile, use the “Create a SAS BI Server Profile” cheat sheet:

1. From Eclipse, select **Help** ⇒ **Cheat Sheets**.
2. Expand **SAS AppDev Studio**, select **Create a SAS BI Server Profile**, and then click **OK**.

If the “Create a SAS BI Server Profile” cheat sheet has already been run, reset it by right-clicking the cheat sheet name and selecting **Restart all tasks**.

### **Metadata Server Connection Profiles**

Connection Profiles help automate logging in to a SAS Metadata Server by managing the relationship between user credentials and SAS BI Server profile. You can connect via one Connection Profile at a time.

To create a new Connection Profile, use the “Create a Connection Profile” cheat sheet:

1. From Eclipse, select **Help** ⇒ **Cheat Sheets**.
2. Expand **SAS AppDev Studio**, select **Create a Connection Profile**, and then click **OK**.

---

## **Templates**

The SAS templates consist of code that helps you rapidly develop SAS Web Applications or implement a particular feature. You can add these templates to a project when it is created, or add them to an existing project later.

The SAS Web Application Examples templates provide you with code and resources in various states of completion. The amount of work needed to make the example functional depends on the template. Some templates create examples that are ready to run using only the information provided when you add the template.

Although this User's Guide focuses on SAS Web Applications created in AppDev Studio 4.4 or migrated from a previous version, non-Web application projects (SAS Java Projects and Eclipse Web Tools Dynamic Web projects) are also supported in AppDev Studio 4.4.

If you add a SAS Web Application Examples template to an Eclipse Dynamic Web project, the project is converted to a SAS Web Application Project.

## Template Descriptions

The SAS AppDev Studio 4.4 templates by category.

### **SAS DataBean**

SAS JDBC Databean Class

creates a JDBC Java data class that provides access to a data table.

### **SAS Information Delivery Portal Portlets**

DisplayURL Portlet

displays the contents of an HTML page. The URL of the page is specified at run time.

Editable Portlet

displays the contents of a string. The string is specified at run time.

Information Map Fixed Portlet

displays the contents of an Information Map. The Information Map is specified at design time and displays the map at run time.

Information Map Runtime Portlet

displays a list of Information Maps defined on your server and displays the selected map.

JSP Portlet

displays the output of a single JSP that is contained in your project.

Remote Portlet

displays the contents of a remote page inside a frame. The URL of the remote page is specified at design time.

Stored Process List Portlet

displays a list of SAS Stored Processes and executes the selected stored processes. The list is specified at design time.

Stored Process Single Portlet

displays the results of executing a single SAS Stored Process. The stored process is specified at design time.

### **SAS Foundation Services Support**

Context Listener For Local Services

creates a ServletContextListener class for deploying and destroying local SAS Foundation Services.

JAAS Login Configuration File

creates a login configuration file for JAAS authentication using a SAS Metadata Server.

Log4J Logging Configuration

adds a simple Log4J logging configuration file to the web application.

## **SAS Web Application Examples**

All the SAS Web Application Examples templates, listed below, use the SAS Web Infrastructure Platform (SAS WIP). For each template listed there is a corresponding template available in the AppDev Studio interface that uses the legacy SAS Foundation Services (SAS FS).

Information Map Default Servlet (uses SAS WIP)

creates a Default Information Map based on the Model 2 (MVC) Web Application Architecture. Both a JSP page and a Java file containing the servlet's class are created.

Information Map OLAPTableView Servlet (uses SAS WIP)

creates an Information Map OLAPTableView example based on the Model 2 (MVC) Web Application Architecture for use with OLAP data. Both a JSP page containing OLAPTableView custom tags and a Java file containing the servlet's class are created.

Information Map TableView Servlet (uses SAS WIP)

creates an Information Map TableView example based on the Model 2 (MVC) Web Application Architecture for use with relational data. Both a JSP page containing TableView custom tags and a Java file containing the servlet's class are created.

JDBC Default Servlet (uses SAS WIP)

creates a Default JDBC example based on the Model 2 (MVC) Web Application Architecture. Both a JSP page and a Java file containing the servlet's class are created.

JDBC TableView Servlet (uses SAS WIP)

creates a JDBC TableView example based on the Model 2 (MVC) web application architecture. Both a JSP page containing TableView custom tags and a Java file containing the servlet's class are created.

Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP)

creates a report viewer example that uses a servlet to access relational and OLAP data. The selected report is viewed in SAS Web Report Viewer. A Java file containing the servlet's class is created.

SAS Stored Process Servlet (uses SAS WIP)

creates a stored process example that uses a servlet to display the output of a stored process.

## **SAS Web Infrastructure Platform Support**

Examples Welcome Page

adds a JSP page that lists the SAS Web Applications Examples added to the project.

SAS Web Infrastructure Platform Applications Metadata Creation

creates support files for creating and deploying Application metadata that is required by the SAS Web Infrastructure Platform.

Log4J Logging Configuration

adds a simple Log4J logging configuration file to the web application.

### **SAS Stored Process**

Java Client for executing a SAS Stored Process  
creates a simple Java client that executes a SAS Stored Process and writes the results  
to a file.

## Chapter 4

# Walk-Through for Web Infrastructure Platform Templates

<b>Introduction</b> .....	<b>13</b>
<b>Part I: Create the Project and the Application Metadata</b> .....	<b>14</b>
Create a SAS Web Application Project .....	14
Add the Application Metadata Creation Template .....	16
Run the Launch File and Create the Application Metadata .....	17
<b>Part II: Add the JDBC TableView Template</b> .....	<b>18</b>
<b>Part III: Add a Welcome Page and Run the Application</b> .....	<b>21</b>
Add a Welcome Page Template .....	21
Run the Application .....	22
<b>Add a ReportViewer Servlet Template to the Project</b> .....	<b>23</b>
Add the ReportViewer Servlet Template .....	23
Restart the Server and Run the Application .....	23
<b>Add a SAS Stored Process Servlet Template to the Project</b> .....	<b>24</b>
Add the SAS Stored Process Servlet Template .....	24
Replace a Value in the Servlet Code .....	26
Restart the Server and Run the Application .....	26
Change a Stored Process Input Parameter .....	27
<b>Input and Output Parameters</b> .....	<b>28</b>
Input Parameters .....	28
Output Parameters .....	29

## Introduction

The following steps guide you through creating and running three of the available Web Infrastructure Platform templates. Many of the templates are similar, and a knowledge of one template transfers to the others. On-screen help is available for every part of every template. You should complete the entire walk-through to become familiar with the AppDev Studio interface and the template requirements.

This walk-through creates a SAS Web Application Project and then builds upon itself, incrementally adding three templates to the project. Because the templates use the same Tomcat Web server, no changes to the server parameters are necessary.

This walk-through assumes the following:

- a Tomcat Web server was installed and configured according to the cheat sheet “Configure a Tomcat Server for Testing” (part of the “New Workspace Setup” cheat sheet)
- a SAS BI Server Profile exists for the servers that you are developing for
- a Metadata Server Connection Profile exists for the server that you are developing for
- a data table, a Web Report Studio report, and a stored process are available on the SAS Metadata Server. You can use any data source of the appropriate type when you build the example, but this walk-through uses the `sashelp.class` table, a report based on that table, and the stored process “Sample: Multiple Output Formats.”

This walk-through adds the following templates to the project in this order:

1. JDBC TableView Servlet (uses SAS WIP)
2. ReportViewer Servlet (uses SAS Web Report Viewer and SAS WIP)
3. SAS Stored Process Servlet (uses SAS WIP)

---

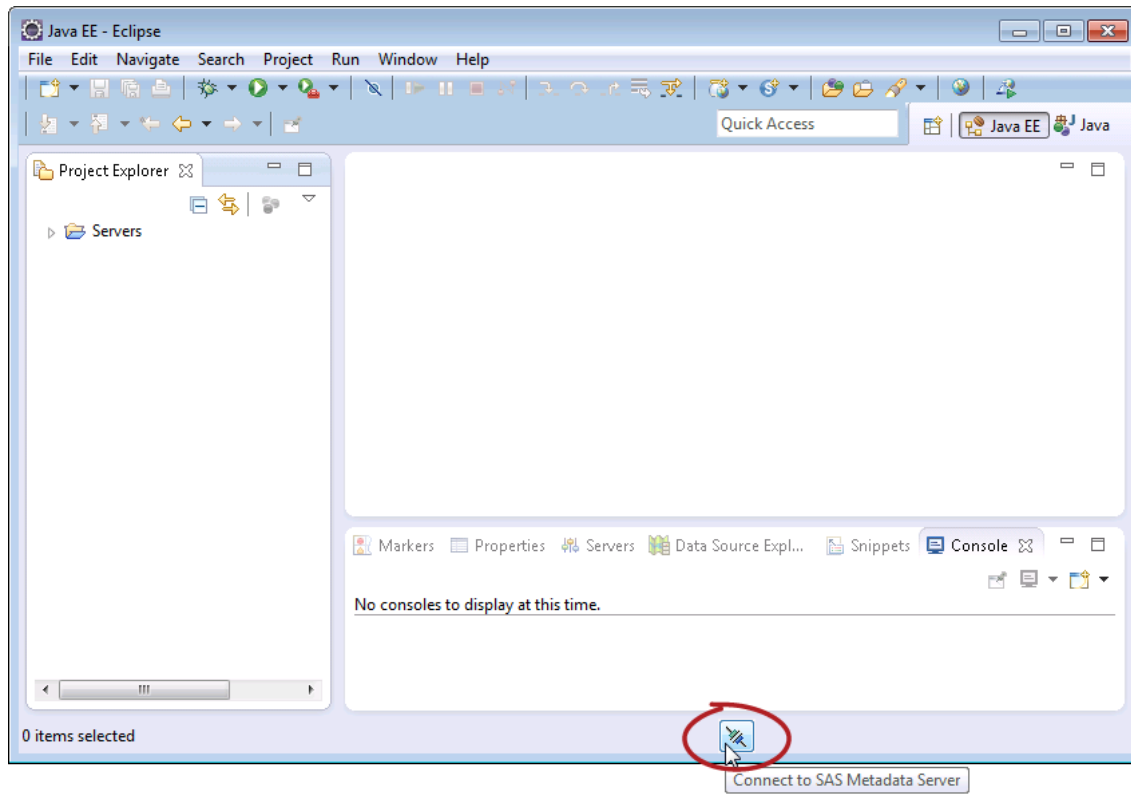
## Part I: Create the Project and the Application Metadata

### *Create a SAS Web Application Project*

Create the project to which you will add the templates.

1. Connect to a SAS Metadata Server.

Although not necessary to create a new project, connecting to a metadata server as the first step in this walk-through ensures that you have defined a BI Server Profile and Metadata Server Connection Profile. For help with setting up the profiles, see “Server Profiles” on page 8.



2. Select **File** ⇒ **New** ⇒ **Other**.
3. Expand **SAS AppDev Studio**.
4. Select **SAS Web Application Project**, and click **Next**.
5. For the project name, enter **MyProject**, and click **Next**. In the next section you add the Metadata Creation template.

The project name is used as the context name, and cannot contain spaces.

**New SAS Web Application Project**

Create a new SAS Web Application in the workspace or in an external location.

Project name: MyProject

Context Root: MyProject

Java Source Folder: src

Web Content Folder: WebContent

Target runtime: ADS Apache Tomcat v7.0

Servlet Version: 2.4

**Project Location**

Create project files in the workspace

Create project files in an external location:

Directory:  Browse...

< Back Next > Finish Cancel

### ***Add the Application Metadata Creation Template***

The SAS Web Infrastructure Application Metadata Creation template adds to a project the files that enable you to create, delete, and copy the metadata needed to communicate with the Web Infrastructure Platform Logon Manager. Because integration with the SAS Web Infrastructure Platform requires this application metadata, this template should be the first that you add to a project.

1. Select the **Add Template Content** check box.
2. Expand the following folders:
  - **SAS Java Web Application**
  - **SAS Web Infrastructure Platform Support**
3. Select **SAS Web Infrastructure Platform Application Metadata Creation**, and click **Next**.
4. Select the **BI Server Profile** that matches the BI installation that you plan to develop for.

The BI Server Profile that is selected by default is the one associated with the metadata connection profile that you used to connect to the metadata server.

5. For the **Port**, enter **8081**. This port should match the port used for the Tomcat Web server.



The screenshot shows a dialog box titled "New Content from a Template" with the subtitle "SAS Web Infrastructure Platform Application Information". The main instruction is "Specify application information for the SAS Web Infrastructure Platform".

The dialog contains the following fields and controls:

- BI Server Profile:** A dropdown menu showing "BIserver.place.sas.com BI Server Profile" and a "New..." button.
- Application Information:**
  - Application Name:** A text box containing "MyProject".
  - Description:** A text box containing "Application description."
- Connection Information:**
  - Protocol:** A dropdown menu showing "http".
  - Web Server Host:** A dropdown menu showing "TestServer.sas.com".
  - Port:** A dropdown menu showing "8081".
- Restore All Defaults:** A button.
- Navigation:** A "?", "< Back", "Next >", "Finish" (highlighted), and "Cancel" buttons.

6. Click **Finish**, and then close the `Application.xml` file.

Although the project is created and the Create Metadata template has been added to the project, you must still create the application metadata needed to communicate with the SAS Web Infrastructure Platform.

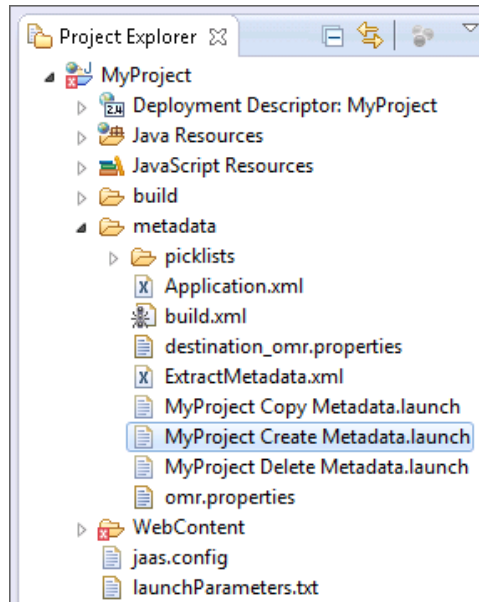
### **Run the Launch File and Create the Application Metadata**

When you added the Create Metadata template, an Ant launch configuration file named **MyProject Create Metadata.launch** was added to the project. The application metadata identifying the web application to the Web Infrastructure Platform Logon Manager is created by executing the launch file (located in `\metadata\`).

Metadata files created during the execution of this launch file appear in `\metadata\temp\`. For information about the files added to the project as a result of adding this template and creating the metadata, see [“Files Added by the Metadata Creation Template” on page 37](#).

1. Ensure that the SAS Metadata Server is running in the BI installation whose SAS BI Server Profile was selected earlier.
2. Select **Window** ⇒ **Show View** ⇒ **Console** to ensure that the Console is visible.
3. Expand the project's metadata folder.
4. Right-click the **MyProject Create Metadata** file, and select **Run as** ⇒ **MyProject Create Metadata**.

Verify that BUILD SUCCESSFUL appears at the end of the output logged to the Console.




---

## Part II: Add the JDBC TableView Template

1. Select **File** ⇒ **New** ⇒ **Other**.
2. Expand **SAS AppDev Studio**.
3. Select **Add Template Content to Project**, and click **Next**.
4. Expand **SAS Java Web Application** and **SAS Web Application Examples**.
5. Select **JDBC TableView Servlet (uses SAS WIP)**, and click **Next**.
6. Click **Next** to accept the Template Configuration Parameters.
7. Accept the **BI Server Profile** by clicking **Next**. The BI Server that you plan to develop for should already be selected.
8. Enter the user name and password for the selected server.

**JDBC Connection Properties**

Select a driver and enter appropriate values to define a JDBC connection.

**Driver Information**

JDBC driver: SAS IOM JDBC Driver

Driver class: com.sas.rio.MVADriver

Datasource URL: jdbc:sasnom://BIserver.place.com:8591

Driver Properties

**Server Information**

Host: BIserver.place.com Port: 8591

Automatically update fields and properties with historical values when host or port is modified

**User Information**

User name: UserID

Password: \*\*\*\*\*

Test Connection

< Back Next > Finish Cancel

9. Click **Test Connection**.

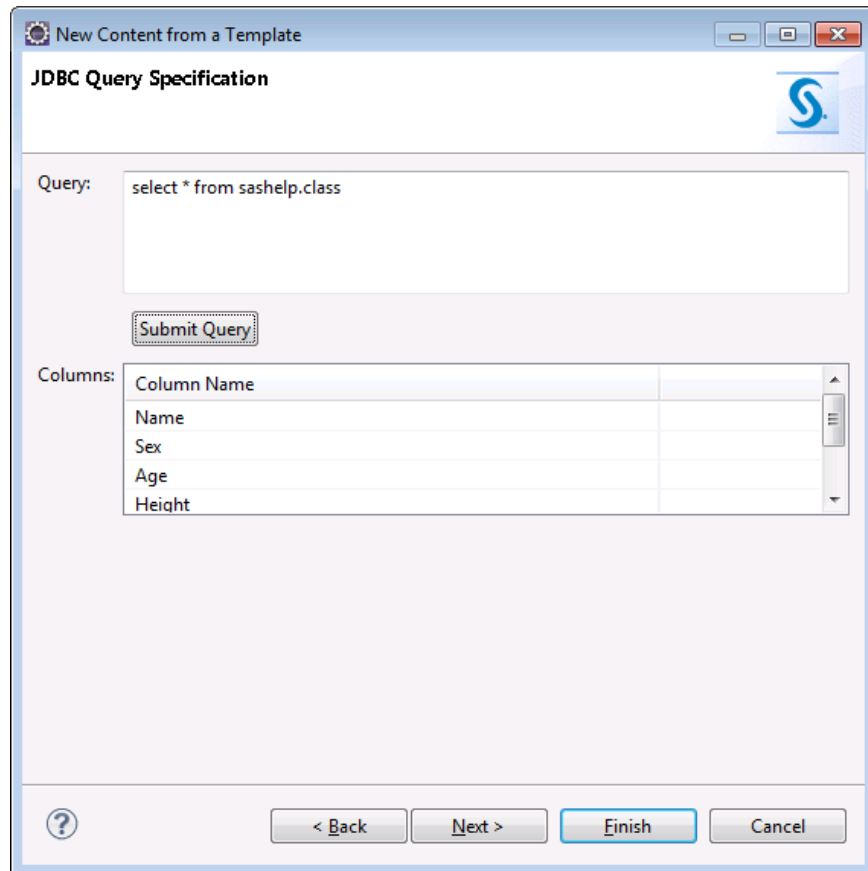
If you do not receive the message “Connection test succeeded,” ensure that the BI Server Profile is correct and the BI Server is running on the expected port.

In the future, when you want to provide a path to your data, click **Driver Properties** and define the **librefs** property.

10. Click **Next**.

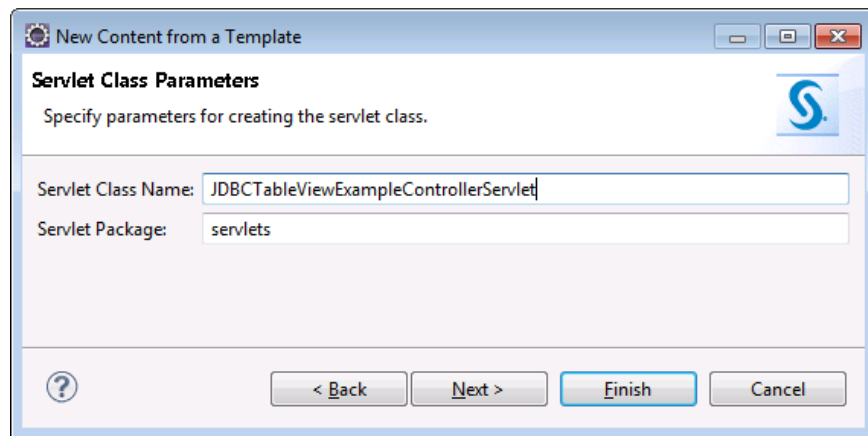
11. For the **Query**, enter `select * from sashelp.class`.

12. Click **Submit Query** to test the query and display the resulting column names.

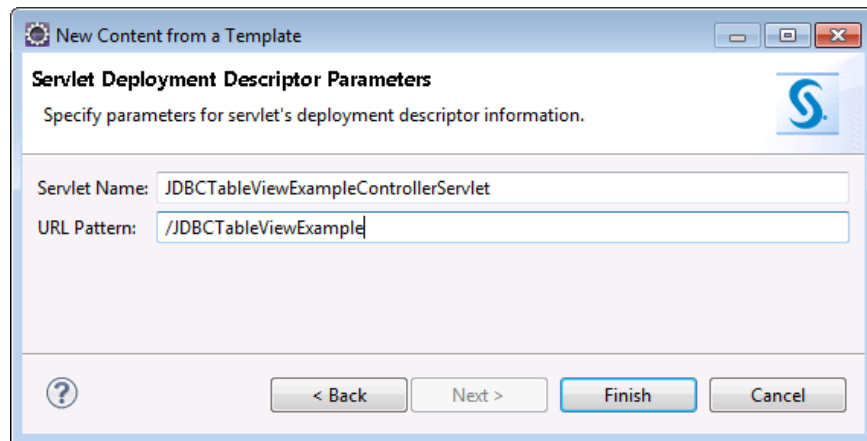


13. Click **Next**.

14. Accept the Servlet Class Parameters by clicking **Next**.



15. Click **Finish** to accept the Servlet Deployment Descriptor Parameters and add the JDBC TableView Servlet template to the project.



The JSP and Java files for the servlet are opened. Close both files.

---

## Part III: Add a Welcome Page and Run the Application

### Add a Welcome Page Template

To help make testing as easy as possible, the Welcome Page template defines a starting point in the web application that is suitable for integration with the SAS Web Infrastructure Platform. The Welcome Page template works in conjunction with the web application example templates that add data about the example to `\WEB-INF\sas_examples.xml`.

The Welcome Page template adds a JSP file (default name `sas_examples.jsp`) to the project, and appends the filename to the welcome-file-list declaration in `\WebContent\WEB-INF\web.xml`.

At run time, after successfully logging on to the SAS Web Infrastructure Platform Logon Manager, this JSP is displayed provided that it is the first resource in the welcome-file-list that exists in the web application. When the Welcome Page is displayed, a link is available for each SAS web application template added to the project.

If other resources in the welcome-file-list exist in the web application, you might need to move the `sas_examples.jsp` entry to the top of the list to ensure that it is executed immediately after logon. Because the Examples Welcome Page is not a production entry point, you can move it to the top of the list when you want to use its features and move it down or remove it when you want a different welcome file served.

Add a Welcome Page template by following these steps:

1. From Eclipse, select **File** ⇒ **New** ⇒ **Other**
2. Select **Add Template Content to Project**, and click **Next**.
3. For **Project**, select **MyProject**.
4. Expand **SAS Java Web Application** and then **SAS Web Infrastructure Platform Support**.
5. Select **Examples Welcome Page**, and then click **Next**, and then **Finish**.

## Run the Application

1. From the Servers View, right-click the server and select **Add and Remove**.
2. Select **MyProject** and add it to the list of **Configured** projects.
3. Click **Finish**.
4. Right-click the server and select **Publish**.

The application is copied to the server. Wait until the status of the server and project is **Synchronized** before proceeding.

The first time an application is published, over 100 megabytes of static content and JAR files are copied to the server. Subsequent publishes only copy files that have changed.

5. From the Servers View, right-click the server and select **Start**. Wait until the server State is **started**.

*Note:* A 120 second server time-out for Tomcat is recommended. Adjust the time according to the speed of your system and the number of web applications that you are attempting to start.

6. In the Project Explorer, right-click **MyProject** and select **Run As** ⇒ **Run on Server**.
7. Ensure that the correct server is selected and click **Finish**.

Because the server is already started, the web browser should open to the BI Server page.

8. Log on to the BI Server.
9. The Welcome Page is displayed.

### SAS AppDev Studio Eclipse Plug-ins Template Examples

This page displays a list of examples added to this Web application by the SAS AppDev Studio Eclipse Plug-ins. A link is provided for each example, organized by example type.

#### Example Type: JDBC TableView Servlet (uses SAS WIP)

[JDBCTableViewExampleControllerServlet](#)

#### Logoff Link

[Log Off](#)

Note: The localized text in this page was hard coded when this page was created by the SAS AppDev Studio Eclipse Plug-ins. It does not otherwise support localization.

10. Click the link for the JDBC TableView Example.

	Name	Sex	Age	Height	Weight
1	Alfred	M	14.0	69.0	112.5
2	Alice	F	13.0	56.5	84.0
3	Barbara	F	13.0	65.3	98.0
4	Carol	F	14.0	62.8	102.5
5	Henry	M	14.0	63.5	102.5
6	James	M	12.0	57.3	83.0

11. Close the JDBC TableViewer window and log off.

You should always log off when you are finished testing. Stopping the test server while still logged in might not completely remove the session.

---

## Add a ReportViewer Servlet Template to the Project

### Add the ReportViewer Servlet Template

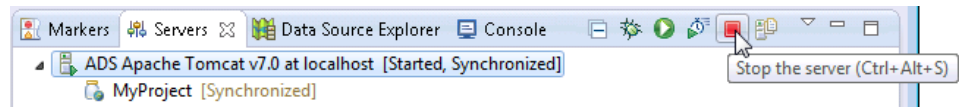
This section adds a template to an existing project. Because the template uses the same Tomcat server, server parameters, and Welcome Page to run the ReportViewer Servlet, you need to provide only the information collected when adding the template.

1. Select **File** ⇒ **New** ⇒ **Other**.
2. Expand **SAS Appdev Studio**, select **Add Template Content to Project**, and click **Next**.
3. Expand **SAS Java Web Application** and **SAS Web Application Examples**.
4. Select **Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP)**, and click **Next**.
5. Accept the Template Configuration Parameters by clicking **Next**.
6. Accept the SAS Web Infrastructure Platform Information by clicking **Next**. The proper BI Server should already be selected.
7. Accept the Servlet Class Parameters by clicking **Next**.
8. Accept the Servlet Deployment Descriptor Parameters by clicking **Finish**.

### Restart the Server and Run the Application

1. From the Servers View, stop the server (do not use Restart). Wait until the server State is **stopped**.

For why you should avoid the Restart command, see the [“Tomcat Shutdown Issue” on page 42](#).



2. From the Servers View, start the server.
3. In the Project Explorer, right-click **MyProject** and select **Run As** ⇒ **Run on Server**.
4. Ensure that the correct server is selected and click **Finish**.
5. Log on to the BI server.
6. The Welcome Page is displayed with links for both added templates: the original JDBC TableView Servlet, and the new ReportViewer Servlet.

**SAS AppDev Studio Eclipse Plug-ins Template Examples**

This page displays a list of examples added to this Web application by the SAS AppDev Studio Eclipse Plug-ins. A link is provided for each example, organized by example type.

**Example Type: JDBC TableView Servlet (uses SAS WIP)**

[JDBCTableViewExampleControllerServlet](#)

**Example Type: Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP)**

[ReportViewerExampleControllerServlet](#)

**Logoff Link**

[Log Off](#)

Note: The localized text in this page was hard coded when this page was created by the SAS AppDev Studio Eclipse Plug-ins. It does not otherwise support localization.

7. Click the link for the ReportViewer Servlet.

Search ▼

Location: SAS Folders Up one level Show description

Name ▲	Type	Date Modified
My Folder	Folder	12/02/2013
Products	Folder	11/30/2013
Shared Data	Folder	11/30/2013
System	Folder	11/30/2013
User Folders	Folder	11/30/2013

8. If available, navigate to a report (a file ending with **.srx**), and select it.
- If you are running AppDev Studio 4.4 and have not applied maintenance, see the AppDev Studio Migration Guide for information about restoring back link behavior.

---

## Add a SAS Stored Process Servlet Template to the Project

### Add the SAS Stored Process Servlet Template

In this section, as you did earlier in the walk-through, you are adding a template to an existing project. The server is already configured for testing.

1. Select **File** ⇒ **New** ⇒ **Other**.



2. Expand **SAS Appdev Studio**, select **Add Template Content to Project**, and click **Next**.
3. Expand **SAS Java Web Application** and then **SAS Web Application Examples**.
4. Select **SAS Stored Process Servlet (uses SAS WIP)**, and click **Next**.
5. Click **Next** to accept the Template Configuration Parameters.
6. Accept the **BI Server Profile** by clicking **Next**. The BI Server that you plan to develop for should already be selected.
7. Select a stored process by clicking **Change**, selecting **Sample: Multiple Output Formats**, and then clicking **OK**.

Click **Next**.

The screenshot shows a dialog box titled "New Content from a Template" with the subtitle "SAS Stored Process Servlet". The main instruction is "Enter settings about the SAS Stored Process which will be executed in a servlet." The dialog is divided into several sections:

- Connection profile:** A text field containing "sasdemo" and a "Change..." button.
- Stored Process:** A section containing:
  - Name:** A text field containing "Sample: Multiple Output Formats" and a "Change..." button.
  - Generate streaming results in browser**
  - Generate list of output parameter results**
- Output:** A section containing:
  - Write SAS log to ServletContext log**

At the bottom of the dialog, there is a help icon (question mark), and four buttons: "< Back", "Next >", "Finish", and "Cancel".

8. Click **Next** to accept the Servlet Class Parameters, and then click **Next** again to accept the Servlet Deployment Descriptor Parameters.
9. Click **Finish**.

The `StoredProcessWebApp` project is now created and the SAS Stored Process Servlet template is added. leave open the Java file containing the servlet (`StoredProcessDriverServlet.java`).

## Replace a Value in the Servlet Code

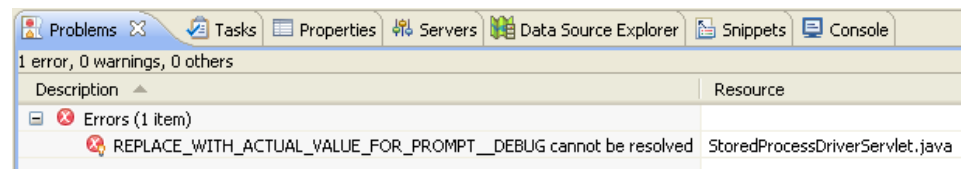
The generated servlet code lacks a value for one of the stored process input parameters. The default value was not defined in the metadata for the stored process, and therefore could not be included in the generated code. You must edit the code and provide a valid value before the servlet can compile and run.

1. Display the Markers view if it is not visible.

Toggle the Markers view by selecting **Window** ⇒ **Show View** ⇒ **Other** ⇒ **General** ⇒ **Markers**.

2. Look in the Markers view under Java Problems for this error:  
REPLACE\_WITH\_ACTUAL\_VALUE\_FOR\_PROMPT\_DEBUG.

If the error is not in the Markers view, ensure that automatic building is enabled (**Project** ⇒ **Build Automatically**).



3. Double-click the error in the Markers view to go to the error in the Java file.

```
// Prompt: _debug
private static final String _DEBUG_KEY = "_debug";
private static final Object _DEBUG_VALUE = REPLACE_WITH_ACTUAL_VALUE_FOR_PROMPT_DEBUG;
```

4. Replace the problem value with **"log"**, including the quotation marks.

```
// Prompt: _debug
private static final String _DEBUG_KEY = "_debug";
private static final Object _DEBUG_VALUE = "log";
```

5. Save the file.

The error disappears, assuming that automatic building is enabled.

For more information about input parameters, including how to see what input parameter values and data types are valid in the code, see [“Input and Output Parameters” on page 28](#).

## Restart the Server and Run the Application

1. Stop and then start the server from the Servers View (do not use Restart). Wait until the server State is Stopped.

For why you should avoid the Restart command, see the [“Tomcat Shutdown Issue” on page 42](#).

2. Right-click the project and select **Run As** ⇒ **Run on Server**.
3. Ensure that the correct server is selected and click **Finish**.
4. Log in.
5. The Welcome Page is displayed with all three templates listed.

### SAS AppDev Studio Eclipse Plug-ins Template Examples

This page displays a list of examples added to this Web application by the SAS AppDev Studio Eclipse Plug-ins. A link is provided for each example, organized by example type.

**Example Type: JDBC TableView Servlet (uses SAS WIP)**

[JDBCTableViewExampleControllerServlet](#)

**Example Type: Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP)**

[ReportViewerExampleControllerServlet](#)

**Example Type: SAS Stored Process Servlet (uses SAS WIP)**

[StoredProcessDriverServlet](#)

**Logout Link**

[Log Off](#)

Note: The localized text in this page was hard coded when this page was created by the SAS AppDev Studio Eclipse Plug-ins. It does not otherwise support localization.

- Click the Stored Process Servlet. The output is HTML, the default for this stored process.

### Data Set SASHELP.RETAIL in HTML Format

Retail sales in millions of \$	DATE	YEAR	MONTH	DAY
\$220	80Q1	1980	1	1
\$257	80Q2	1980	4	1
\$258	80Q3	1980	7	1
\$295	80Q4	1980	10	1

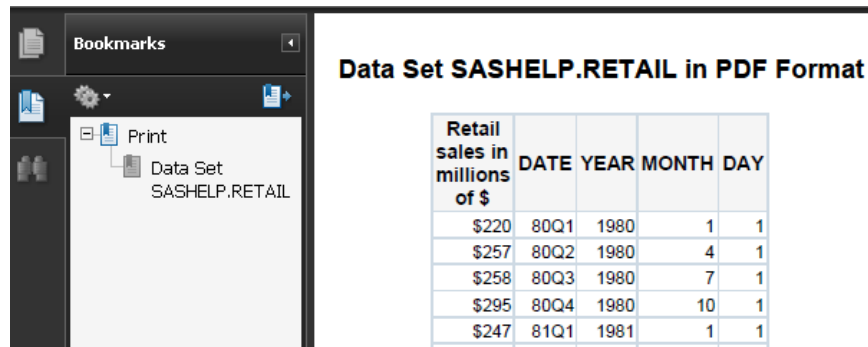
### Change a Stored Process Input Parameter

One of the advantages of using a stored process is that you can use input parameters in the Java code to change the behavior of the servlet. For example, to change the output type of the “Sample: Multiple Output Formats” stored process used in this walk-through to PDF or XML, follow these steps:

- In the project, open the folders **Java Resources** and then **servlets**. Open **StoredProcessDriverServlet.java**.
- Search for `_ODSDEST_VALUE =`, and then set that variable to **PDF** or **XML**.
- Save the file.
- Stop and then start the server (do not use Restart), and then run the application by right-clicking the project and selecting **Run As** ⇒ **Run on Server**.

For why you should avoid using the Restart command, see the “[Tomcat Shutdown Issue](#)” on page 42.

- On the Welcome Page, click the Stored Process Servlet. The output is a PDF (or XML).



You can also change the data set used by the servlet by changing the `DATASET_VALUE`.

Note that the input parameters changed in this walk-through are specific to the “Sample: Multiple Output Formats” stored process. Other stored processes will have different input parameters.

To see how drill-down functionality works, add another SAS Stored Process Servlet template to the project, and use the European Demographic Data stored process.

---

## Input and Output Parameters

### Input Parameters

Input parameters enable you to pass values to a stored process. The default value of an input parameter, if it is defined, is in the metadata for the stored process.

If a default value is defined for an input parameter, and the data type of the value is text or numeric (integer or floating-point), the input parameter is set to the default value.

If there is no default value defined for an input parameter, or if the type of the input parameter is not text or numeric, then the input parameter is set to `REPLACE_WITH_ACTUAL_VALUE_FOR_PROMPT_YYY`, where `YYY` is the name of the input parameter. This fabricated text causes a compiler error that is listed in the Eclipse Markers view. To build the project, replace the text with a valid value for that input parameter.

Note that the input parameter values are declared as class instance variables. When these particular input parameter declarations appear within a servlet, the values should be specified as constants that will be used for all requests to the servlet. If you want to update the servlet code to obtain input parameter values from request parameters, do not to store the input parameter values in these class instance variables because there will be collisions if there are concurrent requests to the servlet. Instead, use local variables to hold request parameter values.

Input parameters are represented as prompts in SAS Management Console. To examine the metadata for an input parameter, including its type and default value, follow these steps while using SAS Management Console:

1. Right-click the stored process and select **Properties**.
2. Change to the **Parameters** tab.
3. Select a prompt (input parameter) and click **Edit**.

4. Change to the **Prompt Type and Values** tab.

For more information about stored processes and input parameters, see the following resources:

- the comments in the generated code (`StoredProcessDriver.java` or `StoredProcessDriverServlet.java`).
- the *SAS Stored Processes: Developer's Guide* available on the SAS Integration Technologies documentation page at <http://support.sas.com/documentation/onlinedoc/inttech/>, in particular Appendix 3, “Formatting Prompt Values and Generating Macro Variables from Prompts”.
- the SAS Foundation Services Javadoc contained in the SAS BI API Documentation on the SAS AppDev Studio Developer's Site at <http://support.sas.com/rnd/appdev/>. In particular, examine the package summary for `com.sas.services.storedprocess`.

## Output Parameters

SAS Stored Processes can return results to a client via output parameters. Output parameters are defined in the metadata for a stored process using SAS Management Console. When a stored process returns output parameters, you can retrieve the values from the `StoredProcessFacade` with the `getOutputParametersWithValues()` method. The values are returned in a `java.util.List`.

An example of retrieving the output parameter values from the `StoredProcessFacade` and writing the results to the output file can be found in the main driver class of the Stored Process Java Client template.

An example of retrieving the output parameter values from the `StoredProcessFacade` and writing the results to the browser can be found in the main driver servlet class of the Stored Process Servlet template.

In both examples, you can modify the processing of the output parameter results as appropriate to the SAS Stored Process executed in the example.

For a more detailed explanation of output parameters, see “Using Output Parameters” in the *SAS Stored Processes Developer's Guide* available on the SAS Integration Technologies documentation page at <http://support.sas.com/documentation/onlinedoc/inttech/>.



## Chapter 5

# Walk-Through for Data-Driven Project Creation

---

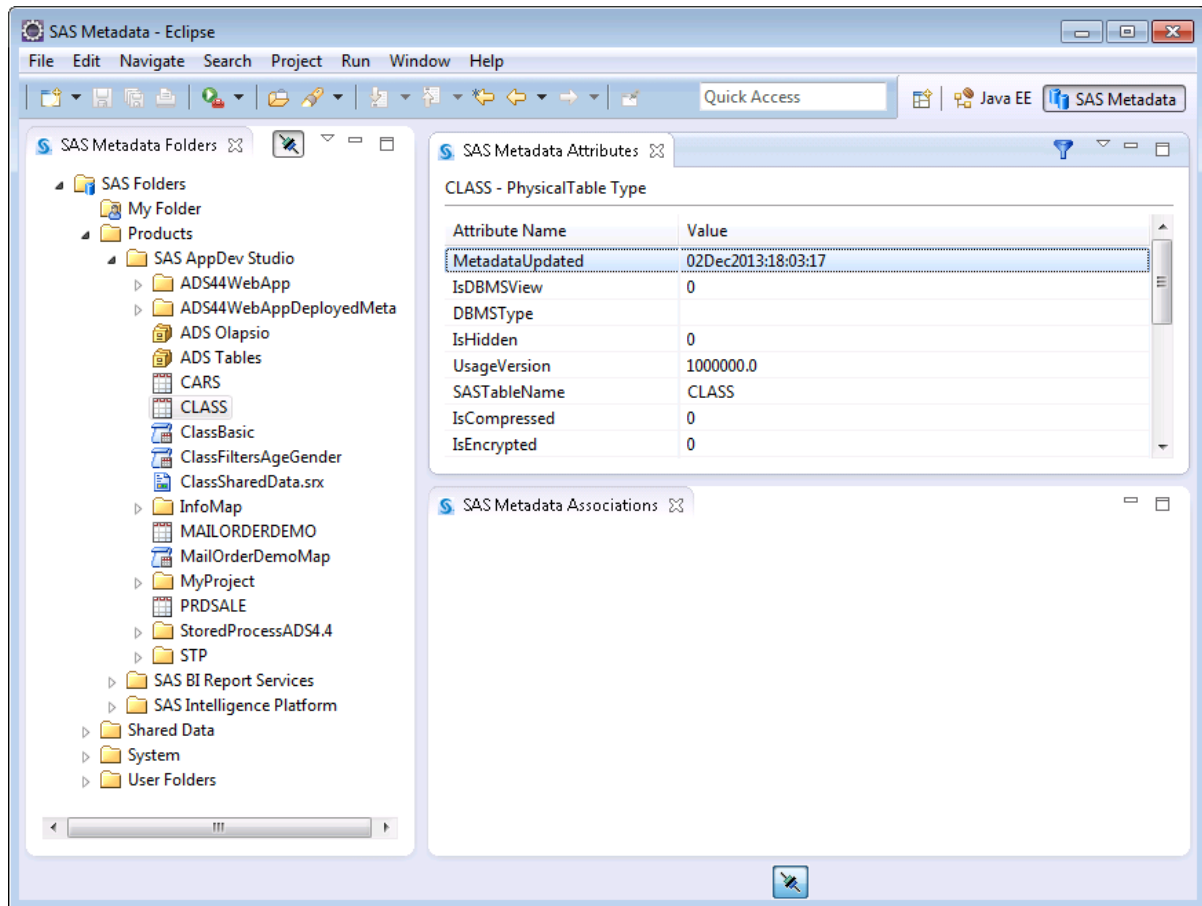
<b>Create an Information Map Fixed Portlet from Data</b> . . . . .	<b>31</b>
Introduction . . . . .	31
Switch to the SAS Metadata Perspective and Create the Project . . . . .	32
Deploy the Portlet . . . . .	34
View the Portlet in the SAS Information Delivery Portal . . . . .	35
<b>The Portlet Editor</b> . . . . .	<b>36</b>

---

## Create an Information Map Fixed Portlet from Data

### *Introduction*

This walk-through explains project creation from the SAS Metadata perspective. The SAS Metadata perspective functions as both a metadata explorer and as a starting point for data-driven development. You can create projects from tables, information maps, and stored processes. From those data sources, you can create SAS Web Application projects or SAS Java Projects. The type of template that you can add to a project is determined by the type of data that you select from the SAS Metadata perspective.



This walk-through assumes the following:

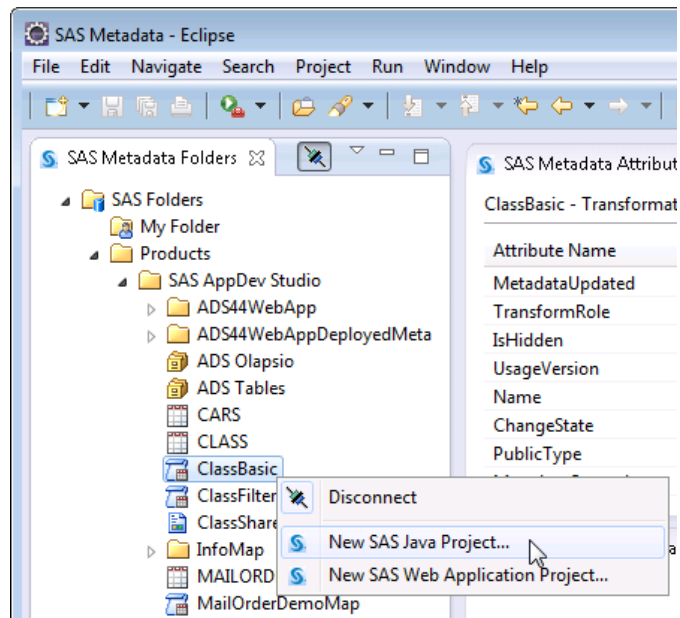
- you have access to a SAS Information Delivery Portal.
- you have access to an Information Map on the Metadata server.
- you have Write access to the portal deployment directory (for example, `\Lev1\Web\Applications\SASPortlets4.4\Deployed`).
- a Metadata Server Connection Profile exists for the server that you are developing against.

### **Switch to the SAS Metadata Perspective and Create the Project**

1. Connect to the SAS Metadata Server if you are not already.
2. Switch to the SAS Metadata perspective. Select **Window** ⇒ **Open Perspective** ⇒ **Other**, and then choose **SAS Metadata**.
3. Navigate to an Information Map. Right-click it and select **New SAS Java Project**.

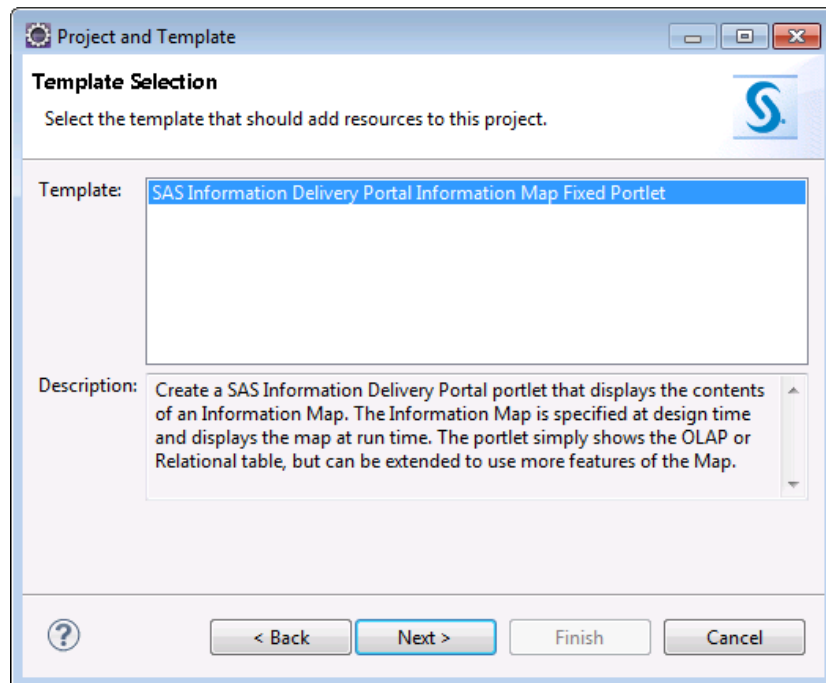
This example uses an Information Map based on the `sashelp.class` data set.



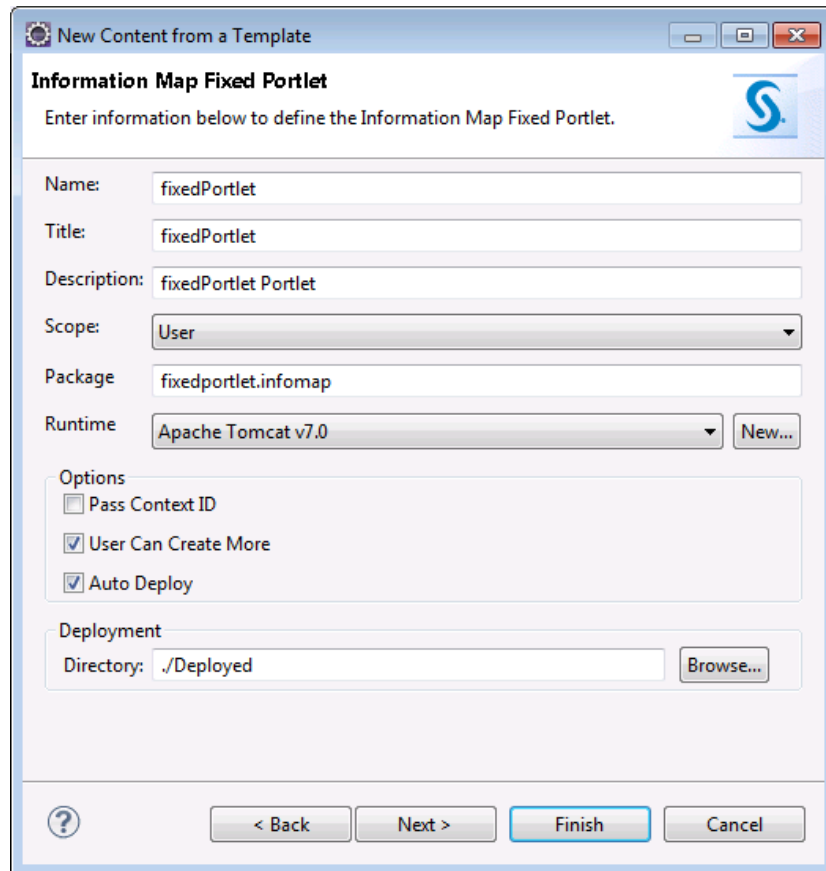


4. Name the project **fixedPortlet**, and click **Next**.
5. Click **Next** to accept the selected template: SAS Information Delivery Portal Information Map Fixed Portlet.

The template list is determined by the type of data that you selected from the Metadata perspective.



6. Click **Finish** to accept the Information Map Fixed Portlet configuration and create the project.



## Deploy the Portlet

To deploy the portlet to the SAS Information Delivery Portal, create a par (portlet archive) file and copy it to the deployment directory. The par file contains all generated classes and static content.

1. Switch to the Java perspective.
2. Close the `portlet.xml` file.

For information about the `portlet.xml` editor, see “The Portlet Editor” on page 36.

3. Open the project directory, right-click `fixedPortletParBuild.xml`, and select **Build and Copy Par**.

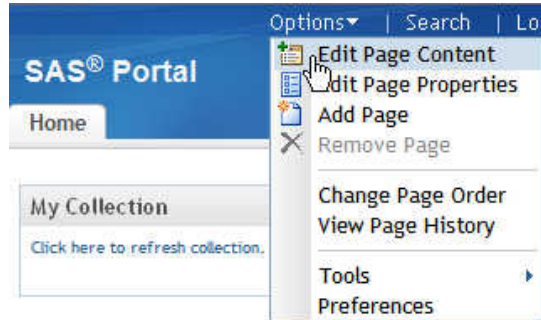
This XML file was created when you added the Fixed Portlet template, and follows the pattern `<projectName>ParBuild.xml`. You can specify additional content to include in the par file by editing this file.

4. Navigate to or enter the path for the deployed portal directory: `\Lev1\Web\Applications\SASPortlets4.4\Deployed`, and click **OK**.

The par file is created, appears in the file list in Eclipse, and is copied to the `\Deployed` directory. The Par filename is created with the pattern `<projectName>.par`. In this case, the Par file is named `fixedPortlet.par`.

### View the Portlet in the SAS Information Delivery Portal

1. Navigate to your SAS Information Delivery Portal. For example:  
BIserver.place.com:8080/SASPortal
2. Select **Customize** ⇒ **Edit Page** ⇒ **Edit Page Content**.



3. Click **Add Portlets**.
4. For the **Portlet type** select **fixedPortlet**.
5. For the **Name**, enter **fixedPortlet**.
6. Click **Add**, **Done**, and then **OK** to confirm your choices and return to the portal Home.
7. The portlet is displayed in the SAS Information Delivery Portal.

	Age	Height	Name	Sex	Weight
1	14	69	Alfred	M	112.5
2	13	56.5	Alice	F	84
3	13	65.3	Barbara	F	98
4	14	62.8	Carol	F	102.5
5	14	63.5	Henry	M	102.5
6	12	57.3	James	M	83
7	12	59.8	Jane	F	84.5

## The Portlet Editor

The portlet editor is a graphical interface for editing the `portlet.xml` file. You can open the portlet editor by double-clicking a `portlet.xml` file.

Using the portlet editor, you can specify several portlet options, and add parameters and actions. Actions are callbacks that the portal executes depending on the situation. For example, when a portlet is displayed the default action is executed. When you create an action, the portlet editor creates a stub and adds it to the `portlet.xml` file. For information about portlets and actions, see *Developing Portlets for the SAS Information Delivery Portal* at <http://support.sas.com/documentation/onlinedoc/portal/>.

The screenshot shows the Portlet Editor interface with the following sections:

- Portlets:** A list of portlets found in the `portlet.xml` file, currently showing 'fixedPortlet (Local)'.
- Portlet Details:** Parameters set for the selected portlet.
  - General:**
    - Name: fixedPortlet
    - Title: fixedPortlet
    - Editor Type:  None  Portal  Portlet
    - Pass Context ID
    - Show Edit Properties
    - Initializer: fixedportlet.infomap.Initializer
  - Deployment:**
    - Scope:  User  Group
    - Group: (empty field)
    - Auto Deploy
    - User Can Create More
  - Parameters:**

Name	Value
display-page	Viewer.jsp
map-name	SBIP://METASERVER/Products/SAS AppDev Studio/ClassBasic(InformationMap)
  - Actions:**

Name	Type/URL	Default
display	fixedportlet.infomap.actions.Display	Yes

Buttons for 'Add...', 'Edit...', and 'Remove' are available for both the Parameters and Actions tables. At the bottom, there are tabs for 'Properties' and 'Source'.

## Chapter 6

# Template and Testing Details

<b>Files Added by the Metadata Creation Template</b> . . . . .	<b>37</b>
Location of Application Metadata . . . . .	37
The Launch Files . . . . .	38
Application.xml . . . . .	38
build.xml . . . . .	38
destination_omr.properties . . . . .	38
omr.properties . . . . .	39
<b>Copying the Application Metadata</b> . . . . .	<b>39</b>
<b>Files Added by the Stored Process Java Client Template</b> . . . . .	<b>39</b>
Introduction . . . . .	39
StoredProcessDriver.java . . . . .	39
StoredProcessConnection.java . . . . .	40
StoredProcessFacade.java . . . . .	40
StoredProcessDriver.properties . . . . .	40
jaas.config . . . . .	40
launchParameters.txt . . . . .	40
<b>Files Added by the Stored Process Servlet Template</b> . . . . .	<b>41</b>
Introduction . . . . .	41
StoredProcessDriverServlet.java . . . . .	41
StoredProcessConnection.java . . . . .	41
StoredProcessFacade.java . . . . .	41
<b>Tomcat Configuration Details</b> . . . . .	<b>42</b>
Tomcat Support . . . . .	42
Configuring a Tomcat Server . . . . .	42
Tomcat Shutdown Issue . . . . .	42
<b>Deployment and Authentication</b> . . . . .	<b>43</b>

## Files Added by the Metadata Creation Template

### *Location of Application Metadata*

When you add the SAS Web Infrastructure Platform Metadata Creation template to a project, and then you use that template to create metadata, files are added to the project. These metadata files are stored in the `\metadata` folder or one of its subdirectories.

## The Launch Files

When you add the Metadata Creation template to a project, three Ant configuration files (`.launch`) are added to the project. Their names indicate the task that they perform when executed:

- **ProjectName Copy Metadata.launch**

See “Copying the Application Metadata” on page 39.

- **ProjectName Create Metadata.launch**

For instructions on how to do this, see “Run the Launch File and Create the Application Metadata” on page 17 in the walk-through.

- **ProjectName Delete Metadata.launch**

## Application.xml

This file contains values that are used in the application metadata. For example, the name of the application is stored in the Name attribute of the Application element, and the Protocol, Host, Port, and Service attributes of the ApplicationUri element are combined to form the URL that the SAS Web Infrastructure Platform’s Logon Manager returns to the web application after a successful login: `<Protocol>://<Host>:<Port><Service>`. The slash that follows the Port in the URL is part of the Service value.

You can edit the information in `Application.xml`, if necessary, but each time you run the metadata creation operation, the metadata for the application is replaced if it already existed.

The template sets the value for the Service attribute of the ApplicationUri element using the current value of the Context Root for the project. This value can also be found in project’s properties, on the Web Project Settings page. The Service value must match the name of the project.

The file is eventually transformed into the required deployment files and then those files are used to deploy the metadata.

## build.xml

This file is an Ant build file that supports the launch file operations.

## destination\_omr.properties

This file contains properties that specify the destination metadata server when copying the application metadata. The values in this file are placeholders that must be manually entered once a destination metadata server has been chosen. This file is not overwritten by repeated additions of the metadata template.

The `copy.can.replace.metadata` property controls whether existing metadata in the destination SAS Metadata Server is deleted before the copy. If false (the default), then the copy is not performed if metadata already exists in the destination metadata server for that application.

**omr.properties**

This file contains properties that specify the destination metadata server when creating and deleting the application metadata, and the source metadata server when copying the application metadata. The values in this file are determined by the SAS BI Server Profile that you selected when adding the template.

---

## Copying the Application Metadata

When moving to a production environment, copy the application metadata from the SAS Metadata Server where it was created to a destination metadata server by following these steps:

1. Enter the information for the destination metadata server in `\metadata\destination_omr.properties`.

The `copy.can.replace.metadata` property controls whether existing metadata in the destination SAS Metadata Server is deleted before the copy. If false (the default), then the copy is not performed if metadata already exists in the destination metadata server for that application.

2. Right-click the *projectName Copy Metadata.launch* file, and select **Run As** ⇒ *projectName Copy Metadata.launch*.
3. Verify that BUILD SUCCESSFUL appears at the end of the output logged to the Console.

The copy operation extracts application metadata directly from the source metadata server and deploys it to the destination metadata server. No values in `Application.xml` or in the files derived from `Application.xml` when the application metadata was created are accessed. Consequently, changes made to the metadata using the SAS Management Console are included in the copy operation.

---

## Files Added by the Stored Process Java Client Template

**Introduction**

When you add the “Java client for executing a SAS Stored Process” template to a SAS Java Project, the following files are added to the source folder for the project.

**StoredProcessDriver.java**

This class is an example of a Java console application client for executing a SAS Stored Process and writing the stored process results (and optionally the SAS Log) to a file. The default name of the class is `StoredProcessDriver`, but the actual name is derived from the class name for the client that was entered in the wizard.

**StoredProcessConnection.java**

A `StoredProcessConnection` contains the information needed to use SAS Foundation Services, a SAS Metadata repository, and the SAS Stored Process Service for executing SAS Stored Processes. The class initializes and connects to SAS Foundation Services in preparation for executing SAS stored processes, and binds a `StoredProcessFacade` with the `StoredProcessDriver`.

SAS does not support customer implementations or extension of this class. The `StoredProcessConnection` generated for a SAS Java Project cannot be used in a SAS Web Application Project.

**StoredProcessFacade.java**

A `StoredProcessFacade` represents a SAS Stored Process and the results of its execution after the stored process has been executed. The `StoredProcessFacade` class wraps two interfaces from the SAS Stored Process Service:

`com.sas.services.storedprocess.StoredProcess2Interface` and  
`com.sas.services.storedprocess.Execution2Interface`.

This class can supply input parameters for the SAS Stored Process. Output parameter results or streaming results can be retrieved after the stored process is executed. The `StoredProcessFacade` provides other API methods for common operations, including the ability to access the underlying `StoredProcess2Interface` and `Execution2Interface` objects.

For more information, see the SAS Foundation Services Javadoc contained in the SAS BI API Documentation on the SAS AppDev Studio Developer's Site (<http://support.sas.com/rnd/appdev/>), in particular the package summary for `com.sas.services.storedprocess`.

SAS does not support customer implementations or extension of this class. The `StoredProcessFacade` generated for a SAS Java Project cannot be used in a SAS Web Application Project.

**StoredProcessDriver.properties**

This file contains information about the SAS Metadata Server, and specifies log and output filenames for the results of executing the SAS Stored Process. These values can be changed so that the generated client can use a different SAS Metadata Server. The default name of this file is `StoredProcessDriver`, but the actual name is derived from the class name for the client that was entered in the wizard.

**jaas.config**

This is the JAAS configuration file that is required by the Local Platform Services that are used in the Java application. Although the comment at the top of the file might claim the file is for a SAS Web Application, the file is also used with SAS Java Applications.

**launchParameters.txt**

This file provides an example of the Java System property that is needed to specify the JAAS configuration file that the Local Platform Services require. This Java System



property must be included in the command line or launch configuration used to run the stored process Java application.

---

## Files Added by the Stored Process Servlet Template

### *Introduction*

When you add the SAS Stored Process Servlet template to a SAS Web Application Project, Java files are added to the source folder for the project.

### ***StoredProcessDriverServlet.java***

The `StoredProcessDriverServlet` is an example of a servlet implementation for executing a SAS Stored Process, displaying the results in the browser, and optionally writing the contents of the SAS Log to the `ServletContext`. The default Example Base Name for a SAS Stored Process servlet is `StoredProcessDriver`. The actual name of the class is derived from the Example Base Name that was entered in the wizard.

### ***StoredProcessConnection.java***

A `StoredProcessConnection` contains the information needed to use SAS Foundation Services, a SAS Metadata repository, and the SAS Stored Process Service for executing SAS Stored Processes. The class binds a `StoredProcessFacade` with the `StoredProcessDriverServlet`.

SAS does not support customer implementations or extension of this class. The `StoredProcessConnection` generated for a SAS Web Application Project cannot be used in a SAS Java Project.

### ***StoredProcessFacade.java***

A `StoredProcessFacade` represents a SAS Stored Process and the results of its execution after the stored process has been executed. The `StoredProcessFacade` class wraps two interfaces from the SAS Stored Process Service:

`com.sas.services.storedprocess.StoredProcess2Interface` and  
`com.sas.services.storedprocess.Execution2Interface`.

This class can supply input parameters for the SAS Stored Process. Output parameter results or streaming results can be retrieved after the stored process is executed. The `StoredProcessFacade` provides other API methods for common operations, including the ability to access the underlying `StoredProcess2Interface` and `Execution2Interface` objects.

For more information, see the SAS Foundation Services Javadoc contained in the SAS BI API Documentation on the SAS AppDev Studio Developer's Site (<http://support.sas.com/rnd/appdev/>), in particular the package summary for `com.sas.services.storedprocess`.

SAS does not support customer implementations or extension of this class. The `StoredProcessFacade` generated for a SAS Web Application Project cannot be used in a SAS Java Project.

---

## Tomcat Configuration Details

### Tomcat Support

SAS AppDev Studio 4.4 continues to support Tomcat as a servlet container for testing SAS web applications. But because Tomcat is not officially supported by SAS 9.4, Tomcat is recommended only for initial functional testing.

Final acceptance testing should be done with an installed SAS Web Application Server. Because the SAS Web Application Server is based on a version of the VMware vFabric tc Server that is using Tomcat 7, only Tomcat 7 is supported by SAS AppDev Studio when testing. For compatibility, Tomcat 7.0.30 or later is recommended.

### Configuring a Tomcat Server

If you have multiple SAS 9.4 BI installations that you want to test against, you can use the **Configure for SAS AppDev Studio Use** menu command on the pop-up menu for Tomcat 7 servers in the Servers view. The command can add, update, or remove the configuration changes required for a Tomcat 7 server to run against the BI installation identified by the chosen SAS BI Server Profile. To configure a Tomcat 7 server, follow these steps:

1. In the Servers view, right-click the Tomcat 7 server that you want to configure and select **Configure for SAS AppDev Studio Use**.
2. In the dialog box, select the BI Server Profile that you want the server to run against, or check the Remove check box if you want to remove prior configuration changes.

The **OK** button is enabled only when a BI Server Profile is selected.

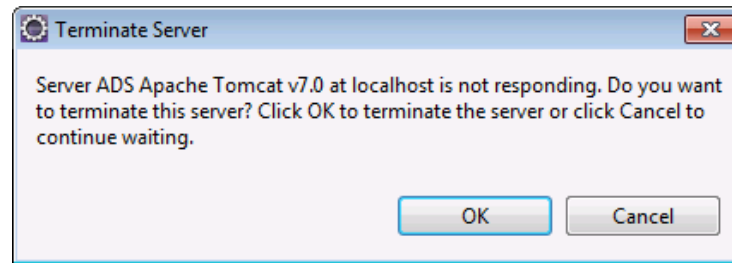
3. Click **OK**.

If you configure additional Tomcat 7 servers, you need to update the ports that they use to avoid conflicts. You should also lengthen the start-up time-out from the default of 45 seconds as this can be insufficient time to start even a single SAS web application on some systems. A time-out of at least 120 seconds is recommended. The actual time-out value that you need depend on your hardware and how many SAS web applications you are attempting to start.

Enabling Java security is neither required nor recommended because of performance issues, and is not supported.

### Tomcat Shutdown Issue

Tomcat can fail to shutdown. The problem is that when Tomcat is stopped, it waits indefinitely for all deployed web applications to fully clean up before exiting. There is a known issue with stopping deployed SAS Web Application projects while the Web Infrastructure Platform middle tier remains running, preventing the SAS Web Application project from fully cleaning up. When Tomcat fails to shutdown within the stop time-out, a dialog box asks if you want to terminate the process. Terminating Tomcat from this dialog box is the only way stop the server in this situation.



The Restart command does not handle the time-out issue well, and leaves the Tomcat process in a permanent “stopping” state without ever displaying the dialog box that asks you if you want to terminate the server. Further, a new Tomcat process is launched while the old Tomcat process is running. The new process runs normally because the old process was able to complete enough of the shutdown to free the ports that it was using. However, the old process continues to run and occupy nearly the same amount of memory that it was using during normal operation. The only way to stop an old Tomcat process is to either restart Eclipse or manually kill the old Tomcat process.

Instead of using the Restart command from the **Servers** tab or when you confirm a restart when using **Run As** ⇒ **Run on Server**, you should separately Stop and then Start the server.

Although some servers might offer you the ability to omit restarting, when a class on the server changes, you should stop and then start the server so that the updated files are picked up by the server. Doing this also avoids potential memory leaks.

---

## Deployment and Authentication

The SAS Web Infrastructure Platform facet manages both the Web Infrastructure Platform Configuration and the support JAR files for a SAS web application. Handling both enables the facet to keep the configuration features synchronized with the JAR files that support those features.

The SAS Web Infrastructure Platform configuration consists of static files found in `\WEB-INF\spring-config, context-param` declarations, and filter and servlet declarations and mappings. There is nothing in the configuration that binds the web application to a specific SAS BI installation. This means that the web application can be deployed, unchanged, to any SAS 9.4 BI installation, provided that the application metadata for the web application and any resources the web application depends on are present in that BI installation.



## Chapter 7

# Exporting Projects

---

Exporting Java and SAS Java Projects as a Set of JAR Files .....	45
Exporting a SAS Web Application Project as a WAR File .....	46
Exporting a Project Using a Deployment Descriptor File .....	47

---

## Exporting Java and SAS Java Projects as a Set of JAR Files

The only projects that can be exported as a JAR file or set of JAR files are Java and SAS Java Projects that have a class with a `public static void main(String[])`.

To export a SAS Java Project, follow these steps:

1. Select **File** ⇒ **Export**.
2. Select **SAS Java Project** as the export destination, and then click **Next**.
3. Select the project to deploy.
 

All of the projects available in the current workspace are listed. Invalid projects are indicated when selected.
4. Choose the resources to include in the output JAR file.
 

Compiled code is not included in the tree. Items that are on the project source path are checked by default.
5. Select the **Include Source Code** check box if you want the project source code to be included in the exported JAR file.
6. Specify the **Deployment directory**.
 

The JAR file containing the project code and other necessary files will be placed in this directory.
7. Specify the **Jar Name** and then click **Next**.
8. Select the **Use SAS Repository** check box if you want to create a JAR file that runs against a copy of the SAS Versioned Jar Repository that is already on the target machine.

If you do not deploy against a SAS Versioned Jar Repository, then all the required JAR files from the project and any dependencies from the SAS Versioned Jar

Repository are copied into the deployment directory. You can then copy all the JAR files to a deployment location if necessary.

If you do deploy against a SAS Versioned Jar Repository, the project contents, a picklist called **SASRepositoryConfig**, and a script (in batch and shell versions) are copied to the deployment directory. The scripts use as their base name the name of the exported JAR filename. You must either edit the script and specify the location of the SAS Versioned Jar Repository on the target machine, or pass the location as a command-line argument. For example:

```
exportedJarName.sh repositoryLocation
```

The script executes the application using the SAS launcher to load the JAR file dependencies from the target machine's SAS Versioned Jar Repository.

9. Select the **Main class** that starts your application. The main class must reside in your project.
10. Click **Next**.
11. Select the **Save Deployment Descriptor in Project** check box if you want to capture the current export settings.

If you choose this option, specify the **Descriptor Location**, relative to the project. The filename must end in **.depdesc**. For more information, see [“Exporting a Project Using a Deployment Descriptor File” on page 47](#).

12. In the **Manifest Location** field, specify the path to the manifest file relative to the project.

The manifest file can be in any directory and with any filename. However, when added to the exported JAR file, the manifest information is placed in the file **META-INF\MANIFEST.MF**.

If the specified manifest file already exists, the export overwrites any existing properties in the file that conflict with the properties needed to run the application properly. The overwriting occurs at the level of individual properties; the entire file is not overwritten.

13. Click **Next** and review the export settings, and then click **Finish**.

The project is exported to the specified deployment directory. Depending on the options that you chose, you might need to take additional configuration steps to run the project.

If the SAS Java project requires a JAAS configuration file, such as one that executes a Stored Process, then the JAAS configuration file must be manually copied to the deployment directory. In addition, the Java command in the startup batch script must be updated to include the Java System property shown in the **launchParameters.txt** file. Ensure the value of the property is modified to reference the copied JAAS configuration file.

---

## Exporting a SAS Web Application Project as a WAR File

To deploy a SAS Web Application Project to a server outside of Eclipse, you must create a WAR file for the web application contained in the project. This is accomplished using Eclipse's Export feature. Perform the following steps to create the WAR file:

1. Right-click the SAS Web Application project and select **Export** ⇒ **Web** ⇒ **WAR file**.
2. Specify the **Destination** for the WAR file.
3. Select a target server run time, or clear the **Optimize for specific server runtime** check box.
4. Click **Finish**.

The WAR file is copied to `C:\SAS\Config\Lev1\Web\WebAppServer\SASServer1_1\sas_webapps`.

The resulting WAR file can be deployed to a SAS Web Application Server in any BI Server installation that matches the version of the SAS facets that were used in the SAS Web Application project. The only requirement is that the application metadata for the web application and any resources the web application depends on are present in that BI Server installation. The connection information in the application metadata (the settings specified in the `ApplicationUri` element of the project's `Application.xml`) must match the location of the web application deployment.

Additional information about web application deployment is available on the SAS AppDev Studio Developers Site at <http://support.sas.com/rnd/appdev/>.

---

## Exporting a Project Using a Deployment Descriptor File

Deployment descriptor files contain the export settings for a project. You can save a descriptor file when you export a project. On subsequent exports, you can use the descriptor file to bypass the Export dialog box. You can have multiple descriptor files per project. Descriptor files must end with `.depdesc`.

To use a descriptor file to bypass the Export dialog box, expand the project in the Package Explorer or Navigator, right-click the descriptor file, and select **Export SAS Project**. The project is exported using the settings in the descriptor file.

To use a descriptor file to populate the Export dialog box with the settings in the file, right-click the file and select **Open SAS Export**.

To inspect or change the settings in a descriptor file, you can open the file as XML by double-clicking it, and editing values as needed. Unless otherwise specified, the first descriptor file listed is used to populate the Export dialog box.





## Chapter 8

# Managing the JAR Files in a Project

---

<b>The SAS Repository</b> .....	<b>49</b>
<b>Opening the SAS Repository Properties Editor</b> .....	<b>50</b>
<b>Identifying Dependent JAR Files</b> .....	<b>51</b>
<b>Removing JAR Files from the Classpath</b> .....	<b>52</b>
<b>Changing the Order of JAR Files in the Classpath</b> .....	<b>53</b>
<b>Adding New JAR Files to the Classpath</b> .....	<b>54</b>
<b>Adding Dependent JAR Files</b> .....	<b>55</b>
<b>Specifying the Current Versions of JAR Files</b> .....	<b>55</b>
<b>Specifying Other JAR File Versions</b> .....	<b>55</b>
<b>Removing Version Restrictions</b> .....	<b>56</b>
<b>Reporting the Classpath JAR Files</b> .....	<b>56</b>
<b>Reporting JAR File Relationships</b> .....	<b>57</b>
<b>Finding a Class in a JAR File</b> .....	<b>58</b>
<b>Changing Default Classes for SAS Java Projects</b> .....	<b>59</b>

---

## The SAS Repository

There are two pieces to managing the JAR files that a project requires: the SAS Versioned Jar Repository, and the project's SAS Repository.

The SAS Versioned Jar Repository is a common storage location for all the JAR files that are supplied by the installed SAS products. This includes some third-party JAR files. This repository is incrementally updated as SAS products are installed.

A project's SAS Repository is a configurable classpath container that is added to the Java build path of all SAS projects to provide access to the JAR files in the SAS Versioned Jar Repository. The SAS Repository determines which SAS JAR files need to be included in a project's build path. It can also determine any JAR file dependencies and automatically include those dependencies in the project's Java build path. This ability means that you need to specify only the primary JAR files that a SAS project requires, the dependencies are handled for you.

When necessary, you can bypass the SAS Repository automation and specify exactly which JAR files, and which versions of each JAR file, to include. This flexibility enables you to manage JAR file dependencies on a per-project basis without disrupting past development, and contributes to a smaller distribution footprint.

If you want to add to the project a JAR file that is not available in the SAS Versioned Jar Repository, open the project properties, select **Java Build Path**, and use one of the mechanisms that does not involve editing the SAS Repository on the **Libraries** tab. You are responsible for adding any JAR file dependencies.

---

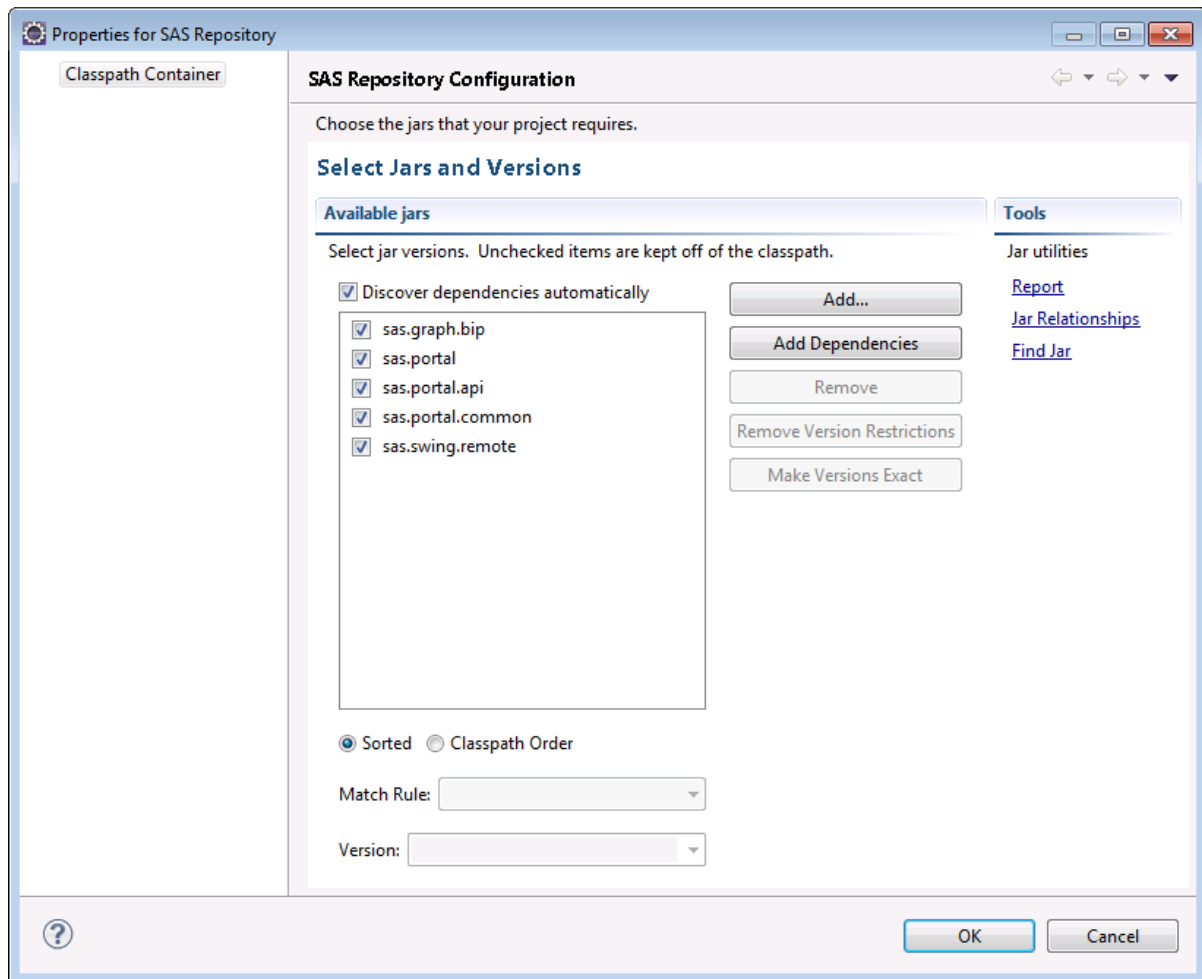
## Opening the SAS Repository Properties Editor

Access to the SAS Repository properties depends on the project type and the current perspective. When in the Java perspective, all project types (SAS Java projects and SAS Web Application projects) display the SAS Repository at the root of the project. To access the Properties for SAS Repository dialog box, follow these steps:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select **Properties**.

The Properties for SAS Repository dialog box appears.

When in the Java EE perspective, the SAS Repository is located at the top level of SAS Java Projects, and in `\sourceFolder\Libraries\` for SAS Web Applications projects.



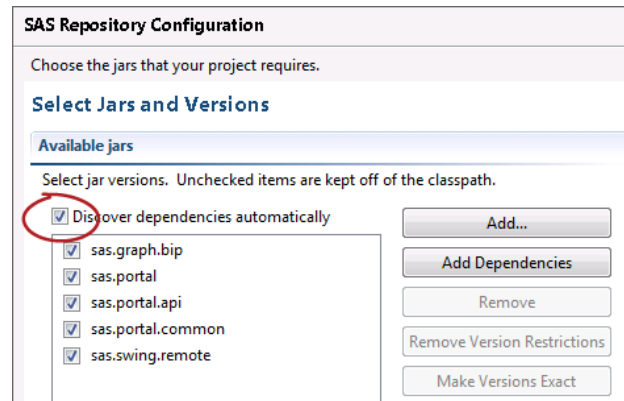
You can also access the SAS Repository as a library:

1. Right-click the project and select **Properties**.
2. Select **Java Build Path**, the **Libraries** tab, then **SAS Repository**, and then click **Edit**.

---

## Identifying Dependent JAR Files

To automatically add the dependencies of all checked JAR files to the classpath, select the **Discover dependencies automatically** check box. With this option selected, the classpath is automatically updated if there are changes in the SAS Versioned Jar Repository that affect the JAR files required by your project.



If you click the **Add Dependencies** button, all the JAR file dependencies for the JAR files that are checked in the **Available jars** list are added to the Available jars list. This enables you to specify a version for each of the dependent JAR files, if needed.

If the **Discover dependencies automatically** check box is not selected, only the checked JAR files are added to the classpath. To exclude a JAR file from the classpath, uncheck it. A JAR file that is not checked is not passed to the compiler or run-time engine.

You should use the **Discover dependencies automatically** option only if you are comfortable with JAR files being automatically added to the classpath, knowing that updates to the SAS Versioned Jar Repository might change which JAR files are required for your project.

For a SAS Web Application project, the **Discover dependencies automatically** option is unchecked by default. Using this option is not recommended for SAS Web Applications because the JAR files in the project must be compatible with the static content provided by the SAS Web Infrastructure Platform. The necessary JAR files are specified internally.

---

## Removing JAR Files from the Classpath

To remove one or more JAR files from your project's classpath, follow these steps:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select **Properties**.

The Properties for SAS Repository dialog box appears.

3. Select the JAR file or files to remove and then click the **Remove** button.

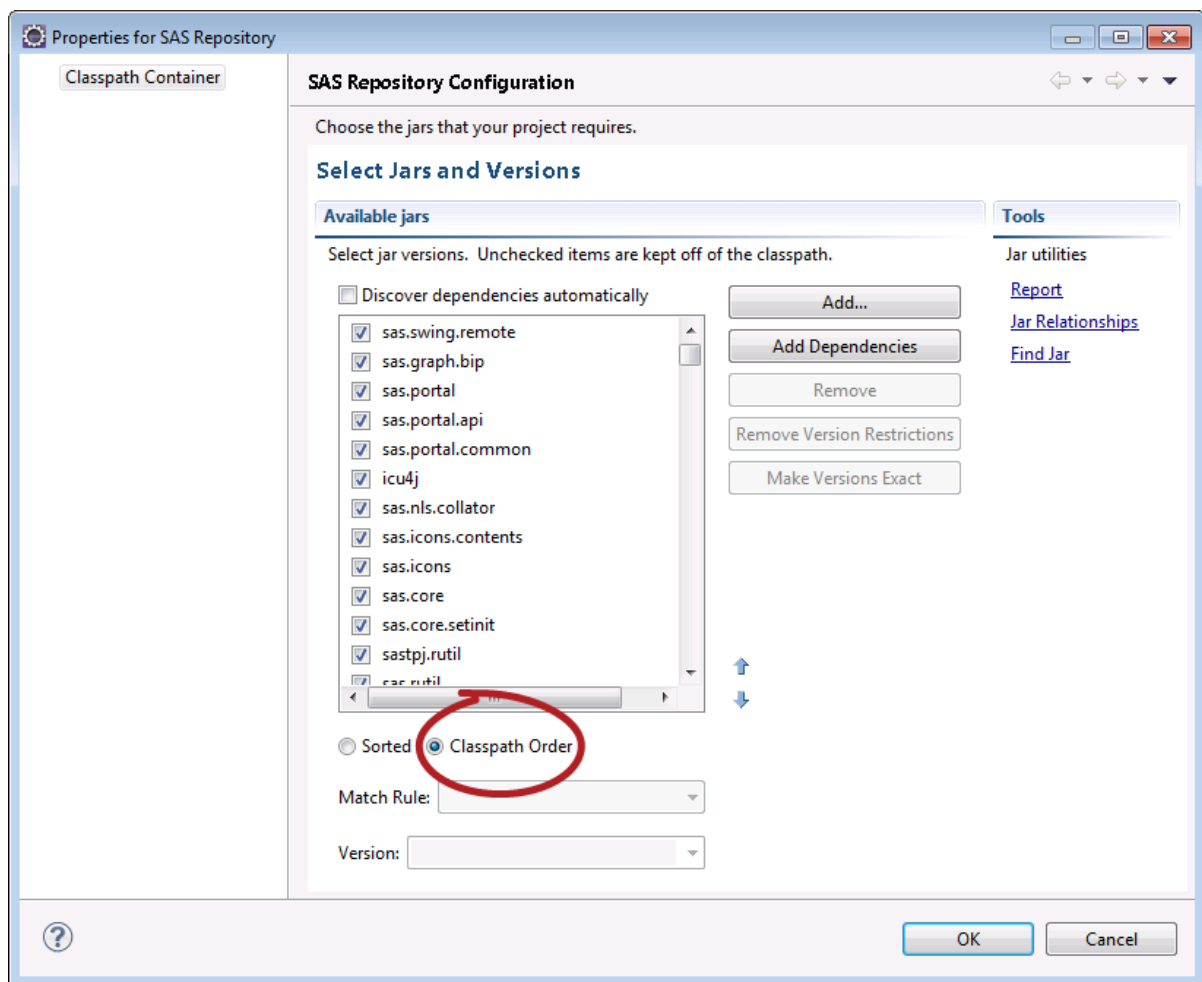
Note that using the Organize SAS Imports action adds a removed JAR file to the classpath (see [“Using the Organize SAS Imports Action” on page 64](#) ). In contrast, when you uncheck a JAR file in the **Available jars** list (see [“Identifying Dependent JAR Files” on page 51](#) ), the Organize SAS Imports command does not add the unchecked JAR file back to the classpath.

## Changing the Order of JAR Files in the Classpath

The order of the JAR files in the **Available jars** list determines the order that the JAR files appear in the classpath. The higher a JAR file is listed in the interface, the closer to the front of the classpath it appears.

To change the order of the JAR files, follow these steps:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select **Properties**.  
The **Properties for SAS Repository** dialog box appears.
3. Select **Classpath Order**.



4. Select a JAR file in the **Available jars** list and click the up or down arrow.

---

## Adding New JAR Files to the Classpath

To add new JAR files to the classpath, follow these steps:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select **Properties**.

The Properties for SAS Repository dialog box appears.

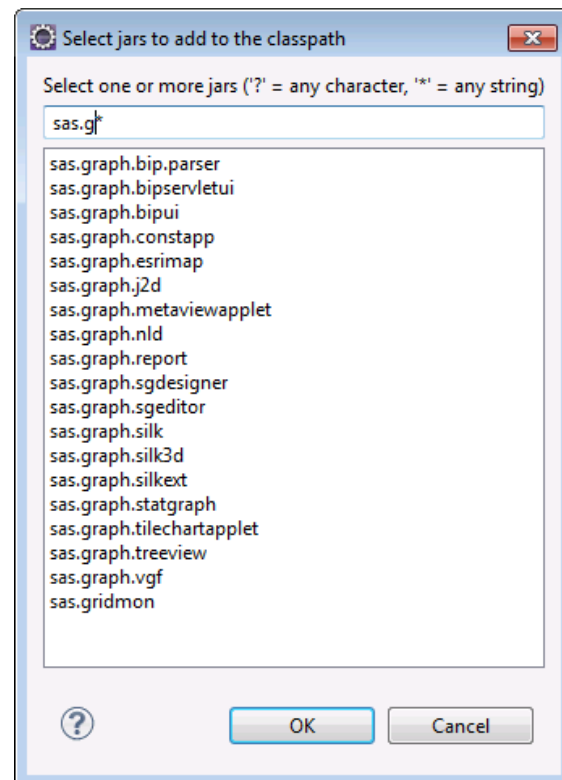
3. Click **Add**.

A dialog box appears that lists all the JAR files that are available in the SAS Versioned Jar Repository.

4. Select one or more JAR files and click **OK**.

To narrow the list of available JAR files, enter a substring in the text field at the top of the dialog box. Use the regular expression characters provided.

A second dialog box might appear listing JAR files that are required by the JAR files you explicitly selected. Click **Yes** to add those JAR files, or **No** to add only the JAR files that you explicitly selected.



---

## Adding Dependent JAR Files

The **Add Dependencies** button determines whether there are missing JAR files that the project's current JAR files require. This is useful when the **Discover dependencies automatically** is not selected, and you need to find missing JAR files, or **Discover dependencies automatically** is selected but you need to find a dependency so that you can override the version. Follow these steps to identify dependent JAR files:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select **Properties**.

The Properties for SAS Repository dialog box appears.

3. Click **Add Dependencies**.

If a missing JAR file is detected, the latest version of the JAR file in the repository is appended to the list of selected JAR files.

---

## Specifying the Current Versions of JAR Files

Specifying the current version of a JAR file (as opposed to the most recent) is useful for removing any ambiguity about which version of a JAR file is specified on the classpath. To specify that the current version of a JAR file should always be used, and you do not want to use a new version when it becomes available, follow these steps:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select Properties.

The Properties for SAS Repository dialog box appears.

3. Select the JAR file or files and click **Make Versions Exact**.

For each selected JAR file, the **Match Rule** is changed to **Exact**, and the **Version** is changed to the version of the JAR file currently in use.

---

## Specifying Other JAR File Versions

You can control which JAR files get used with varying degrees of specificity. Eclipse JAR file versions follow the pattern **Major\_version.Minor\_version.Micro\_version.tag**, where **tag** represents characters used to differentiate versions. If more than one version of a JAR file that matches your specifications is found in the repository, the latest version is used. You can have only one version of a JAR file in your classpath.

To specify a version of a JAR file that is not the latest version, follow these steps:

1. Expand the project in the Package Explorer.

2. Right-click **SAS Repository** and select **Properties**.

The Properties for SAS Repository dialog box appears.

3. Select the JAR file or files to which you want to apply a versioning rule.
4. Select a **Match Rule**, as follows:
  - **Latest** - The highest version of the JAR file.
  - **At Least** - The highest version of the JAR file that is at least the version selected in the Version list.
  - **Exact** - The JAR file specified in the Version list. No higher or lower version can be substituted. Only for an Exact match is the tag of the version tested.
  - **Up To** - The highest version of the JAR file up to and including the version in the Version list.
  - **Latest Within Major Version** - The JAR file with the highest version but with the same major version specified in the Version list.
  - **Latest Within Minor Version** - The JAR file with the highest version but with the same major and minor versions specified in the Version list.
  - **Latest Within Micro Version** - The JAR file with the highest version but with the same major and minor and micro versions specified in the Version list.
5. Select the **Version**.

If the currently selected **Match Rule** requires version information, select or enter the version number in the **Version** combo box, which is populated with all of the version numbers that are currently available in the SAS Versioned Jar Repository. Note that the JAR files themselves might have their own version requirements for dependent JAR files.

---

## Removing Version Restrictions

To remove version restrictions on a JAR file and get the latest version, follow these steps:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select Properties.
 

The Properties for SAS Repository dialog box appears.
3. Select the JAR file and click **Remove Version Restrictions**.
 

The **Match Rule** is changed to **Latest**, and the **Version** is cleared.

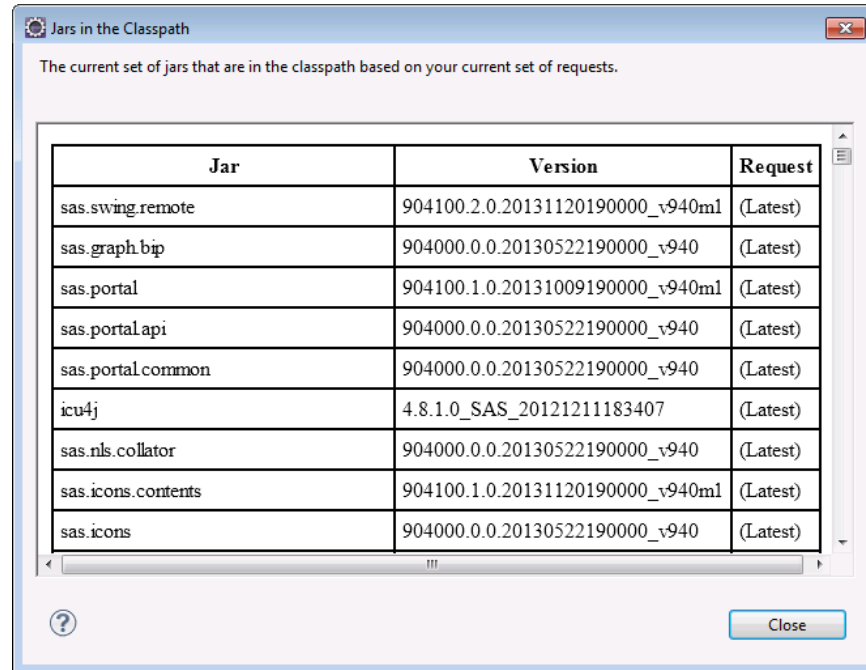
---

## Reporting the Classpath JAR Files

To generate a report of all the JAR files that will be used on the classpath, including the reason that a specific version of a JAR file version was included, follow these steps:



1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select **Properties**.  
The Properties for SAS Repository dialog box appears.
3. Click **Report**.



Jar	Version	Request
sas.swing.remote	904100.2.0.20131120190000_v940m1	(Latest)
sas.graph.bip	904000.0.0.20130522190000_v940	(Latest)
sas.portal	904100.1.0.20131009190000_v940m1	(Latest)
sas.portal.api	904000.0.0.20130522190000_v940	(Latest)
sas.portal.common	904000.0.0.20130522190000_v940	(Latest)
icu4j	4.8.1.0_SAS_20121211183407	(Latest)
sas.nls.collator	904000.0.0.20130522190000_v940	(Latest)
sas.icons.contents	904100.1.0.20131120190000_v940m1	(Latest)
sas.icons	904000.0.0.20130522190000_v940	(Latest)

---

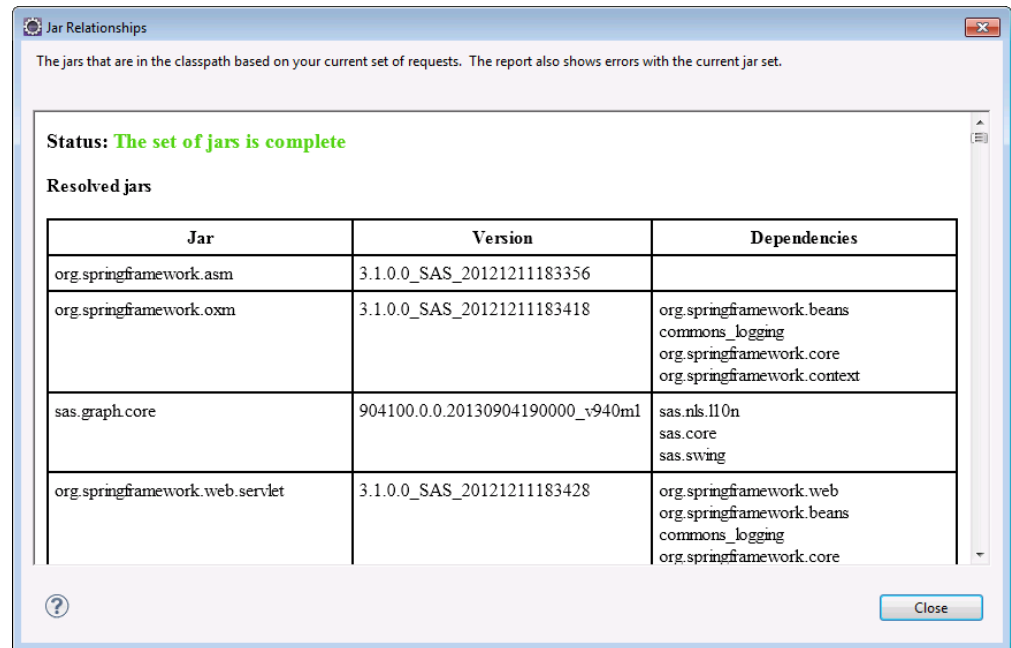
## Reporting JAR File Relationships

To generate a report showing the relationship between the JAR files in the project's SAS Repository, and which JAR files are missing or have constraints that cannot be satisfied, follow these steps:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select **Properties**.  
The Properties for SAS Repository dialog box appears.
3. Click **Jar Relationships**.

The resulting report can include the following sections:

- **Resolved jars** – lists the name and version of each JAR file that was included and also the names of dependent JAR files that the included JAR file requires.
- **Missing jars** – lists JAR files that are required by the JAR files that are currently checked in the Available Jars list, but are not currently included. The JAR file IDs are listed, along with the name of the JAR file that directly requested them.
- **Unresolved jars** – Displays the version request string and the likely error.



## Finding a Class in a JAR File

To find a class in your project's JAR files, and then add that JAR file to the SAS Repository and the classpath, follow these steps:

1. Expand the project in the Package Explorer.
2. Right-click **SAS Repository** and select **Properties**.

The Properties for SAS Repository dialog box appears.

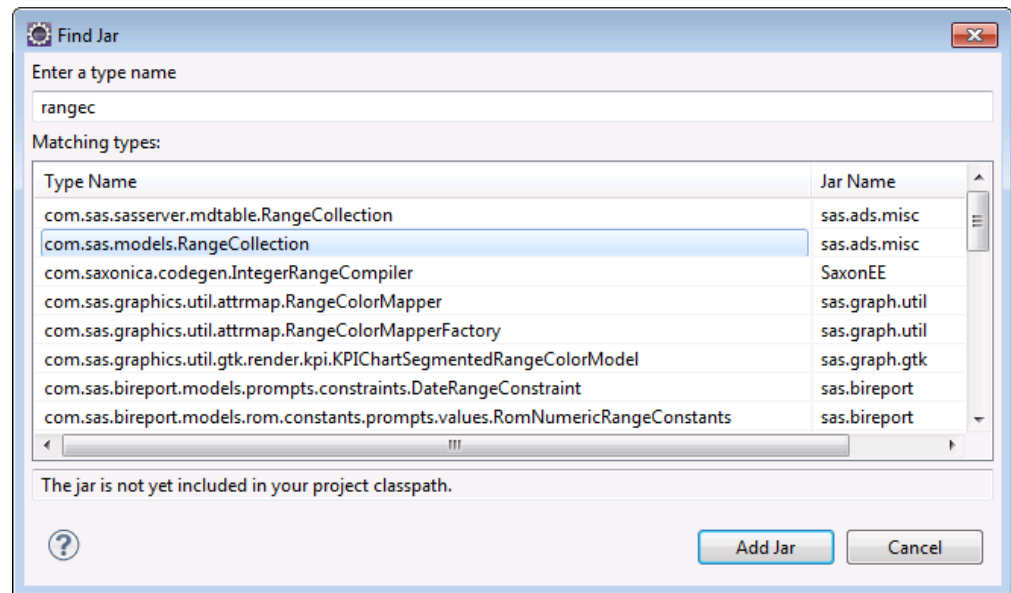
3. Click **Find Jar**.
4. Enter the local name of a class (not the package).

The search is case insensitive and returns all classes in the project's SAS Repository that contain the search string.

5. Select the class that you want from the list.

If the **Add Jar** button is enabled, then the JAR file is not in the project's SAS Repository. Click **Add Jar** to add it.

If the **Add Jar** button is dimmed, then the JAR file is already in the project's SAS Repository.



## Changing Default Classes for SAS Java Projects

By default, the classpath for a new SAS Java Project includes `sas.swing.remote`, `sas.graph.bip`, and all their dependent JAR files. To change the default classes that are added to a SAS Java project, follow these steps:

1. Select **Window** ⇌ **Preferences**.
2. Expand **SAS AppDev Studio**, and select **SAS Project dependencies**.
3. Change the included JAR files just as you would for a project.
4. Click **OK**.

New projects will use the new default JAR files. Existing projects are not affected.



## Chapter 9

# Using the SAS Editor Extensions

---

<b>Introduction to SAS Editor Extensions</b> .....	<b>61</b>
<b>Accessing SAS Component API Documentation</b> .....	<b>62</b>
Introduction .....	62
Eclipse Help System .....	62
Context-Sensitive Javadoc in an Eclipse Window (F1) .....	62
Context-Sensitive Javadoc in an External Browser (Shift+F2) .....	62
Configuring External Javadoc for Use within Eclipse .....	62
<b>Adding Missing Import Statements</b> .....	<b>63</b>
Introduction .....	63
Using a SAS Import from Repository Quick Fix .....	63
Using the Organize SAS Imports Action .....	64
<b>Attaching a SAS Model to a Viewer</b> .....	<b>64</b>
Overview of Model/Viewer Connections .....	64
Using the SAS Model/Viewer Connection Quick Assist .....	65
<b>SAS Snippets</b> .....	<b>67</b>
Snippets Introduction .....	67
Displaying the SAS Snippets .....	67
Inserting a SAS Snippet .....	67

---

## Introduction to SAS Editor Extensions

The SAS AppDev Studio 4.4 Eclipse plug-ins add the following functionality to the default Eclipse setup:

- access to the SAS API documentation from code
- identification and addition of missing import statements
- assistance with model/view attachments
- SAS snippets

---

## Accessing SAS Component API Documentation

### Introduction

You can access the SAS Component API Javadoc from within Eclipse via the following:

- the Eclipse Help system
- context-sensitive Help in an Eclipse window (F1)
- context-sensitive Help in an external browser (Shift+F2)

### Eclipse Help System

To browse the SAS Component API documentation, follow these steps:

1. Select **Help** ⇒ **Help Contents**.
2. Expand **SAS Components Guide**.
3. Expand **Reference**.

### Context-Sensitive Javadoc in an Eclipse Window (F1)

To view the Javadoc for a class in an Eclipse window, place the text insertion point on a class name in the source code and press F1. A Help view appears in the workbench. The first link in the Help view is a link to the Javadoc, if Javadoc is available for the class.

The link in the Help view is context sensitive, and changes if you reposition the insertion point to a different class.

### Context-Sensitive Javadoc in an External Browser (Shift+F2)

If the class is on the build path of the project, you can view the Javadoc for the class in an external browser by placing the insertion point on the class name in the source code and then pressing Shift+F2.

### Configuring External Javadoc for Use within Eclipse

To specify more current documentation for a JAR file, or to add a reference to Javadoc for a third-party JAR file, follow these steps:

1. Open a project that uses the JAR file.
2. In the Package Explorer, expand **SAS Repository**.
3. Right-click on the JAR file with which you want to associate Javadoc and select **Properties**.
4. Select **Javadoc Location** and then specify the location.

Javadoc for SAS Components is not displayed when you position your mouse pointer over a class name or method call in the source code because the Javadoc for tooltips is generated from source code, and the SAS source code is not distributed.

## Adding Missing Import Statements

### Introduction

When you declare or use a class that is in the classpath but is not yet imported into your Java program, Eclipse provides a Quick Fix that can add an import statement for that class to the top of the source file. Eclipse also provides an Organize Imports action that can add the necessary imports for a given source file. However, neither the Eclipse Quick Fix nor the Organize Imports action are aware of the SAS Versioned Jar Repository.

If you are using content from the SAS Versioned Jar Repository, use the SAS Import from Repository Quick Fix and the Organize SAS Imports action because both are aware of the SAS Versioned Jar Repository.

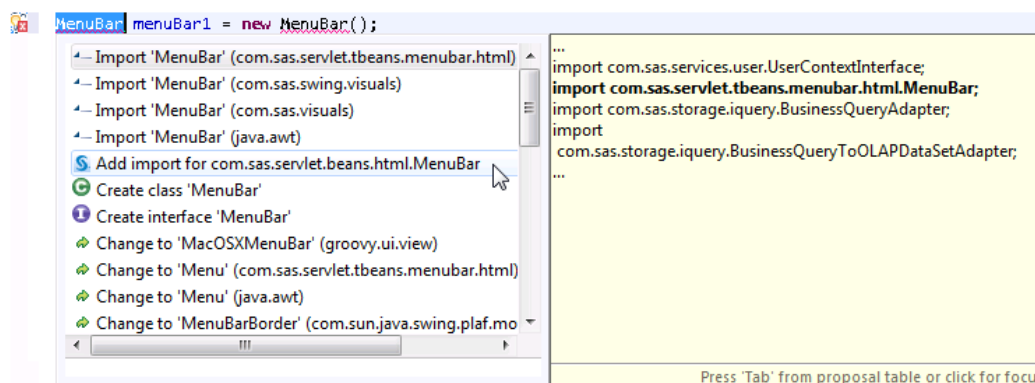
### Using a SAS Import from Repository Quick Fix

A **SAS Import from Repository Quick Fix** is displayed when there is a compilation error because a class used in the code cannot be resolved in the current source file.

If you accept the Quick Fix, the SAS Import from Repository Quick Fix adds the following items:

- the JAR file containing the class to the project classpath
- an import statement for the class to the current file.

If the name of the class (excluding the package) is included on the current project classpath, Eclipse provides a Quick Fix for each of the candidate class names. If there are additional candidate class names in the latest versions of one or more JAR files in the SAS Versioned Jar Repository, Quick Fixes are also provided for those class names.



To use one of the SAS Import from Repository Quick Fixes (marked with a SAS icon), double-click the Quick Fix or select the Quick Fix and press Enter. If you use a SAS Import from Repository Quick Fix and the Build Automatically option is enabled, the entire project is rebuilt because the classpath has changed.

## Using the Organize SAS Imports Action

In cases where there are multiple unknown types used in a source file, the Organize SAS Imports action can guide you through the process of resolving all of the unknown classes (both classes from the SAS Versioned Jar Repository and other classes) in the current source file. If you add any new JAR files to the project dependencies, the project must be rebuilt.

A Java file that cannot be found in a class exists in one of two states:

- the class is currently on the project classpath, but it has not been imported into the Java file. In this case, an import statement is added to the file and a build of that particular file is required.
- the class is not currently on the project classpath. In this case, the classpath must be modified to include the required jar file or files, and an import statement must be added to the file. A full build of the project is required because of the change to the classpath.

To start the Organize SAS Imports action, right-click in a Java file in the Java editor and select **Source** ⇒ **Organize SAS Imports**.

## Attaching a SAS Model to a Viewer

### Overview of Model/Viewer Connections

#### Introduction

SAS AppDev Studio 4.4 does not support attaching models to viewers using drag-and-drop, but there is an editor Quick Assist that facilitates model/viewer connections.

Note that the Quick Assist cannot determine whether you have already attached a model to a viewer. You can run the Quick Assist multiple times on the same viewer instantiation, each time attaching it to a different model. To determine which attachment is in effect, you must examine the code and identify the last attachment called.

#### Enabling the Quick Assist Icons

Eclipse uses a lightbulb icon in the editor gutter to notify you when a Quick Assist is available. By default, the Quick Assist icons are not displayed. To enable the Quick Assist icons, follow these steps:

1. Select **Window** ⇒ **Preferences**.
2. Expand **Java**.
3. Select the **Editor** item.
4. Select the **Light bulb for quick assists** check box.

#### Supported Viewers for Model/Viewer Connection Quick Assist

The Connection Quick Assist supports viewers that have a `setModel` method that takes as an argument one of the following types:

- `javax.swing.ComboBoxModel`



- `javax.swing.ListModel`
- `com.sas.table.StaticRowTemplateTableInterface`
- `com.sas.table.StaticTableInterface`
- `javax.swing.table.TableModel`
- `javax.swing.tree.TreeModel`

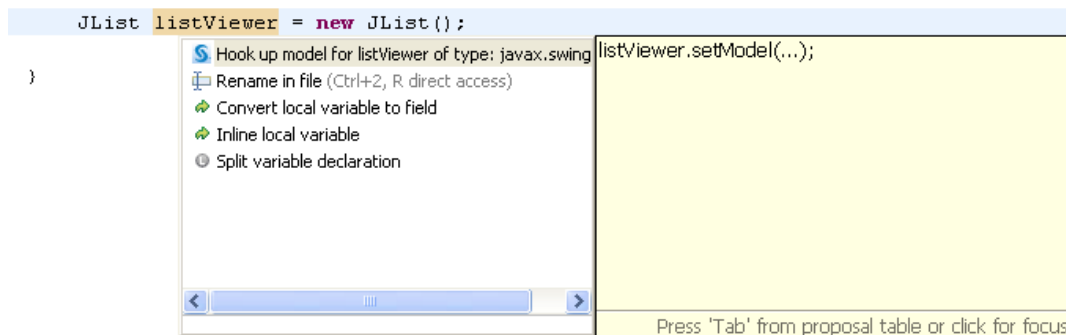
### Using the SAS Model/Viewer Connection Quick Assist

To use the SAS Model/Viewer Connection Quick Assist, follow these steps:

1. Place the cursor on a statement that contains a variable instantiation of a viewer type (for example, `javax.swing.JList`). The Quick Assist will not function if you place the cursor on a static method.
2. Press Ctrl+1.

From the list of Quick Assist choices, select **Hook up model**.

You can also access the connection Quick Assist by right-clicking and selecting **Source** ⇒ **Add SAS Attachments**.



3. Specify how you want to create and link a model to the viewer:

- **Use existing field**
- **Create new field**

If you selected **Use existing field**:

- a. Select the field to use.

Only the models declared as fields that are within the same scope as the cursor location and match the available model type(s) are listed. If there are no available fields, the **Use existing field** option is disabled.

- b. Click **Finish** to generate the code that creates the model and attaches it to the viewer.

If you selected **Create new field**:

- a. Select the model to attach to the viewer from the **Choose a class to declare** list.

When you expand an available class, the child nodes represent the classes in the current classpath that extend or implement the expanded class.

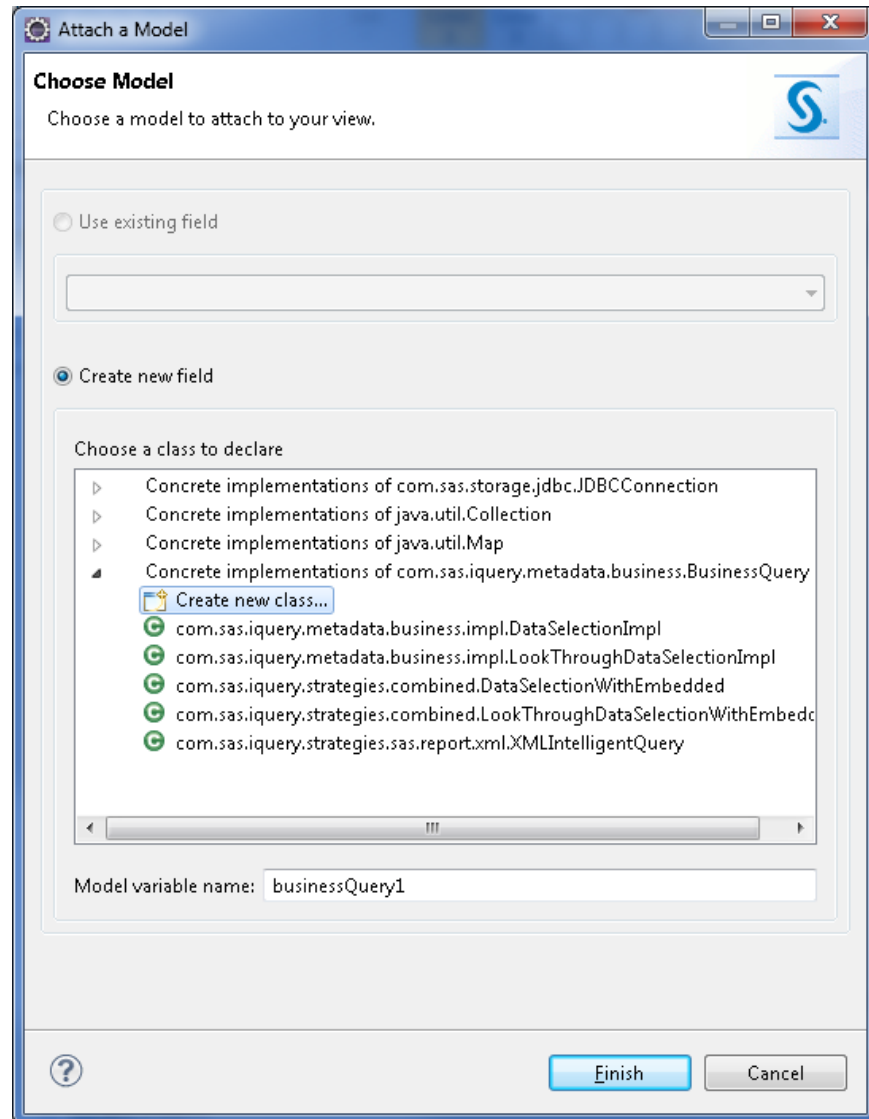
- b. Enter the **Model variable name**.

This variable name is used in the declaration in the source file. The variable is declared in the top-most enclosing class.

A default variable name is supplied, based on the currently selected class. If you modify this name, it will not change when you select a different class.

- c. Click **Finish** to add an initialization method to the source file with statements that instantiate the appropriate type for this variable (multiple fields are written with the same initialization method).

A secondary dialog box appears, giving you the opportunity to name and adjust other creation options for the new class before the initialization method is created.



---

## SAS Snippets

### *Snippets Introduction*

Snippets are pieces of code that perform some action for you. Typically, they enable you to configure a piece of code and then insert it, saving you from a lot of typing. Snippets can also perform actions for you, like searching a SAS Metadata server. Snippets are a robust form of the Eclipse Editor snippets.

### *Displaying the SAS Snippets*

To display the SAS snippets, follow these steps:

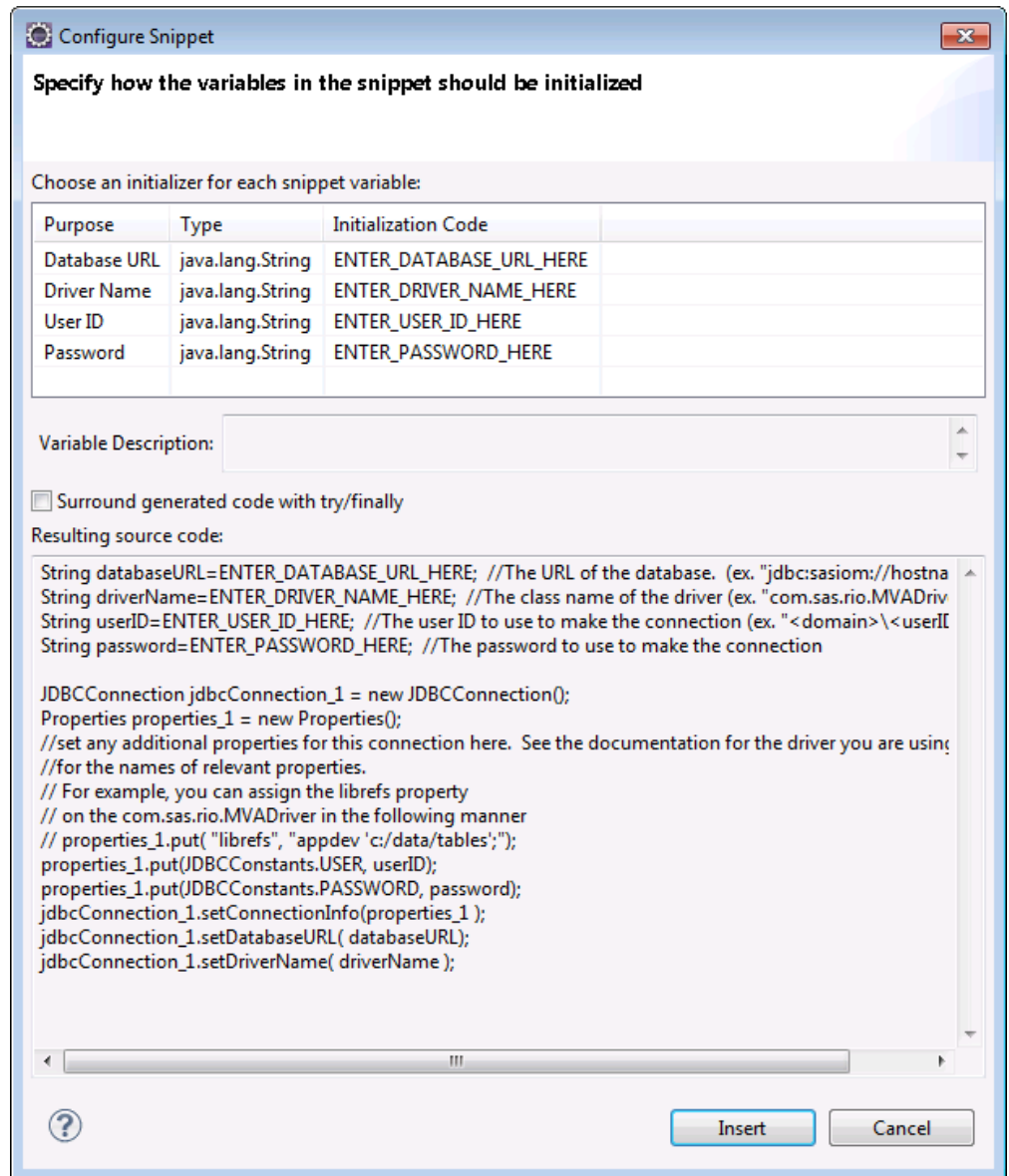
1. Select **Window** ⇒ **Show View** ⇒ **Other**.
2. Expand **General**.
3. Select **Snippets** and then click **OK**.
4. Inside the **Snippets** tab, click **SAS Snippets**.

### *Inserting a SAS Snippet*

To insert a snippet, follow these steps:

1. Place your cursor inside a non-static method body where you want to insert the snippet.
2. Double click on the snippet that you want to insert (or right-click and select **Insert**).
3. Configure the variables used in the snippet.

Variables in the snippet are either declared at the top of the snippet, or are replaced by variables that already exist in the code.



The **Initialization Code** field for each variable defines how the variable is assigned. The **Initialization Code** can be set one of three ways:

- An illegal value that will not compile so that you are forced to fix it (for example, **ENTER\_HOST\_NAME\_HERE**).
- Arbitrary text that you enter.

You can enter any text, method call, or other complex expression for the **Initialization Code**. If entering a string, include the appropriate quotation marks and escape characters.

- A variable from your code.

The list of values is populated with all of the variables in scope that are compatible with this variable type. If you select one of these variables, the declaration at the top of the snippet is removed and the variable that you have selected is used throughout the snippet.

4. Choose whether to **Surround generated code with try/finally**. This option encloses the generated code in a `try/finally` block. This is useful if you want to handle the declared exceptions in the current method.
5. Click **Insert**.

The snippet code is added at the cursor location and the necessary import statements are added to the top of your source file. Inserted type references are fully qualified if they would conflict with an existing import.



## Appendix 1

# Creating a SAS Web Application That Does Not Use the Web Infrastructure Platform

### Introduction

In AppDev Studio 4.4, a standard SAS Web Application Project contains both the “SAS Java Components” facet and the “SAS Web Infrastructure Platform” facet. The SAS Web Application Project wizard always adds both facets when creating a new SAS Web Application Project. Once added, neither SAS facet can be removed from the project. Consequently, the SAS Web Application Project wizard cannot be used to create a SAS Web Application Project that does not include the “SAS Web Infrastructure Platform” facet.

However, you can create a SAS Web Application Project without the Web Infrastructure Platform facet. Such a project deploys Local Services using "Local Platform Services," which is in all SAS 9.4 BI Server installations. You can choose a different Local Services deployment if you have one available.

Note that because the project does not use the Web Infrastructure Platform, it cannot integrate with the Logon Manager application.

### Create a Dynamic Web Project

Create a new SAS Web Application Project that contains only the SAS Java Components facet by following these steps:

1. From Eclipse, select **File** ⇒ **New** ⇒ **Project**.
2. Expand the **Web** folder.
3. Select **Dynamic Web Project** and click **Next**.
4. Specify a project name.  
This name is used as the context name. The name must not contain spaces.
5. Select the **Target Runtime** to match the server that you plan to use.  
The default target run time is ADS Apache Tomcat v7.0.
6. For the Dynamic Web Module version, select **2.4**.
7. For the Configuration, click **Modify**, select **SAS 9.4m1 Java Components Configuration**, and then click **OK**.

For AppDev Studio 4.4 that has not had a maintenance update applied, select **SAS 9.4 Java Components Configuration**.

## Add a Context Listener

The last step in creating a SAS Web Application Project that does not use the Web Infrastructure Platform is to add a context listener that deploys the Local Services.

However, if you are using one of the SAS Java Web Application example templates provided with AppDev Studio 4.4, a Local Services context listener is automatically added to the project on a successful build. In such a case, you must add a context listener only if you are changing which BI Server Profile or Local Services deployment the project is using.

To add or update the context listener, follow these steps:

1. From Eclipse, select **File** ⇒ **New** ⇒ **Other**.
2. Expand the **SAS AppDev Studio** folder.
3. Select **Add Template Content to Project**, and then click **Next**.
4. For the **Project**, select the host project that was created earlier.
5. Expand the **SAS Java Web Application** and then **SAS Foundation Services Support** folders.
6. Select **Context Listener For Local Services**.
7. Click **Next**.
8. Click **Next** to accept the default Template Configuration Parameters.
9. Select the BI Server Profile for the BI installation that you want to target. If you want to change the Local Services deployment from the current or default "Local Platform Services," ensure the SAS Metadata Server in the BI installation is running.
  - a. Click **Advanced**.
  - b. In the Local Services Deployment section, select **Other**.
  - c. Click **Browse** and select the service deployment.
  - d. Click **OK**.
10. If the SAS Metadata Server for the BI installation is running, you can click **Test Configuration** to verify that the service deployment can be read from the metadata server.
11. Click **Finish**.

## Adding a Stored Process Servlet to a Foundation Services Application

Before starting this process, ensure that if a Metadata Server Connection profile is currently open, it targets the BI installation whose metadata server contains the stored process that you want to add. If it does not, you must disconnect that Connection profile before starting this process. Switching Connection profiles (and hence, metadata server connections) within the wizard is not currently supported.

To add a Stored Process Servlet, follow these steps:

1. From Eclipse, select **File** ⇒ **New** ⇒ **Other**.
2. Expand the **SAS AppDev Studio** folder.



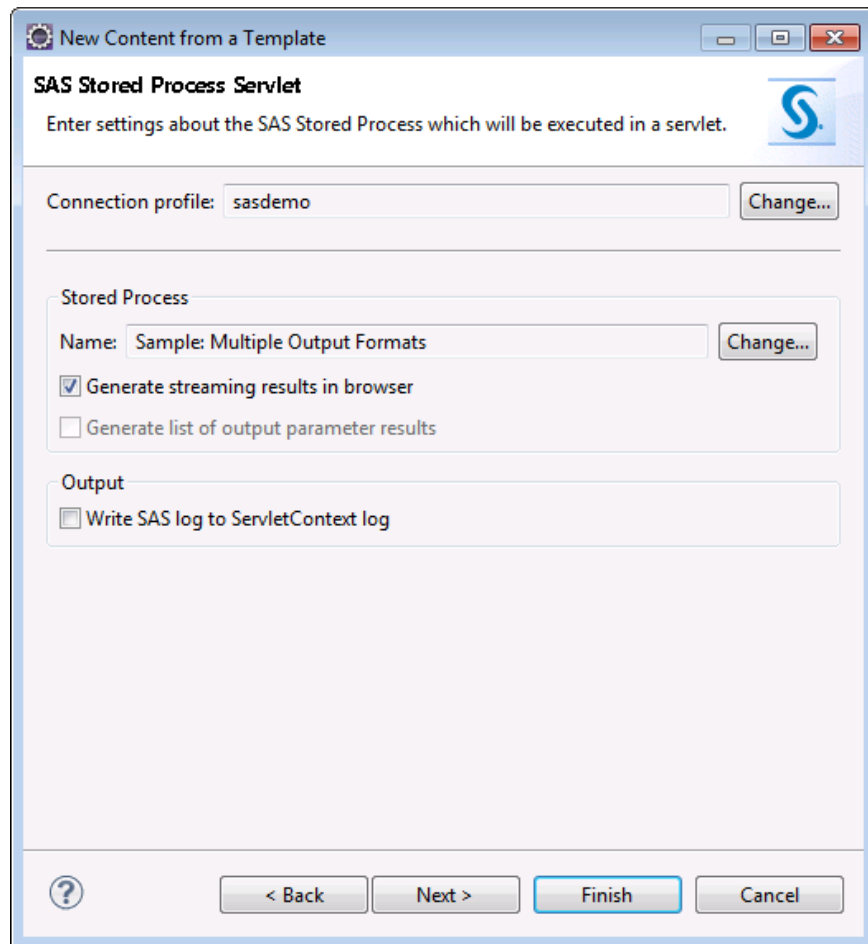
3. Select **Add Template Content to Project**, and then click **Next**.
4. For the **Project**, select the host project that was created earlier.
5. Expand the **SAS Java Web Application** and then **SAS Web Application Examples** folders.
6. Select **SAS Stored Process Servlet (uses SAS FS)**.
7. Click **Next**.
8. Click **Next** to accept the default Template Configuration Parameters.
9. Select the BI Server Profile for the BI installation that contains the stored process that you want to add.

If you want to change the Local Services deployment from the current or default "Local Platform Services," ensure the SAS Metadata Server in the BI installation is running.

- a. Click **Advanced**.
  - b. In the Local Services Deployment section, select **Other**.
  - c. Click **Browse** and select the service deployment.
  - d. Click **OK**.
10. If the SAS Metadata Server for the BI installation is running, you can click **Test Configuration** to verify that the service deployment can be read from the metadata server.
  11. If you do not have an open metadata connection, click the "Connection profile" **Change** button and log on before selecting a stored process.

Choose a stored process by clicking **Change**, selecting the stored process, and then clicking **OK**.

Click **Next**.



*Note:* If the Connection profile displays “Invalid connection,” the currently open metadata connection is to a different BI installation than the one targeted by the BI Server profile. Because changing the metadata connection in the wizard is not supported, you must click **Cancel**, disconnect from the metadata server, and then restart this process.

12. Click **Next** to accept the Servlet Class Parameters, and then click **Next** again to accept the Servlet Deployment Descriptor Parameters.
13. Click **Finish**.

The StoredProcessWebApp project is now created and the SAS Stored Process Servlet template is added. leave open the Java file containing the servlet (`StoredProcessDriverServlet.java`).

### **Replace Required Values in the Servlet Code**

The generated servlet code might lack a value for one or more of the stored process input parameters. This occurs when a default value was not defined for the parameter in the metadata for the stored process, and therefore could not be included in the generated code. You must edit the code and provide valid values before the servlet can compile and run.

1. Display the Markers view if it is not visible.

Toggle the Markers view by selecting **Window** ⇒ **Show View** ⇒ **Other** ⇒ **General** ⇒ **Markers**.

2. Look in the Markers view under Java Problems for errors that start with `REPLACE_WITH_ACTUAL_VALUE_FOR_PROMPT_`.  
If the error is not in the Markers view, ensure that automatic building is enabled (**Project** ⇒ **Build Automatically**).
3. Double-click the error in the Markers view to go to the error in the Java file.
4. Replace the problem value with an appropriate constant for that particular input parameter. The input parameter is indicated in the comment above the declaration and by the portion of the variable name that precedes “\_VALUE”.
5. Save the file.

The error disappears, assuming that automatic building is enabled.

For more information about input parameters, including how to see what input parameter values and data types are valid in the code, see [“Input and Output Parameters” on page 28](#).

### ***Restart the Server and Run the Application***

1. Stop and then start the server from the Servers View (do not use Restart). Wait until the server State is Stopped.  
For why you should avoid the Restart command, see the [“Tomcat Shutdown Issue” on page 42](#).
2. Right-click the servlet’s `StoredProcessDriverServlet.java` file and select **Run As** ⇒ **Run on Server**.
3. Ensure that the correct server is selected and click **Finish**.



# Index

---

## A

- Accessibility features [4](#)
  - exceptions [4](#)
- API documentation [62](#)
- AppDev Studio
  - new features [5](#)
  - overview [5](#)
- application metadata
  - copying [39](#)
  - files [37](#)
  - launch files [38](#)
- Application.xml [38](#)
- authentication
  - and deployment [43](#)

## B

- BI Server profiles [9](#)
- build.xml [38](#)

## C

- classes
  - changing default for SAS Java Projects [59](#)
- classpath
  - adding dependent JAR files [55](#)
  - adding new JAR files [54](#)
  - changing JAR file order [53](#)
  - listing JAR files [56](#)
  - removing JAR files [52](#)
- Connection profiles [9](#)

## D

- dependent JAR files
  - identifying [51](#)
- deployment
  - and authentication [43](#)
- deployment descriptor [47](#)
- destination\_omr.properties [38](#)

- documentation
  - accessing [62](#)

## E

- Eclipse
  - memory settings [3](#)
- exporting
  - SAS Java projects as JAR files [45](#)
  - SAS web applications as a WAR file [46](#)

## I

- import statements
  - adding [63](#)
  - Organize SAS Imports action [64](#)
  - Repository Quick Fixes [63](#)
- input parameters [27, 28](#)
- installation [2](#)
  - Eclipse 4.2.2 [1](#)
  - post-install [3](#)

## J

- JAR files
  - adding dependent [55](#)
  - adding to classpath [54](#)
  - changing order on classpath [53](#)
  - finding a class in [58](#)
  - identifying dependencies [51](#)
  - listing classpath JAR files [56](#)
  - listing relationships [57](#)
  - removing from classpath [52](#)
  - removing version restrictions [56](#)
  - specifying current version [55](#)
  - specifying version [55](#)
- Javadoc [62](#)
  - configuring with Eclipse [62](#)

**L**

launch files 38

**M**

memory

    Eclipse settings 3

Metadata Server Connection profiles

*See* [Connection profiles](#)

migrating 6

model/viewer connections 64, 65

    supported viewers 64

**N**

new features 5

**O**

omr.properties 39

Organize SAS Imports action 64

output parameters 29

**P**

portlet editor 36

post-install 3

projects

    opening and closing 8

**Q**

Quick Assists

    enabling 64

**R**

requirements

    Java 2

    SAS software 1

**S**

SAS Repository 49

    opening 50

SAS web applications 7

SAS Web applications

    creating without Web Infrastructure  
    Platform 71

snippets 67

StoredProcessConnection.java 40, 41

StoredProcessDriver.java 39

StoredProcessDriver.properties 40

StoredProcessDriverServlet.java 41

StoredProcessFacade.java 40, 41

**T**

templates 9

Tomcat 42

    shutdown issue 42

**V**

Versioned Jar Repository 49, 71

**W**

web application projects 7

Web application templates 9