Paper 009-2012

Integrating Your Java Web Application into the SAS® 9.2 or SAS 9.3 Enterprise Business Intelligence Environment

Guillaume Curat, SAS Institute Inc., Cary, NC

ABSTRACT

Do you want your existing Java Web application to look and behave like the other SAS applications? The key to achieving this goal is the use of the SAS Logon Manager, SAS metadata roles and capabilities, SAS themes, the SAS Logging Service, and metadata.

This paper explains in detail all the steps to integrate your Java Web Application to an existing SAS Enterprise Business Intelligence (BI) environment. SAS® AppDev Studio™ is used to demonstrate how easily it can be done.

INTRODUCTION

Over the years, companies have developed a lot of Web applications, but integration among each other and the ability to share common services were not really a priority. Now, it is time to try to integrate them into a common platform: the SAS® Web Infrastructure Platform.

The *SAS Web Infrastructure Platform* is a collection of services and applications that provide a common infrastructure and integration features to be used by SAS Web applications. These services and applications provide the following benefits:

- consistency in installation, configuration, and administration tasks for Web applications
- greater consistency in users' interactions with Web applications
- integration among Web applications as a result of the ability to share common resources

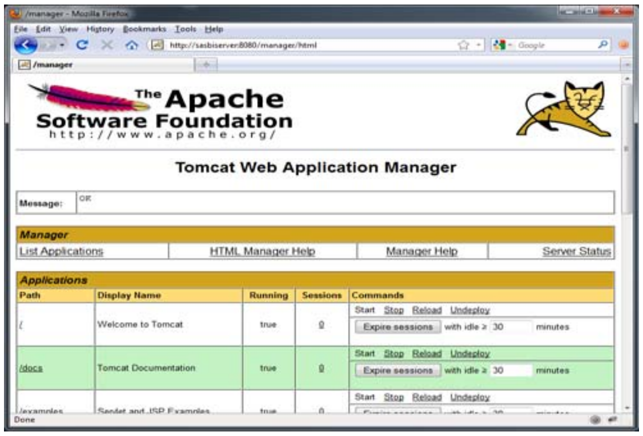
This paper explains in detail how to integrate your Java Web applications into an existing SAS Enterprise BI environment by using the main components of the SAS Web Infrastructure Platform:

- SAS Logon Manager
- SAS Logging Service
- SAS themes
- SAS Web Infrastructure Platform Services
- Role and capabilities

SAS AppDev Studio is used to demonstrate how easily this integration can be done.

SAMPLE WEB APPLICATION TO INTEGRATE

To illustrate the different steps of the integration process, we will use the Tomcat Web Application Manager application provided with Apache Tomcat 6.


/manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://sasbiserver:8080/manager/html
/manager
The Apache Software Foundation http://www.apache.org/

Tomcat Web Application Manager

Message: OK

Manager

List Applications | HTML Manager Help | Manager Help | Server Status

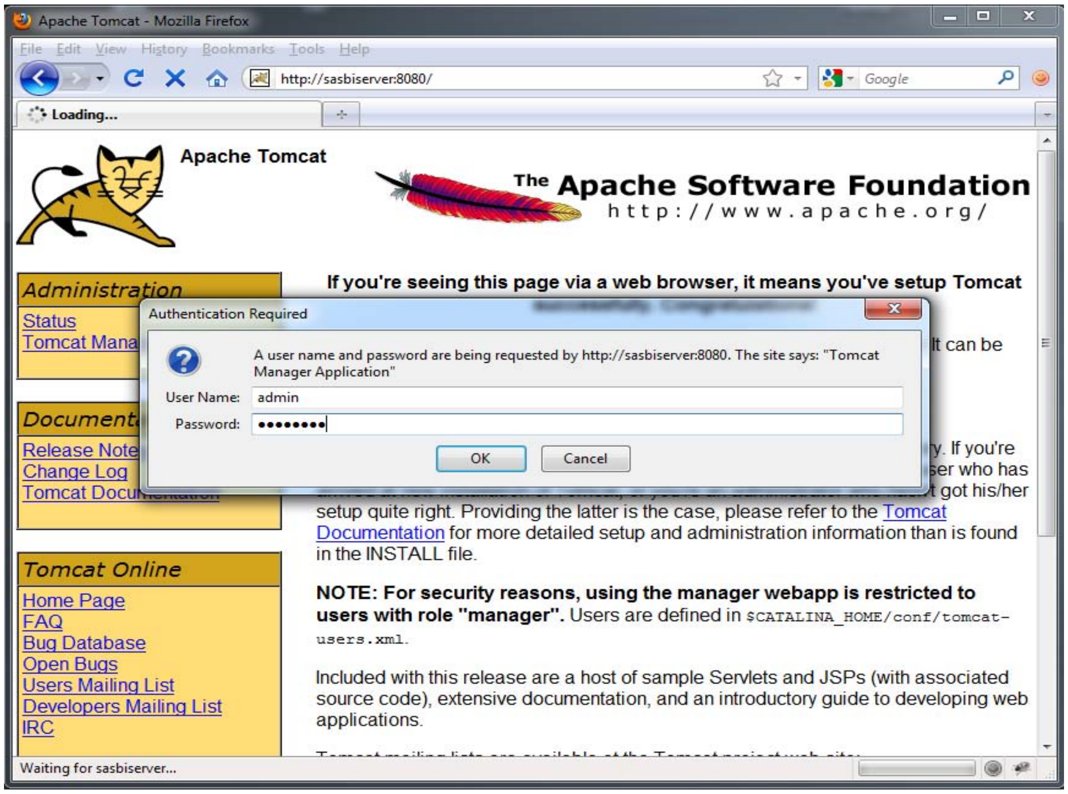
Applications

Path	Display Name	Running	Sessions	Commands
/	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy

Done

Display 1. The Apache Tomcat 6 Web Application Manager

This application contains Java Server Page (JSP) servlets and resource files, and it uses HTTP basic authentication, as shown below.



If you're seeing this page via a web browser, it means you've setup Tomcat

It can be

A user name and password are being requested by http://sasbiserver:8080. The site says: "Tomcat Manager Application"

User Name: admin

Password: ••••••••

y. If you're ser who has t got his/her setup quite right. Providing the latter is the case, please refer to the Tomcat Documentation for more detailed setup and administration information than is found in the INSTALL file.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager". Users are defined in `$CATALINA_HOME/conf/tomcat-users.xml`.

Included with this release are a host of sample Servlets and JSPs (with associated source code), extensive documentation, and an introductory guide to developing web applications.

Display 2. HTTP Basic Authentication for Accessing the Tomcat Web Application Manager

We will use Apache Tomcat version 6.0.32 as a servlet container for testing SAS Web applications under SAS® AppDev Studio. Tomcat runs on a machine named sasbiserver on port 8180.

The SAS Enterprise BI environment (which includes the SAS Web Infrastructure Platform) also runs on the machine named sasbiserver, but it is on port 8080.

SAS WEB APPLICATION COMMONALITIES

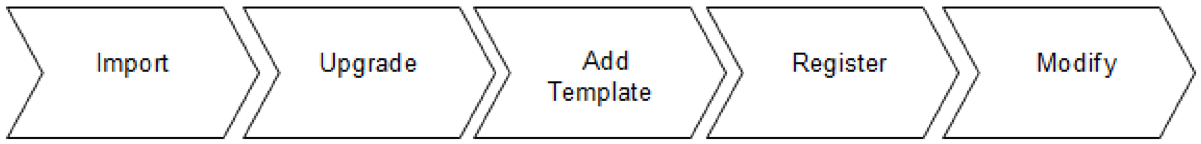
Before discussing the integration process, we need to look at the commonalities of the SAS Web applications. Understanding the commonalities will help you determine which SAS features and functions you want to integrate with.

If you look at several SAS Web applications, you will notice that they all have certain features in common:

- Visual aspects:
  - Uses the same branding, which is defined by a SAS Theme and banner.
  - Accesses applications from a single point: the SAS Logon Manager.
- Metadata: SAS metadata information is attached to each Web application. The metadata defines connection properties and settings.
- Common services: SAS Web applications are using a set of common core infrastructure services that enable integration with the SAS platform. These services are all part of SAS Foundations Services: Logging Service, Information Service, Session Service, User Service, Authentication Service, and so on.
- Same way of managing functionalities: Roles and capabilities are used to limit the functionality of an application.

INTEGRATION PROCESS

The integration process consists of five steps:



1. Import an existing project into SAS AppDev Studio as a Dynamic Web Project.
2. Upgrade the Dynamic Web Project to a SAS Web Application project.
3. Add template content to the project.
4. Register the application in the SAS® Metadata Server.
5. Modify the application to use the SAS components (Foundation Services, SAS themes, roles and capabilities, and so on).

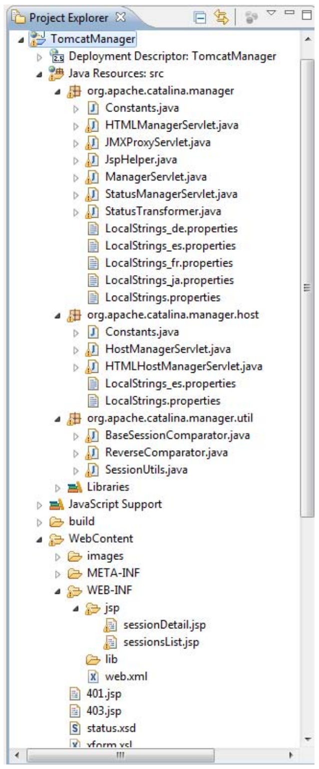
STEP 1: IMPORT

To import the Tomcat Web Application Manager to SAS AppDev Studio as a Dynamic Web Project:

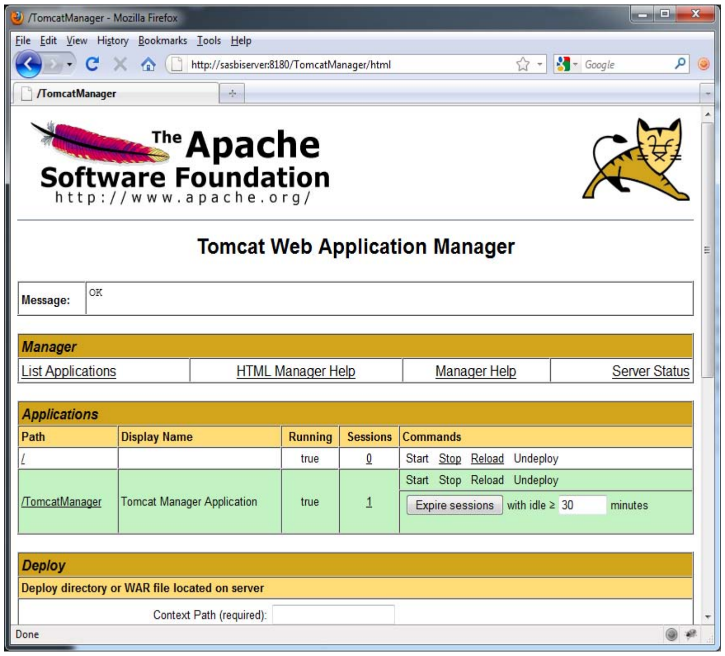
1. Create a new Dynamic Web Project named TomcatManager.
2. Copy and replace all the files from ***`installation-directory`***`\apache-tomcat-6.0.32\webapps\manager` directory to the TomcatManager project in the **`WebContent`** folder.
3. From the Java archive **`apache-tomcat-6.0.32\lib\catalina.jar`**, delete the folder **`org\apache\catalina\manager`**.
4. Create a new Java package named **`org.apache.catalina.manager`** under the **`TomcatManager`** project.
5. Download the source code for Apache Tomcat 6.0.32, and copy all of the files from **`apache-tomcat-6.0.32-src\java\org\apache\catalina\manager`** to the Java package that is created in step 4.
6. Refresh the **`TomcatManager`** project to ensure that the copied files are picked up. (See Display 4, Project Explorer).
7. Add the project to the ADS Apache Tomcat test server. Then start the server.
8. Test the **`TomcatManager`** project by opening the following URL:

   ```
   http://sasbiserver:8180/TomcatManager/html
   ```

   When the URL opens, the Tomcat Web Application Manager should look like the one shown in Display 3 on the following page.



```
Project Explorer
TomcatManager
  Deployment Descriptor: TomcatManager
  Java Resources: src
    org.apache.catalina.manager
      Constants.java
      HTMLManagerServlet.java
      JMXProxyServlet.java
      JspHelper.java
      ManagerServlet.java
      StatusManagerServlet.java
      StatusTransformer.java
      LocalStrings_de.properties
      LocalStrings_es.properties
      LocalStrings_fr.properties
      LocalStrings_ja.properties
      LocalStrings.properties
    org.apache.catalina.manager.host
      Constants.java
      HostManagerServlet.java
      HTMLHostManagerServlet.java
      LocalStrings_es.properties
      LocalStrings.properties
    org.apache.catalina.manager.util
      BaseSessionComparator.java
      ReverseComparator.java
      SessionUtils.java
    Libraries
  JavaScript Support
  build
  WebContent
    images
    META-INF
    WEB-INF
      jsp
        sessionDetail.jsp
        sessionsList.jsp
      lib
      web.xml
    401.jsp
    403.jsp
    status.xsd
    xform.xsl
```



/TomcatManager - Mozilla Firefox

http://sasbiserver:8180/TomcatManager/html

The Apache Software Foundation
http://www.apache.org/

Tomcat Web Application Manager

Message:	OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Display Name	Running	Sessions	Commands
/		true	0	Start Stop Reload Undeploy
/TomcatManager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):

Done

Display 3. Project Explorer (left) and the Tomcat Web Application Manager (right) as It Appears When Launched

STEP 2: UPGRADE

To upgrade the Dynamic Web Project to a SAS Web Application Project:

1. Right-click the **`TomcatManager`** project and select **`Properties`** to open the Properties for TomcatManager dialog box.
2. In the dialog box under **`SAS Java Project Properties`**, select the check box **`Upgrade to SAS Web Application Project`**.
3. Click the **`Apply`** button.

The following project facets are added:

- SAS Java Components:
  - SAS Java Components are a collection of .jar files provided by SAS to the build path of the project. All of the .jar files are grouped under a library called **`SAS Repository`**.
  - Images, scripts, styles, and templates provided by SAS are added to the **`WebContent`** folder of the project.
  - The Web deployment descriptor file (web.xml) is updated.
- SAS Web Infrastructure Platform
  - SAS Web Infrastructure Platform adds Spring Framework configuration files (the **`spring-config`** folder) to the project.
  - The Web deployment descriptor file (web.xml) is updated.

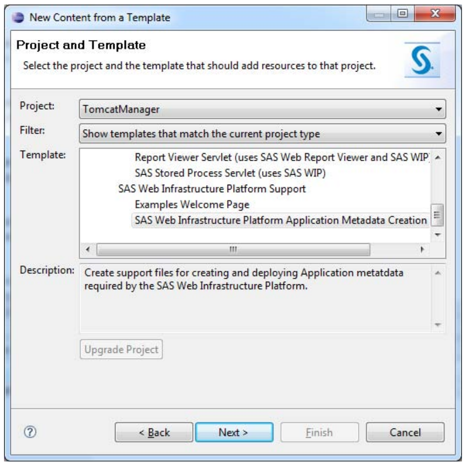
Facets used by a project can be viewed by selecting **Project ► Properties ► Project Facets**.

STEP 3: ADD TEMPLATES

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.

In this step, the **SAS Web Infrastructure Platform Applications Metadata Creation** template is added to the **TomcatManager** project. It creates support files for creating and deploying application metadata that is required by the SAS Web Infrastructure Platform.

1. Right-click the **TomcatManager** project and select **New ►Other**.
2. Select **SAS AppDev Studio / Add Template Content to Project**.
3. In the **Template** section, select **SAS Web Infrastructure Platform Support / SAS Web Infrastructure Platform Application Metadata Creation.**(Display 4)
4. Click the **Next** button.
5. Select the **BI Server Profile** information. (Display 5)
6. Enter the name of the **Application Name**. The name must be unique because it is used by the SAS Logon Manager to identify the Web application. (Display 5)
7. Leave the **Application ID** empty. (Display 5)
8. Enter the appropriate values under **Connection Information**. This information defines the URL that is used by the SAS Logon Manager to return to the Web application. (Display 5)


New Content from a Template

Project and Template

Select the project and the template that should add resources to that project.

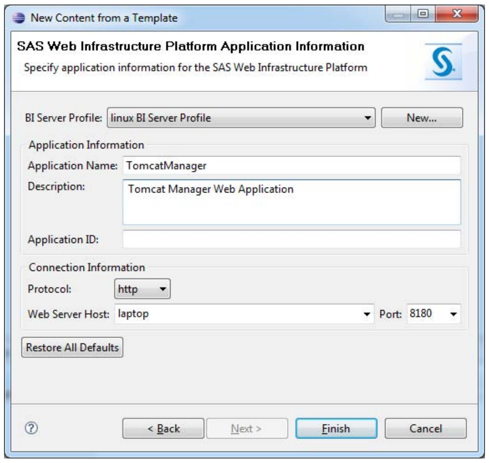
Project: TomcatManager

Filter: Show templates that match the current project type

Template:
Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP
SAS Stored Process Servlet (uses SAS WIP)
SAS Web Infrastructure Platform Support
Examples Welcome Page
SAS Web Infrastructure Platform Application Metadata Creation

Description: Create support files for creating and deploying Application metatdata required by the SAS Web Infrastructure Platform.

Display 4. New SAS Content Template Wizard: Template selection


New Content from a Template

SAS Web Infrastructure Platform Application Information

Specify application information for the SAS Web Infrastructure Platform

BI Server Profile: linux BI Server Profile

Application Information

Application Name: TomcatManager

Description: Tomcat Manager Web Application

Application ID:

Connection Information

Protocol: http

Web Server Host: laptop Port: 8180

Display 5. New SAS Content Template` Wizard: Application information

You can add several templates to the project if needed.

After the Upgrade and Add Template steps, your project should have the following new components:

- a reference to **SAS Repository**
- a metadata folder
- spring-framework integration that creates a **spring-config** folder
- folders for images, scripts, styles, and templates
- an updated web.xml file

STEP 4: REGISTER

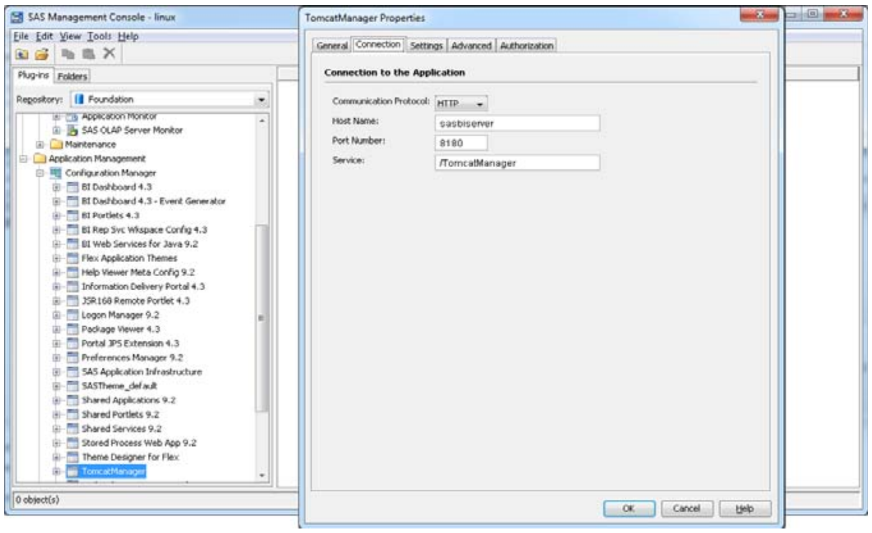
At this point, you have to register the Web application in the SAS Metadata Server. Application metadata is required for the Web application to communicate with the SAS Web Infrastructure Platform.

To create the metadata that is to be associated with the TomcatManager application:

1. Under the **metadata** folder of the project, right-click on the **TomcatManager Create Metadata.launch**.
2. Select **Run As ► TomcatManager Create Metadata**.

3. Verify that the message **`BUILD SUCCESSFUL`** appears at the end of the output logged to the console.
4. Open SAS® 9.2 Management Console by selecting **Application Management ► Configuration Manager**. You can see the metadata that is associated with the Web application. (Display 6)

   Note: For SAS® 9.3, you can view the application metadata by selecting **Application Management ► Configuration Manager ► SAS Application Infrastructure**.



Connection to the Application

Communication Protocol: HTTP
Host Name: sasbiserver
Port Number: 8180
Service: /TomcatManager

Display 6. TomcatManager As It Is Defined in SAS Management Console 9.2

STEP 5: MODIFY

By default, the Tomcat Web Application Manager uses HTTP basic authentication to request a user name and a password from the Web browser whenever the browser requests a resource of the protected Web application.

We want to use the SAS Logon Manager to handle the security access to the Web application. Therefore, we need to remove the HTTP basic authentication mechanism from the Web application.

Under the TomcatManager project in SAS AppDevStudio:

1. Open **`WebContent\WEB-INF\web.xml`**
2. Remove the XML element **`login-config`**.
3. Remove all four occurrences of the **`security-constraint`** XML elements.
4. Remove all five occurrences of the **`security-role`** XML elements.

The default page for the TomcatManager Web application should point to **`/TomcatManager/html`**.

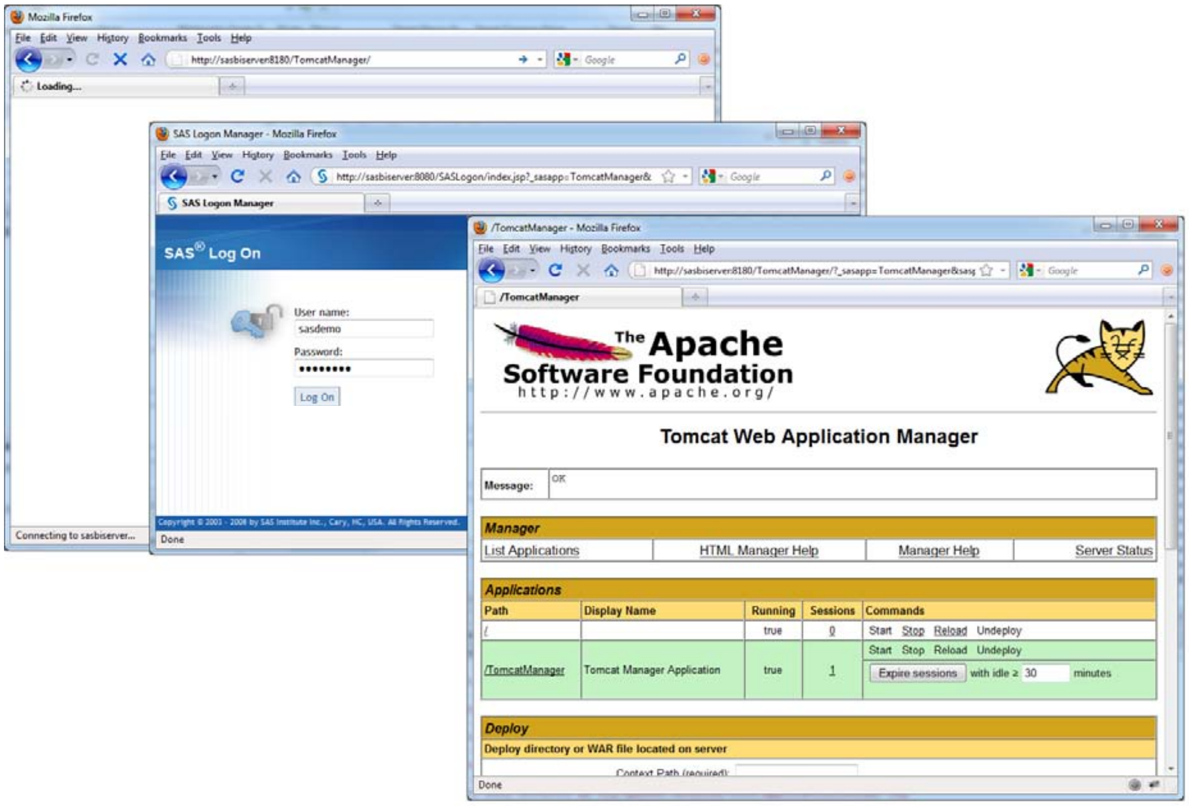
Next, add the following lines to the web.xml file just after the `<web-app. . .>` line:

```
<welcome-file-list>
   <welcome-file>html</welcome-file>
</welcome-file-list>
```

Now, when you try to connect to your Web application (**`http://sasbiserver:8180/TomcatManager`**), you are redirected automatically to the following SAS Logon page:

`http://sasbiserver:8080/SASLogon/index.jsp?_sasapp=TomcatManager&`

After you log on successfully, you are again redirected back to `http://sasbiserver:8180/TomcatManager`.


Mozilla Firefox
http://sasbiserver8180/TomcatManager/
Loading...
Connecting to sasbiserver...
SAS Logon Manager - Mozilla Firefox
http://sasbiserver:8080/SASLogon/index.jsp?_sasapp=TomcatManager&
SAS Logon Manager
SAS® Log On
User name:
sasdemo
Password:
••••••••
Log On
Copyright © 2003 - 2008 by SAS Institute Inc., Cary, NC, USA. All Rights Reserved.
Done
/TomcatManager - Mozilla Firefox
http://sasbiserver8180/TomcatManager/?_sasapp=TomcatManager&sas
/TomcatManager
The Apache Software Foundation
http://www.apache.org/
Tomcat Web Application Manager
Done

Message: OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Display Name	Running	Sessions	Commands
/		true	0	Start Stop Reload Undeploy
/TomcatManager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):

Display 7. Initial Integration with SAS Logon Manager

So far, you have seen how to use the SAS AppDev Studio wizards to integrate your Web application into the SAS environment. However, there are some advanced modifications that are necessary for full integration. These modifications are discussed later in the section Advanced Integration.

BEHIND THE SCENES

At this point, we need to understand what happened to the Dynamic Web Project when we upgraded it to a SAS Web application project.

Spring Framework is the selected method for configuring SAS Web applications to consume services that are provided by the SAS Web Infrastructure Platform.

SAS AppDev Studio helps you generate all of the Spring Framework configuration files, and it updates the Web deployment descriptor (web.xml) of your application.

SPRING FRAMEWORK INTEGRATION

Spring Framework provides a comprehensive programming and configuration model for modern enterprise applications that are Java based. Spring Framework was created to address the complexity of enterprise application development.

- SAS 9.2 is using Spring Framework version 2.5.5.0.
- SAS 9.3 is using Spring Framework version 3.0.5.0.

The following tables list and explain the roles of the Spring Framework configuration files that are created by the SAS AppDev Studio wizard during the Upgrade step.

SAS 9.2

Configuration Files That Are Added to the `spring-config` Folder	Description of the Configuration Files	Method Used to Define the Spring Elements
infrastructure-config.xml	Imports JavaBean definitions related to platform infrastructure such as JMS, data sources, SAS® Foundation Services.	Standard Spring dependency injection: • `classpath*:META-INF/wip-services-client-config.xml` • `classpath*:META-INF/spring-config/aop-config.xml` • `classpath*:META-INF/spring-config/data-config.xml` • `classpath*:META-INF/spring-config/presentation-config.xml` • `classpath*:META-INF/spring-config/jps-config.xml` • `classpath*:META-INF/xss-config.xml`
webapp-config.xml	Specifies or imports JavaBean definitions related to Java Web application components, such as servlet filters.	Standard Spring dependency injection: `classpath*:META-INF/spring-config/webapp-config.xml`

SAS 9.3

Configuration Files That Are Added to the `spring-config` Folder	Description of the Configuration Files	Method Used to Define the Spring Elements
aop-config.xml	Contains AOP bean definitions for the SAS Web infrastructure Platform	
data-config.xml	Specifies the bean definitions associated with the data access layer (tables, SAS metadata, properties files…).	Explicit definition of the JavaBeans.
infrastructure-config.xml	Specifies or imports JavaBean definitions related to platform infrastructure such as JMS, SAS Foundation Services.	Explicit definition of JavaBeans and dependency injection of the following: `classpath*:META-INF/remote-jps.xml`
presentation-config.xml	Specifies JavaBean definitions related to user interface elements, such as themes.	Explicit definition of the JavaBeans *(table continued)*

Configuration Files That Are Added to the `spring-config` Folder	Description of the Configuration Files	Method Used to Define the Spring Elements
services-config.xml	Imports JavaBean definitions for service use	Spring dependency injection: `classpath*:/META-INF/wip-services-client-config.xml`
webapp-config.xml		(This file is empty.)
wip-config.xml	Specifies JavaBean definitions related to Java Web application components, such as servlet filters.	Explicit definition of the JavaBeans
xss-config.xml	Specifies JavaBean definitions related to cross-site scripting protection	Explicit definition of the JavaBeans

WEB DEPLOYMENT DESCRIPTOR

The Web deployment descriptor of the SAS Web application (**`WEB-INF/web.xml`**) contains all of the components to integrate with the Spring Framework and the SAS Web Infrastructure Platform.

The Upgrade step adds the following main elements to the web.xml file:

Main Elements of the web.xml File	Description
`context-param: locatorFactorySelector` **`context-param: parentContextKey`**	Used to access common properties across the middle-tier applications. The beanRefContext.xml file is provided as part of the sas.svcs.cluster.jar file.
`context-param: contextConfigLocation`	Initializes the Spring Framework integration.
`filter: SanitizingRequestFilter`	Addresses cross-site scripting vulnerabilities in Web applications.
`filter: CharacterEncodingFilter`	Sets the character encoding for input.
`filter: WIPSecurityFilter`	The main security Filter. It is used to ensure that the user is authenticated and has a SAS session established.
`filter:  ThemeSupportFilter`	Enables access to the current theme. (The following sessions attributes are set: DISPLAY_THEME_OBJECT, BROWSER_TYPE.) *(table continued)*

Main Elements of the web.xml File	Description
`filter: WIPPlatformServicesFilter`	Sets a number of HTTP session attributes: PFS_SESSIONID, REMOTE_SESSION_CONTEXT, REMOTE_USER_CONTEXT, USER_ID, and USER_NAME.
`servlet: director`	Constructs URIs for redirection to Web applications.
`servlet: logoff`	Handles the process of logging out a user from the application.

ADVANCED INTEGRATION

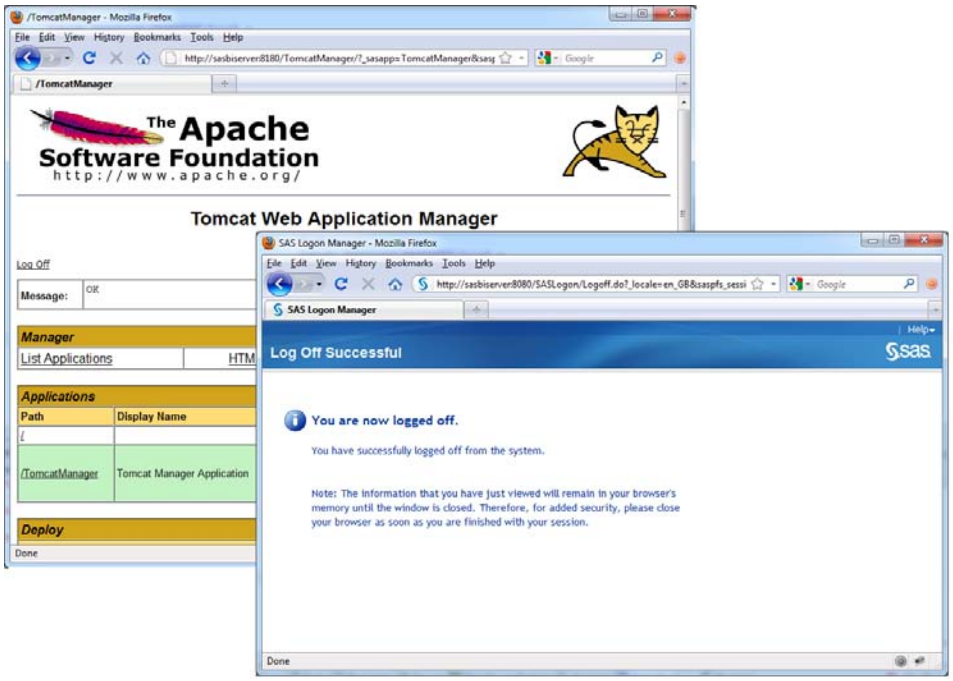
So far, we have used SAS AppDev Studio wizards to complete the initial integration (Upgrade, Add Template, and Register steps). To fully integrate with the SAS Web Infrastructure Platform, we need to make some manual modifications to the Web application, as explained in the following sections.

SAS LOGON MANAGER

During the previous integration steps, the Web application was automatically configured to use the SAS Logon Manager to authenticate users. To enable the user to log off from the Web application, we just need to call the servlet named Logoff. To do that, add the following HTML code to the main page:

```
    <a href="/TomcatManager/Logoff">Log Off</a>
```

After this code is added, users should be able to log off successfully, as shown in the following display:


/TomcatManager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://sasbiserver8180/TomcatManager/?_sasapp=TomcatManager&sas
/TomcatManager
The Apache Software Foundation http://www.apache.org/
Tomcat Web Application Manager
Log Off
Manager
List Applications
HTM
Applications
Path
Display Name
Deploy
Done
SAS Logon Manager - Mozilla Firefox
http://sasbiserver8080/SASLogon/Logoff.do?_locale=en_GB&saspfs_sessi
SAS Logon Manager
Help
Log Off Successful
sas
Done

Message: OK

Path	Display Name
/	
/TomcatManager	Tomcat Manager Application

You are now logged off.

You have successfully logged off from the system.

Note: The information that you have just viewed will remain in your browser's memory until the window is closed. Therefore, for added security, please close your browser as soon as you are finished with your session.

Display 8. Logoff Enabled

SAS LOGGING SERVICE

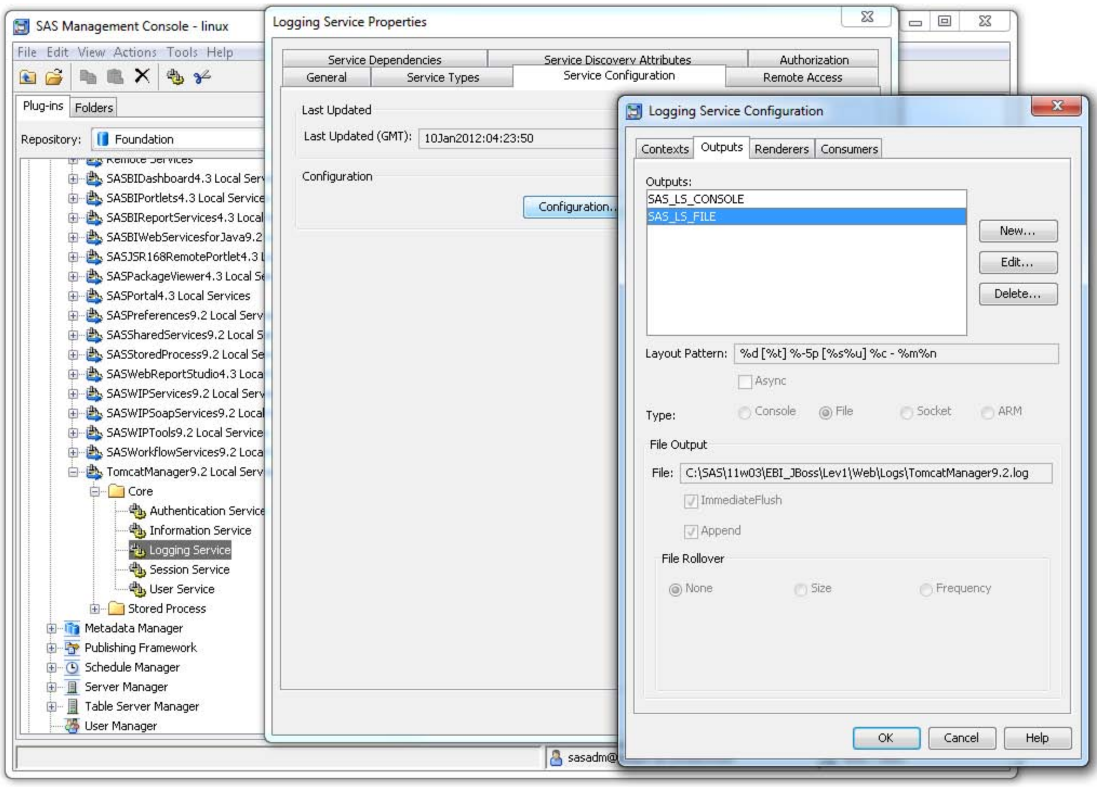
The *SAS Logging Service* provides a standardized mechanism for generating and handling logged messages. To enable the service for the Web application, we have to create a new Local Foundation Services deployment descriptor. Afterwards, we need to modify the web.xml file in order to use the new descriptor.

Note: The following steps **are valid only for SAS 9.2**. The SAS Logging Service is deprecated in SAS 9.3**.** For details about how to handle logged messages in SAS 9.3 details, see the section Logging Service Deprecation in SAS® 9.3.

Definition of the Local Services

Define the Local Services in SAS Management Console, as follows:

1. Under **`Foundation Services Manager`**, right-click **`SASWIPServices9.2 Local Services`** and select **`Duplicate Service Deployment`**. Rename the new deployment to **`TomcatManager9.2 Local Services`**.
2. Select **TomcatManager 9.2 Local Services ► Core.**
3. Under **`Core,`** right-click **`Logging Service`** and select **`Properties`**. In the Logging Service Properties dialog box, click the **`Service Configuration`** tab.
4. Click the **`Configuration`** button to open the Logging Service Configuration dialog box.
5. In the dialog box, click the **`Outputs`** tab.
6. Select **`SAS_LS_FILE`** in the **`Outputs`** box. Then, under **`File Output`**, add the following path to the **`File`** text field: ***`sas-configuration-directory`*`\Lev1\Web\Logs\TomcatManager9.2.log`**.


SAS Management Console - linux
File Edit View Actions Tools Help
Plug-ins Folders
Repository: Foundation
Remote Services
SASBIDashboard4.3 Local Serv
SASBIPortlets4.3 Local Service
SASBIReportServices4.3 Local
SASBIWebServicesforJava9.2
SASJSR168RemotePortlet4.3
SASPackageViewer4.3 Local Se
SASPortal4.3 Local Services
SASPreferences9.2 Local Serv
SASSharedServices9.2 Local S
SASStoredProcess9.2 Local Se
SASWebReportStudio4.3 Loca
SASWIPServices9.2 Local Serv
SASWIPSoapServices9.2 Local
SASWIPTools9.2 Local Service
SASWorkflowServices9.2 Loca
TomcatManager9.2 Local Serv
Core
Authentication Service
Information Service
Logging Service
Session Service
User Service
Stored Process
Metadata Manager
Publishing Framework
Schedule Manager
Server Manager
Table Server Manager
User Manager
sasadm@
Logging Service Properties
Service Dependencies
Service Discovery Attributes
Authorization
General
Service Types
Service Configuration
Remote Access
Last Updated
Configuration
Configuration...
Logging Service Configuration
Contexts
Outputs
Renderers
Consumers
New...
Edit...
Delete...
OK
Cancel
Help

Last Updated (GMT): 10Jan2012:04:23:50

Outputs:
SAS_LS_CONSOLE
SAS_LS_FILE

Layout Pattern: %d [%t] %-5p [%s%u] %c - %m%n

Async

Type: Console [✓ File] Socket ARM

File Output

File: C:\SAS\11w03\EBI_JBoss\Lev1\Web\Logs\TomcatManager9.2.log

[✓ ImmediateFlush]

[✓ Append]

File Rollover

[✓ None] Size Frequency

Display 4, TomcatManager Logging Service Properties and Configuration Dialog Boxes

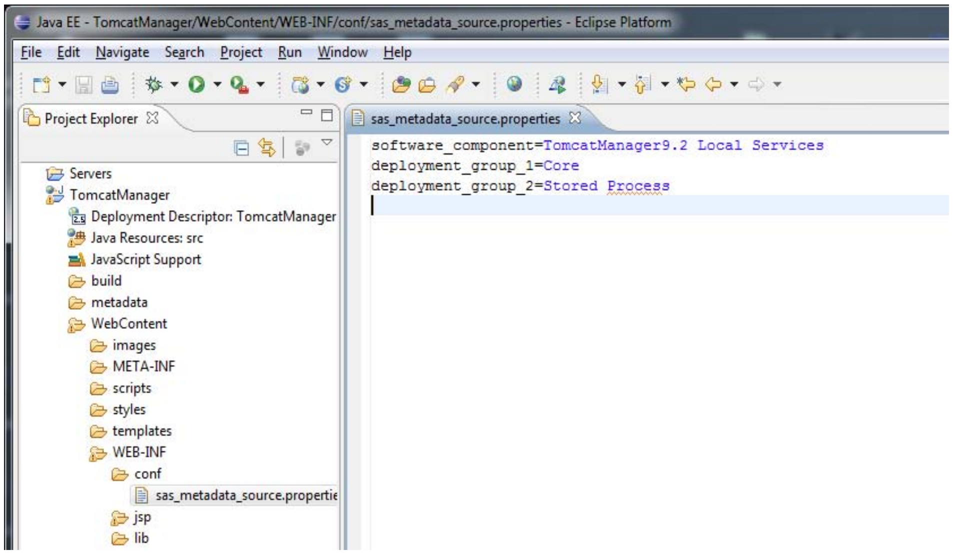
Modify the Web Application to Use the Local Services

Under the TomcatManager project in SAS AppDev Studio:

1. Create a new folder named **`conf`** in the **`WebContent/WEB-INF`** folder.
2. In the **`conf`** folder, create a new file named sas_metadata_source.properties that contains the following content:

```
software_component=TomcatManager9.2 Local Services
deployment_group_1=Core
deployment_group_2=Stored Process
```

The `software_component` name must match the name of the Local Services created previously.



Java EE - TomcatManager/WebContent/WEB-INF/conf/sas_metadata_source.properties - Eclipse Platform
File Edit Navigate Search Project Run Window Help
Project Explorer
Servers
TomcatManager
Deployment Descriptor: TomcatManager
Java Resources: src
JavaScript Support
build
metadata
WebContent
images
META-INF
scripts
styles
templates
WEB-INF
conf
sas_metadata_source.propertie
jsp
lib
sas_metadata_source.properties

```
software_component=TomcatManager9.2 Local Services
deployment_group_1=Core
deployment_group_2=Stored Process
```

Display 5. Contents of the sas_metadata_source.properties File

3. Modify **`WEB-INF/spring-config/infrastructure-config.xml`**, as follows.
   a. Locate the following line in the file:

      ```
      <import resource="classpath*:META-INF/spring-config/jps-config.xml" />
      ```

   b. Replace the line above with the following line:

      ```
      <import resource="classpath*:META-INF/spring-config/jps-config-local-
      remote.xml" />
      ```

4. Add the following elements to **`WEB-INF/web.xml`**:

```
    <context-param>
                  <param-name>log4j-configuration-name-prefix</param-name>
                  <param-value>TomcatManager</param-value>
    </context-param>

<!-- logging context separation listener (this should be the FIRST
     listener!!!) -->

<listener>
      <listener-class>com.sas.svcs.logging.LoggingContextListener</listener-
                                                                     class>
</listener>
```

5. Create a new log4j configuration file in ***`sas-configuration-directory`*`\Lev1\Web\Common\LogConfig\TomcatManager-log4j.xml`**. To do that, make a copy of the SASLogon-log4.xml file and rename it TomcatManager-log4j.xml.
6. Add the following Java Virtual Machine (JVM) option to the launch configuration of the Tomcat application server:

```
-Dcom.sas.log.config.url=file:///sas-configuration-directory/Lev1/
                                 Web/Common/LogConfig/
```

Write to the Log File

Use the following Java code to write to the log file. (Modify **`org/apache/catalina/manager/HTMLManagerServlet.java`**, under the list method.)

```
import com.sas.services.logging.LoggerInterface;
import com.sas.services.logging.LoggingServiceInterface;
import com.sas.services.webapp.ServicesFacade;

LoggingServiceInterface _loggingServiceInterface =
         ServicesFacade.getLoggingService();
LoggerInterface logger =
         _loggingServiceInterface.getLogger(this.getClass().getName());

logger.warn("Tomcat Manager Accessed.");
```

When you access the TomcatManager application, the **`TomcatManager9.2.log`** file is created under ***`sas-configuration-directory`*** **`\Lev1\Web\Logs\`**.

The TomcatManager9.2.log output looks similar to the following:

```
2012-01-10 13:32:42,708 [http-8180-1] WARN  []
org.apache.catalina.manager.HTMLManagerServlet - Tomcat Manager Accessed.
```

Output 1. TomcatManager9.2.log

Logging Service Deprecation in SAS® 9.3

The Logging Service is deprecated in SAS 9.3. Therefore, you should use Apache log4j.

Under the TomcatManager project in SAS AppDev Studio:

1. Add the following parameters to **`WEB-INF/web.xml`**

```
     <context-param>
            <param-name>log4j-config-name-prefix</param-name>
            <param-value>TomcatManager</param-value>
     </context-param>

<!--Logging context separation listener (this should be the FIRST
     listener!!!)-->
<listener>
    <listener-class>com.sas.svcs.logging.LoggingContextListener</listener-
    class>
</listener>
```

2. Create a new log4j configuration file in ***`sas-configuration-directory`*****`\Lev1\Web\Common\LogConfig\TomcatManager-log4j.xml`**.To do that, make a copy of the SASLogon-log4.xml file and rename it TomactManager-log4j.xml. Open TomcatManager-log4j.xml with a text editor, and replace all occurrences of SASLogon9.3.log by TomactManager93.log.

   You might want to change the root priority from ERROR to WARN or DEBUG.

3. Add the following JVM option to the launch configuration of the Tomcat application server:

```
-Dcom.sas.log.config.url=file:///sas-configuration-directory/Lev1/Web
                                /Common/LogConfig/
```

4. Use the following Java code to write to the log file. (Modify **`org/apache/catalina/manager/HTMLManagerServlet.java`**, under the list method.)

```
import org.apache.log4j.Logger;

Logger log = Logger.getLogger(this.getClass().getName());
log.warn("Tomcat Manager Accessed");
```

SAS THEMES

The following code shows how to add the SAS banner and apply a SAS theme to the Web application. Adding the banner and a theme ensures that all Web applications are consistent in their general appearance. We can use either of the following options to add the banner and apply a theme.

Option1: Use the SAS taglib Directive (in a JSP page)

a. Add the following taglib directive into the header section of a JSP page.

```
<%@ taglib uri="http://www.sas.com/taglib/sas" prefix="sas" %>
<sas:InitializeComponents/>
<sas:StyleSheet/>
```

b. At the beginning of the HTML body tag, insert the following code:

```
<sas:Banner title="Tomcat Manager" secondaryTitle="Welcome" userName=""
logOffURL="Logoff" helpDocURL="doc.do" helpAboutURL="help.do"
helpAboutTitle="XYZ" preferencesURL="preferences.do"/>
```

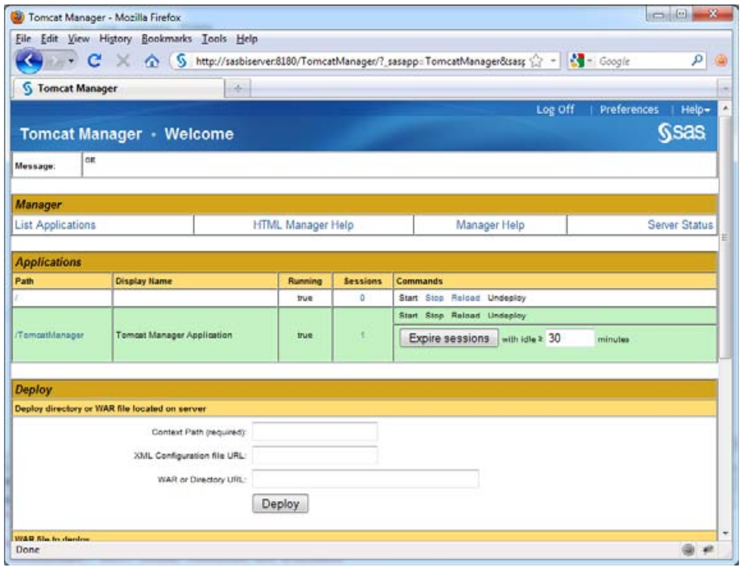
Option 2: Use Java Code

Add the following code to **`org/apache/catalina/manager/HTMLManagerServlet.java`**, under the list method.

```
import com.sas.services.webapp.ServicesFacade;
import com.sas.servlet.util.BaseUtil;
import com.sas.web.keys.CommonKeys;
import com.sas.servlet.tbeans.html.Banner;
import com.sas.framework.themes.client.Theme;
import com.sas.framework.themes.client.BrowserType;

  //Initialize SAS components to add Javascript to the HTML page.
com.sas.servlet.util.Components.init(writer, request);

  // Retrieve the current theme.
Theme theme = (Theme) request.getAttribute(CommonKeys.DISPLAY_THEME_OBJECT);

  // Retrieve the browser type.
BrowserType browserType =
         (BrowserType) request.getAttribute(CommonKeys.BROWSER_TYPE);

  //Retrieve the SAS style sheets according to the current theme and browser.
String links = BaseUtil.getStyleSheetLinks(theme, browserType,"SAS Style");
writer.println(links);

Banner banner = new Banner();
banner.setRequest(request);
banner.setTitle("Tomcat Manager");
banner.setSecondaryTitle("Welcome");
banner.setUserName("");
banner.setLogOffURL("/TomcatManager/Logoff");
banner.setHelpAboutURL("help.do");

banner.setHelpDocURL("doc.do");
banner.setPreferencesURL("preferences.do");

banner.setShowDivider(false);
banner.write(writer);
```

For better output, set the margin to **`0`** for the HTML body.

```
<body style="margin: 0pt;">
```


Tomcat Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://sasbiserver:8180/TomcatManager/?_sasapp=TomcatManager&sasp
Google
Tomcat Manager
Log Off | Preferences | Help
Tomcat Manager • Welcome
sas
Deploy
Expire sessions
Done

Message: OK

Manager

List Applications	HTML Manager Help	Manager Help	Server Status

Applications

Path	Display Name	Running	Sessions	Commands
/		true	0	Start Stop Reload Undeploy
/TomcatManager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Deploy

Deploy directory or WAR file located on server

Context Path (required):

XML Configuration file URL:

WAR or Directory URL:

WAR file to deploy

Display 6. Banner and Theme Integration

SAS® WEB INFRASTRUCTURE PLATFORM SERVICES

Now, let's see how we can use some of the SAS Web Infrastructure Platform services in order to obtain information about the connected user and the application environment.

Spring Web Application Context

The Spring Framework *Web Application context* provides an interface to access configuration information about the Web application. You can obtain the context name of the application and access all of the Java Beans that are defined by using the following code:

```
import org.springframework.web.context.WebApplicationContext;
import org.springframework.web.context.support.WebApplicationContextUtils;
import javax.servlet.ServletContext;

    //Retrieve the servlet context.
ServletContext sc = getServletContext();
    // Retrieve the Web application context
WebApplicationContext wac=
                     WebApplicationContextUtils.getWebApplicationContext(sc);
    //Retrieve the context name of the Web application.
String  appName = sc.getInitParameter("application-name");
```

Configuration Service

The *Configuration Service* provides a standard way to define, store, modify, discover, and retrieve application configuration information for SAS components deployed in the middle tier.

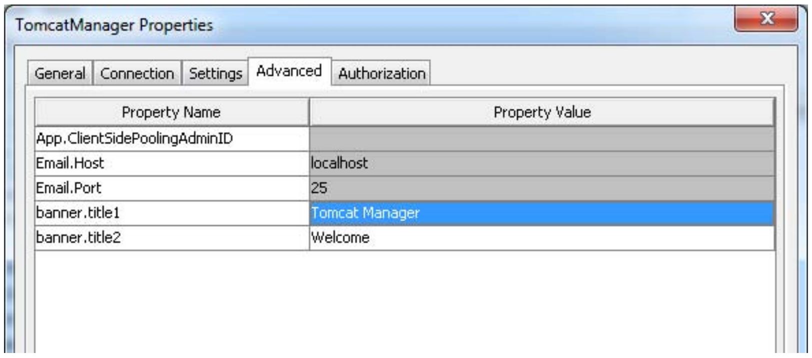
For example, you can retrieve the Web application properties that are defined in the metadata by using this code:

```
import com.sas.framework.config.ConfigurationServiceInterface;
import java.util.Properties;

ConfigurationServiceInterface configService =
     (ConfigurationServiceInterface)wac.getBean("configurationService");
```

(code continued)

```
      //Retrieve the properties of the Web application.
   Properties prop = configService.getSettings(appName);
   String title1 = prop.getProperty("banner.title1"); //Tomcat Manager
   String title2 = prop.getProperty("banner.title2"); //Welcome
```

The TomcatManager application depends on the Web Infrastructure Platform. Therefore, the properties that are returned by the Configuration Service are all of the properties of the Web Infrastructure Platform plus the properties of the TomcatManager application.



Property Name	Property Value
App.ClientSidePoolingAdminID	
Email.Host	localhost
Email.Port	25
banner.title1	Tomcat Manager
banner.title2	Welcome

Display 7. TomcatManager Properties in SAS Management Console

Each time modifications are made to the application metadata properties, the SAS Remote Services and the Web application server must be restarted.

With SAS 9.3, the class `com.sas.framework.config.ConfigurationServiceInterface` has been deprecated. Therefore, we need to use the class `com.sas.svcs.config.client.ConfigurationServiceInterface` instead and use the bean named `svcs.configurationService`:

```
   import com.sas.svcs.config.client.ConfigurationServiceInterface;
   import java.util.Properties;

   ConfigurationServiceInterface configService =
        (ConfigurationServiceInterface)wac.getBean("svcs.configurationService");
```

Security Service

The *Security Service* context contains information about the authenticated user. You can retrieve the Security context from the session with the following code.

```
   import com.sas.svcs.authentication.client.SecurityContext;

      //Retrieve the Security context from the session.
   SecurityContext sec =
        (SecurityContext)request.getSession().getAttribute("waf_security");
   String user_name = sec.getName(); //SAS Demo User
   String user_id = sec.getId();     //sasdemo
```

To retrieve only the user name and ID of the authenticated user, use this code:

```
   import com.sas.web.keys.CommonKeys;
   String user_id =
       (String)request.getSession().getAttribute(CommonKeys.USER_ID);
   String user_name =
       (String)request.getSession().getAttribute(CommonKeys.USER_NAME);
```

User Service

The *User context* contains information about the authenticated user. You can retrieve that context with the following code:

```
import com.sas.services.user.UserContextInterface;
import com.sas.web.keys.CommonKeys;
import com.sas.services.user.UserIdentityInterface;

UserContextInterface userContext =
   (UserContextInterface)session.getAttribute(CommonKeys.REMOTE_USER_CONTEXT);

String authDomain = userContext.getAuthServer().getDomain();
UserIdentityInterface userIdentity =
                                  userContext.getIdentityByDomain(authDomain);
String username = (String)userIdentity.getPrincipal();      //sasdemo
String password  = (String)userIdentity.getCredential();    //password
```

UserInfo Service

The *UserInfo Service* provides mechanisms for returning information about a specific user (name, job title, e-mail address, phone, and so on). You can retrieve this information with the following code:

```
import com.sas.svcs.userinfo.client.UserInfoServiceInterface;
import com.sas.svcs.userinfo.client.UserContactInfo;
import com.sas.svcs.userinfo.client.UserDetails;

   //Retrieve userInfoService.
UserInfoServiceInterface userInfoService =
   (UserInfoServiceInterface)wac.getBean("userInfoService");
   //Retrieve the user contact information  for a specific user.
UserContactInfo userContactInfo = userInfoService.getContactInfo(user_id);
List emails = userContactInfo.getEmailAddresses();

   //Get the UserDetails Info for a specific user.
UserDetails userDetails = userInfoService.getDetails(user_id);
String  job_title = userDetails.getTitle();
```

Theme Service

The *Theme Service* provides access to SAS Theme information and resources. We can retrieve this information with this code:

```
import com.sas.framework.themes.client.ThemeServiceInterface;
import com.sas.framework.themes.client.Theme;
import com.sas.framework.themes.client.Image;

    //Retrieve  themeService.
ThemeServiceInterface  themeService =
                           (ThemeServiceInterface)wac.getBean("themeService");
    //Retrieve the list of all the themes that are available.
List themes = themeService.getThemeNames();

    //Retrieve a theme.
Theme theme = themeService.getTheme((String)themes.get(0));

    //Retrieve a theme resource to display(an image that is defined in
      SASthemes.xml)
Image image = theme.getImage("tbar_Email");
String imagePath = image.getFile();
//imagePath conatins the full path to the image:
//http://sasbiserver:8080/SASTheme_default/themes/default/images/TbarEmail.gif
```

ROLES AND CAPABILITIES

Roles and capabilities are used to limit the functionality of an application.

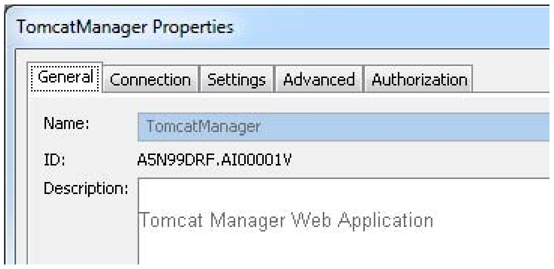
In the following TomcatManager example, the banner menus are restricted so that only certain users can see the **Preferences**, **Help**, and **Logoff** menus. This example also controls access to the sections (deploy, diagnostics, and so on).

Create New Capabilities in Metadata

Before making any changes to the SAS metadata server, you should perform a backup.

To add new capabilities in the metadata:

1. Retrieve the metadata ID of your Web application definition.
   a. Open SAS Management Console and select **Application Management ► Configuration Manager**.
   b. Right-click **TomcatManager** and select **Properties** to open the TomcatManager Properties dialog box. The metadata ID of the application (**A5N99DRF.AI00001V**) should be displayed on the **General** tab .



Name: TomcatManager
ID: A5N99DRF.AI00001V
Description: Tomcat Manager Web Application

Display 8. Metadata ID of the Web Application

2. Enable the XML Metadata Interface in SAS Management Console.
   a. Navigate to the ***SAS_HOME*\SASManagementConsole\9.2\plugins\advanced** directory.
   b. Copy the **omitoolsmc** directory to ***SAS_HOME*\SASManagementConsole\9.2\plugins\**.
   c. Restart SAS Management Console.
3. Create metadata.
   a. In SAS Management Console, launch the XML Metadata Interface, which is available from the **Tools** menu.
   b. On the **Update Metadata** tab, enter the following XML code in the **Input XML** field.

Note: The SoftwareComponent ID must match the ID of the Web application.

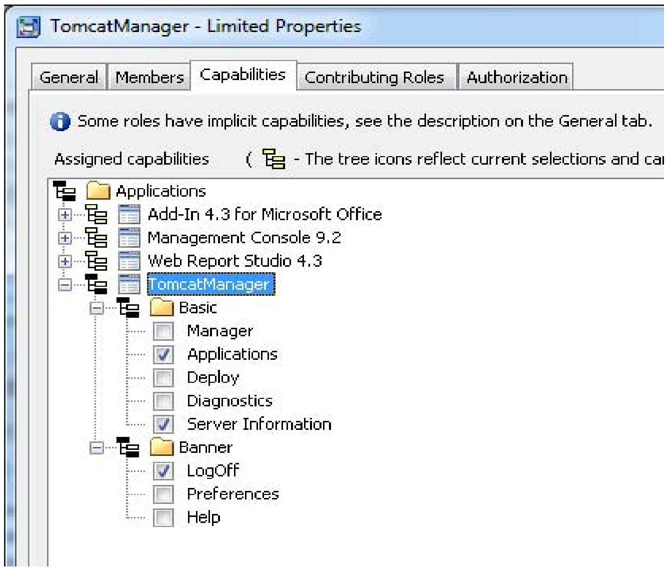
```
<!-- Update the Software Component ID with the ID of the Web
     application. -->
<SoftwareComponent Id="A5N99DRF.AI00001V" PublicType="Application" >
   <SoftwareTrees>
      <Tree Name="ApplicationActions" PublicType=""
            TreeType="ApplicationActions">
        <SubTrees>
           <!-- Create a new group of capabilities; TreeType must have
                a value of ApplicationActions. -->
           <Tree Name="Basic" Desc="Basic features" PublicType=""
                 TreeType="ApplicationActions">
```

(code continued)

```
              <Members>
               <!-- Add capabilities to the group; ActionType must
               have a value of Feature. -->
               <ApplicationAction Name="Manager"
                                  Desc="Enable Manager Menu"
                                  ActionIdentifier="Manager"
                                  ActionType="Feature" />
                 <ApplicationAction Name="Applications"
                                    Desc="Enable Applications Menu"
                                    ActionIdentifier="Applications"
                                    ActionType="Feature" />
                 <ApplicationAction Name="Deploy"
                                    Desc="Enable Deploy Menu"
                                    ActionIdentifier="Deploy"
                                    ActionType="Feature" />
                 <ApplicationAction Name="Diagnostics"
                                    Desc="Enable Diagnostics Menu"
                                    ActionIdentifier="Diagnostics"
                                    ActionType="Feature" />
                 <ApplicationAction Name="Server Information"
                                    Desc="Enable Server Menu"
                                    ActionIdentifier="ServerInfo"
                                    ActionType="Feature" />
              </Members>
          </Tree>
          <Tree Name="Banner" Desc="Banner features" PublicType=""
                 TreeType="ApplicationActions" >
              <Members>
                 <ApplicationAction Name="LogOff"
                                    Desc="Show Log Off in Banner"
                                    ActionIdentifier="banner.logoff"
                                    ActionType="Feature"  />
                 <ApplicationAction Name="Preferences"
                                    Desc="Show Preferences in Banner"
                                    ActionIdentifier="banner.pref"
                                    ActionType="Feature"  />
                 <ApplicationAction Name="Help"
                                    Desc="Show Help in Banner"
                                    ActionIdentifier="banner.help"
                                    ActionType="Feature"  />
              </Members>
          </Tree>
      </SubTrees>
   </Tree>
</SoftwareTrees>
</SoftwareComponent>
```

c. Click the **Execute** button.

4. Create a new role and select new capabilities under the TomcatManager application.


TomcatManager - Limited Properties

General | Members | Capabilities | Contributing Roles | Authorization

Some roles have implicit capabilities, see the description on the General tab.

Assigned capabilities (- The tree icons reflect current selections and ca

- Applications
  - Add-In 4.3 for Microsoft Office
  - Management Console 9.2
  - Web Report Studio 4.3
  - TomcatManager
    - Basic
      - [ ] Manager
      - [✓] Applications
      - [ ] Deploy
      - [ ] Diagnostics
      - [✓] Server Information
    - Banner
      - [✓] LogOff
      - [ ] Preferences
      - [ ] Help

Display 9. Roles and Capabilities in SAS Management Console

Integrate Capabilities into the Web Application

1. Use the following code to retrieve the roles and granted capabilities:

```
    import com.sas.services.security.ApplicationAuthorization;

       //Retrieve the roles.
     List roles = userContext.getRoles(); //[Add-In for Microsoft
                                          // Office: Advanced, TomcatManager -
                                          // Limited]

        //Retrieve the granted capabilities.
     List caps = userContext.getActions(appName,true);

     List<String> capabilities = new ArrayList<String>(caps.size());
     for (int i=0;i<caps.size();i++)
         capabilities.add(((ApplicationAuthorization)caps.get(i)).getName());

 //Granted capabilities contain [Applications, Server Information, LogOff]
```

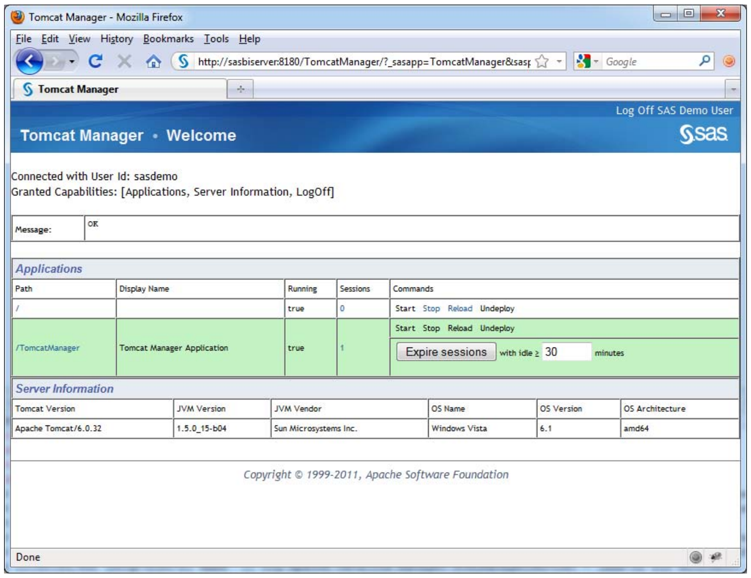
2. Now modify the banner code, as follows, to take advantage of the capabilities defined in metadata.

```
Banner banner = new Banner();
banner.setRequest(request);
// The banner titles are read from the Web application properties.
banner.setTitle(prop.getProperty("banner.title1"));
banner.setSecondaryTitle(prop.getProperty("banner.title2"));
banner.setUserName(user_name);

if (capabilities.contains("Help"))
{
      banner.setHelpAboutURL("About.do");
      banner.setHelpDocURL("Help.do");
}

if (capabilities.contains("LogOff"))
      banner.setLogOffURL("/"+appName+"/Logoff");
```

(code continued)

```
        if (capabilities.contains("Preferences"))
              banner.setPreferencesURL("/Preferences.do");

        banner.setShowDivider(false);
        banner.write(writer);
```

Once we modify the banner, the Tomcat Manager application is fully integrated with SAS.


Tomcat Manager - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://sasbiserver:8180/TomcatManager/?_sasapp=TomcatManager&sasp
Google
Tomcat Manager
Log Off SAS Demo User
Tomcat Manager • Welcome
sas
Done

Connected with User Id: sasdemo
Granted Capabilities: [Applications, Server Information, LogOff]

Message:	OK

Applications

Path	Display Name	Running	Sessions	Commands
/		true	0	Start Stop Reload Undeploy
/TomcatManager	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Server Information

Tomcat Version	JVM Version	JVM Vendor	OS Name	OS Version	OS Architecture
Apache Tomcat/6.0.32	1.5.0_15-b04	Sun Microsystems Inc.	Windows Vista	6.1	amd64

Copyright © 1999-2011, Apache Software Foundation

Display 10. Tomcat Manager Application Fully Integrated into the SAS Environment

CONCLUSION

This paper describes the steps necessary to integrate a Web application into the SAS Enterprise BI environment using SAS AppDev Studio wizards. You have learned how to use the following main components of the SAS Web Infrastructure Platform:

- SAS Logon Manager
- SAS Logging Facility
- SAS Foundation Services
- SAS themes
- Roles and capabilities

This paper covers only the integration of a Java Web application. However, you can also integrate Web applications that use other technologies (such as Adobe Flex or Adobe Flash) into the SAS Enterprise BI environment.

RECOMMENDED READING

SAS Institute Inc. 2010. *SAS® AppDev Studio™3.4 Eclipse Plug-ins: User's Guide*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/rnd/appdev/V34/ADS34UsersGuide.pdf**.

SAS Institute Inc. 2012. SAS AppDev Studio 3.41 Developer's Site. Available at **support.sas.com/rnd/appdev/**. Accessed March 6, 2012.

SAS Institute Inc. 2011. *SAS® 9.3 Foundation Services: Administrator's Guide*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/fndsvcag/62765/PDF/default/fndsvcag.pdf**. (The comparable SAS 9.2 edition of this document is available at **support.sas.com/documentation/cdl/en/fndsvcag/61502/PDF/default/fndsvcag.pdf**.)

(continued)

SAS Institute Inc. 2011. *SAS® 9.3 Integration Technologies: Java Client Developer's Guide.* Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/itechjcdg/62762/PDF/default/itechjcdg.pdf**. (The SAS 9.2 edition of this document is available at **support.sas.com/documentation/cdl/en/itechjcdg/61499/PDF/default/itechjcdg.pdf**.)

SAS Institute Inc. 2011.*SAS® 9.3 Integration Technologies: Overview.* Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/itechov/62757/PDF/default/itechov.pdf.** (The comparable SAS 9.2 edition of this document is available at **support.sas.com/documentation/cdl/en/itechov/60309/PDF/default/itechov.pdf**.)

SAS Institute Inc. 2011 *SAS® 9.3 Intelligence Platform: Overview*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/biov/63143/PDF/default/biov.pdf**. (The comparable SAS 9.2 edition of this document is available at **support.sas.com/documentation/cdl/en/biov/63145/PDF/default/biov.pdf**.)

SAS Institute Inc. 2011. *SAS 9.3 Intelligence Platform: Web Application Administration Guide, Second Edition*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/biwaag/64769/PDF/default/biwaag.pdf**. (The comparable SAS 9.2 edition of this document is available at **support.sas.com/documentation/cdl/en/biwaag/63149/PDF/default/biwaag.pdf**.)

SAS Institute Inc. 2012. *SAS® 9.3 Intelligence Platform: Middle-Tier Administration Guide, Second Edition*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/bimtag/64862/PDF/default/bimtag.pdf**. (**Note:** This document is first available with the SAS 9.3 release.)

SAS Institute Inc. 2012. *SAS® 9.3 Intelligence Platform: System Administration Guide*. Cary, NC: SAS Institute Inc. Available at **support.sas.com/documentation/cdl/en/bisag/63132/PDF/default/bisag.pdf**. (The comparable SAS 9.2 edition of this document is available at **support.sas.com/documentation/cdl/en/bisag/64088/PDF/default/bisag.pdf**.)

SpringSource. 2012. SpringSource Documentation: Spring Framework. Available at **www.springsource.org/documentation.** Accessed March 8, 2012.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Guillaume Curat
SAS Institute Inc.
Guillaume.Curat@sas.com
support.sas.com