One Pager: Scripting support in GlassFish v3

Table of Contents

1. Introduction

This one pager describes support for JRuby based framework such as Rails and Merb , Jython/Django and Grails application support in GlassFish v3.

1.1. Project/Component Working Name

Scripting Support for GlassFish(TM) Application Server https://glassfish-scripting.dev.java.net

1.2. Name(s) and e-mail address of Document Author(s)/Supplier

Vivek Pandey, vivek.pandey@sun.com, Jacob Kessler, jacob.kessler@sun.com

1.3. Date of This Document

10/01/2009

2. Project Summary

2.1. Project Description

This project enables deployment, administration and monitoring of scripting applications, namely **JRuby on Rails, Merb**, **Groovy and Grails** and **Jython and Django**.

2.2. Risks and Assumptions

Risks:

- We are dependent on the Rails and Merb APIs (both third party) to bootstrap our automatic support for those frameworks. A change in their APIs may break compatibility with future version
- Grails community is now part of Spring and directions could change any time
- JRuby and Jython are other projects and their priorities may not align with our requirements
- Django schedule is not in our control.

Assumption: No assumption

3. Problem Summary

3.1. Problem Area

Deployment, administration and monitoring, namely **JRuby on Rails**, **Merb**, **Groovy on Grails Grails, Jython on Django**.

3.2. Justification

There are growing number of web applications/web-sites that are developed using scripting web frameworks such as Rails, Grails. It is good opportunity to position GlassFish v3 as a deployment platform for not only Java language but also for the web applications developed using scripting languages and associated framework. This also aligns with extensibility theme of v3.

4. Technical Description

4.1. Details

The following scripting web frameworks are supported through this project for v3 release:

4.1.1 Ruby Rack based frameworks: Rails, Merb, Sinatra, ...

JRuby v3 module is provided to enable deployment of applications using any Ruby Rack web framework, e.g. Rails, Merb, Sinatra etc.. The default web frameworks supported will be Rails and Merb. A Rails or Merb application can be deployed using two approaches – directory based deployment or WAR-based deployment.

- Ruby web framework (Rails, Merb...) support

  Support for any ruby based framework is enabled through Rack adapters. Rack is a spec that enables

  integration of a web server and ruby web frameworks.

  The Rack support is implemented by providing a Grizzly handler (Rack::Handler::Grizzly) and an adapter for the respective ruby based frameworks.

  For example, Rails ver 2.2 and less does not have in built Rack adapter, so JRuby connector provides a Rack adapter for Rails.

To plug in another ruby based framework, all that will be needed is to write a Rack adapter for it. Many Ruby frameworks, such as Merb, Sinatra, Campsite, Rails 2.3 etc. already have Rack support built in, and to support them all that is needed is minor glue code between the provided Rack adapter and JRuby connector.

- Directory based Deployment

  In directory based deployment, a pure Rails or Merb application is simply deployed using v3's directory based deployment. For example, this is how a Rails application will be deployed:

  ```
  asadmin deploy myRailsApp/
  ```

  The deployment and request processing happens through the integration of JRuby and JRuby connector code. The JRuby connector code involves JRuby Sniffer, Container, Deployer, GrizzlyAdapter and Ruby Rack support.

- WAR based deployment

  For WAR based deployment there is no extra work that needs to be done in v3. A Rails application WAR is created using Warbler. Warbler is a tool available as Ruby gem. It consists of a servlet Filter, a ServletContextListener and uses JRuby APIs to integrate the JRuby runtime and serves the Ruby based frameworks (Rails, Merb etc.) using Rack.

  A Ruby web application has a defined directory structures such as app, config, lib, log, vendor, tmp. Warbler puts all of these inside the WEB-INF directory of the war file. It packages any jar files inside the rails application's lib sub-directory at WEB-INF/lib. It also puts static files inside the public subdirectory of the rails application at the root directory of .war file. Details on Warbler can be found at http://caldersphere.rubyforge.org/warbler/.

- JRuby/Rails IPS package for GlassFish Updatecenter

  The **JRuby** IPS package will contain only the JRuby bundle. Rails and other useful gems such as the JDBC-MySQL gem will be available as a separate IPS package. See 4.11 for details on packaging.

4.1.2 Grails

Grails builds on top of Spring, Hibernate and Servlet technology. It provides easy to use tools to create a CRUD based web application in groovy. The mechanism by which it serves the Grails application is through Servlet and IOC support from Spring.

The Grails release along with GlassFish value adds is bundled as an IPS package and hosted at the contrib repository. It can be installed on GlassFish using the update center tool. Once installed, the Grails IPS package is installed inside the glassfish install directory.

Grails programming model defines two ways to develop and deploy the Grails applications.

4.1.2.1 Development

During development, a grails application can be run in place using the run-app command, which deploys the grails application on a jetty web server using jetty APIs.

The way to run a grails application on (embedded jetty) is:

```
grails run-app
```

The **run-app** command runs the grails application during development. The application will be deployed and run in place, and subsequent changes in the grails files, such as groovy code or gsp files, will be loaded dynamically. The reloading or re-deployment of application happens by checking to see if a recompile is required by calling back into the Grails framework. Since we are replacing the Jetty based run-app script with the one for GlassFish, shipping

jetty jars with the bundle is not required. For this reason, the bundle does not contain Jetty jars.

Starting Grails 1.1 run-app script is pluggable. A GlassFish plugin that can be installed on a Grails 1.1 or later distribution will result in to run-app script running the Grails application on GlassFish.

The Grails UpdateCenter module will be pre-installed with the GlassFish Grails plugin and run-app command will run the grails application out of the box.

GlassFish Grails plugin will be provided at Grails plugin repository. This will enable developers who are using grails distribution from grails.org to install this plugin and start running their Grails app on GlassFish.

4.1.2.2 Production deployment

For production deployment, a war file is created by running the following command in the grails application directory:

```
grails war
```

The above command will create a WAR file and this can be deployed simply by using the 'asadmin deploy' command or by using the admin console's web application deployment option. The Grails war file generated this way is packaged with all the grails dependent jar files – about 4 dozen in total. The prominent ones are – Hibernate, Spring, Grails and Groovy jars.

The v3 prelude Grails IPS package provided scripts (SharedWar.groovy) to create a smaller WAR file. This script does not package any Grails dependencies.

This will not be needed any more. Grails 1.1 introduces --nojars option with war command.

grails war --nojars, would do what **SharedWar.groovy** or **grails shared-war** command. We can provide SharedWar.groovy script even now that prints the message to use grails war --nojars command instead.

To deploy such WARs without any of the grails dependencies, Grails IPS package contains a wrapper jar - glassfish-grails.jar, which references all the Grails dependent jars in its manifest.

In GlassFish v3 Prelude release, at the deployment time, the **--libraries** option is needed with the **deploy** command. If a user does not provide --library option to indicate where the wrapper jar (glassfish-grails.jar) is, an error message is printed to tell users the correct command to type. Here is the deploy command that user would need to type:

```
asadmin deploy --libraries $GRAILS_HOME/lib/glassfish-grails.jar MyGrailsApp.war
```

In GlassFish v3 release, the **--libraries** option should not be needed. This will create a good value add for the developers, whether the Grails application WAR has dependencies or not, just one command takes care of deployment:

asadmin deploy MyGrailsApp.war.
Here are the main components we deliver with the Grails IPS package as well as part of GlassFish plugin.

- Grails EmbeddedServer interfaces implementation as mentioned as part of Grails-3524

  Used to run grails application during development using the GlassFish embedded API.

  ```
  grails run-app
  ```

- glassfish-embedded-api.jar

  Glassfish embedded API. Available thru GlassFish v3.

- SharedWar.groovy

  Prints message to use **grails war --nojars** instead.

```
grails shared-war
```

- gf-grails-connector.jar

  Grails connector code. It does some sanity check on the grails application being deployed and can provide deploy time messages to ease the deployment. For example, if a grails shared-war is deployed and the grails dependencies could not be resolved, it would throw error message to deploy the WAR using --libraries glassfish-grails.jar.

4.1.3 Jython and Django

Django is a Python based web framework. Users of GlassFish will be able to deploy Django applications using directory deployment. Jython and Django support will be enabled through Jython connector (Sniffer, Container, Deployer etc.) as for JRuby.

asadmin deploy djangoApp/

Jython/Django support will be provided as a UpdateCenter module.

4.1.4 Alignment with JavaEE 6

This document defines Scripting and JavaEE 6 alignment.

4.2. Bug/RFE Number(s)

https://glassfish.dev.java.net/servlets/ProjectIssues, select **jruby** category for JRuby related issues. For Groovy on Grails, select **groovy** and for Jython on Django select **jython** category.

4.3. In Scope

Deployment of Rails and Merb applications, deployment of Grails applications, RAD support for Grails application, deployment of Django applications.

NetBeans 6.5 supports development and deployment of Rails application. Netbeans 7.0 plans to add development and deployment of Grails application on GlassFish v3.

4.4. Out of Scope

4.5. Interfaces

4.5.1 Exported Interfaces

Interface	Stability	Former Stability (if changing)	Comments
Warbler	External		Warbler is an external tool that is used to package a Rails application as a WAR file and deploy it on any servlet container. The details of Warbler can be found here.

4.5.2 Imported interfaces

Interface	Stability	Exporting Project: Name, Specification or other Link.	Comments
	Contracted		

org.glassfish.api.*	Contracted Project Private	GlassFish	Everything from this package
org.jvnet.hk2.*	Contracted Project Private	HK2	Everything from this package
org.glassfish.embed.*	Contracted Project Private	GlassFish	Everything from this package
com.sun.grizzly.tcp.*	Contracted Project Private	Project Grizzly. Java doc can be found here	Everything from this package
org.glassfish.flashlight.*	Contracted Project Private	GlassFish	Everything from this package
org.jruby.*	Committed	JRuby	Everything from this package. JRuby lead, Charles Nutter confirmed through email exchange that org.jruby.* is committed.
Ruby Rack	external	Rack Project	Uses Rack interface using rack ruby gem

4.5.3 Other interfaces (Optional)

// Any private interfaces that may be of interest?

Interface	Stability	Exporting Project: Name, Specification or other Link.	Comments

4.6. Doc Impact

JRuby on Rails, Merb deployment, and Django deployment on GlassFish v3 tutorials. Grails on GlassFish v3 tutorial.

4.7. Admin/Config Impact

Jruby will provide container specific configuration under <jruby-container> element. This element will contain the properties that will apply to all the applications deployed inside this container.

<jruby-container/> element is not present in domain.xml by default. It gets added automatically when *cofigure-jruby-container* command is executed or a Ruby application is deployed on GlassFish. Here are details of <jruby-container/> element in domain.xml.

jruby-home

Location to the directory where JRuby is installed. It may be overriden by deployment time property jruby.home

jruby-runtime-pool
JRuby runtime pool configuration. This can be overridden using deployment time properties **jruby.runtime** or **jruby.runtime.min** or **jruby.runtime.max** as well as using asadmin CLI command: configure-jruby-container.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="jruby-runtime-pool">
  <xs:complexType>
    <xs:attribute name="initial-pool-size" type="xs:int" default="1"/>
    <xs:attribute name="min-pool-size" type="xs:int" default="1"/>
    <xs:attribute name="max-pool-size" type="xs:int" default="1"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

The container level properties can be overridden by the deployment specific properties.

For example, the container specific jruby.home setting is jruby-1.2.0 but the deployed jruby application will be using jruby-1.3.1.

```
<jruby-container jruby-home="/tools/jruby-1.2.0">
  <jruby-runtime-pool initial-pool-size='1' min-pool-size='1' max-pool-size='1'/>
</jruby-container>

<application ....>
  <engine sniffer="jruby"/>
  <property name="jruby.home" value="/tools/jruby-1.3.1"/>
</application>
```

There will be similar container element for Jython as well. The details will be provided latter as Jython support is still in POC phase.

Below are the admin functionalities required by scripting:

- Setting JRuby/Jython container specific properties
  - asadmin set/get command will be used to set or get JRuby or Jython container specific properties. This functionality is already there in asadmin CLI.
  - admin console needs to provide UI to set JRuby or Jython container specific properties. The properties are mentioned in the table below.
- Deployment time properties for JRuby and Jython application
  - deploy **--property name=value[:name=value]** option will be used with asadmin CLI. This functionality is already there in asadmin.
  - admin console needs to provide UI to set JRuby/Jython deployment time properties

- Admin UI should provide some kind of interface where JRuby can plugin any new Rack based framework deployed on GlassFish. This way, it would no more be limited to Rails, Merb but can be easily extended to other frameworks, such as Sinatra, Campsite etc.

4.7.1. Admin console

The Admingui should provide the user interface to configure JRuby (Rails, Merb etc.) and Jython/Django applications.

Admin console should also provide any other administration deploy functionalities to Jython and Django applications. For exaple, specifying contextroot, etc.

NOTE: Assuming jruby admin console will be a plugin to admin console and some of the jruby configuration aspects can be extensible without making changes in the admin console. For example, supported **applicationType**

in JRuby container is Rails, Merb. But it can grow to others over a period of time. It is expected to isolate the GUI from the content where the content is furnished by the container.

4.7.1.1 JRuby

The following are the configuration properties and the related details as they would appear inside domain.xml.

Property	Default value	Possible values	Persisted in domain.xml	Description
jruby.home	jruby.home system property if not then an error	Must be path to a directory	yes	Path to the installation to JRuby
jruby.rackEnv	development	One of development, production or test	yes	Describes the environment in which a JRuby application such as Rails or Merb would run
jruby.runtime	1	>0 <= jruby.rumtime.max >= jruby.runtime.min	yes	Initial number of JRuby runtimes to start
jruby.runtime.min	1	>0 <= jruby.runtime.max <= jrubt.runtime	yes	Minimum number of JRuby runtime in the pool
jruby.runtime.max	2	>0 >=jruby.runtime.min >=jruby.runtime	yes	Maximum number of JRuby runtime in the pool
jruby.applicationType	No default. Computed through auto detection	Either the name of a supported framework (Rails or Merb at this time) or the path to a script which will properly initialize the user's framework. Possible values: - **rails** - **merb** - **sinatra**	yes	Setting this property will bypass the normal, and potentially lengthy, auto-detection process and force deployment on the specified framework. If the deployed application is not written for the specified framework, errors will result.
jruby.MTSafe	No default value. Computed through auto	true or false	yes	This will indicate to glassfish that a framework being started using jruby-applicationType is thread-safe, and thus does not need a pool created for it. It will also affect applications started using an auto-detected user-provided startup script. If jruby-applicationType is set and jruby-MTsafe is not set or is set to false, the application will start with a pool of application instances, and each instance of the application will be accessed by one thread at a time

	detection.			be accessed by one thread at a time. This property only affects frameworks being launched where the thread safety cannot be automatically determined: setting MTsafe to true will not cause an autodetected Rails 2.1.x app to be launched in thread-safe mode, nor can it be used to force a thread-safe framework to start in pooled mode.

Below are the sample domain.xml fragment with the above properties

```
<applications>

<application directory-deployed="true"
object-type="user" enabled="true"
location="file:/myhome/vivekmz/dev/rails/uploader/"
name="uploader">

<engine sniffer="jruby"/>
<property name="jruby.home" value="/tools/j">
<property name="jruby.rackEnv" value="production">
<property name="jruby.runtime" value="1">
<property name="jruby.runtime.min" value="2">
<property name="jruby.runtime.max" value="3">


<property name="jruby.MTSafe" value="false">
<property name="jruby.applicationType" value="campsite">

</application>
</applications>
```

```
<!-- Sample jruby-container configuration -->
<jruby-container>
        <property name="jruby.home" value=""/>
        <property name="jruby.rackEnv" value="development">
        <property name="jruby.runtime" value="1">
        <property name="jruby.runtime.min" value="1">
        <property name="jruby.runtime.max" value="2">
</jruby-container>
```

4.7.1.2 Jython/Django

Jython/Django application support will be provided as a UpdateCenter module. The admin console for Jython will be provided through the Jython UC module.

Property	Default value	Possible values	Persisted in domain.xml	Description
jython.home	None. Error if not present.	Path to a directory	yes	Path to Jython installation
jython.mediaRoot	None	Path to a directory	yes	Optional parameter containing the location for Glassfish to serve static files from. If this parameter is null, Glassfish will not attempt to serve static files unless the static files are placed at the GlassFish docroot.
jython.frameworkRoot	None	Path to a	yes	Optional parameter containing the path to the framework to be used (currently Django).

jython.frameworkRoot	None	directory	yes	This will be added to the python path during app creation. Default is null.
jython.applicationType	None. Jython auto-detects supported framework (Django).	String representing application type. Possible values: **django**	yes	Optional parameter, similar to jruby.applicationType property. It can be used to specify a non-Django framework to load. This property can be used to allow plugging in other WSGI based frameworks.

4.7.2. Admin CLI

Using asadmin deploy --property option the properties mentioned in 4.7.1 can be provided to JRuby or Jython Deployer and also can be persisted in domain.xml.

4.7.2.1. configure-jruby-container

configure-jruby-container is a asadmin CLI command. The purpose of this is to configure JRuby container. The command execution results in to changes in <jruby-container> domain.xml element. The settings applies to all the ruby applications deployed unless overriden thru the deploy time properties mentioned in 4.7.1.1.

Here are options of **configure-jruby-container** command:

Option	Default	Constraint	Notes
--jruby-home	1	>0 AND >= --jruby-runtime-min AND <= --jruby-runtime-max	Path to JRuby installation directory
--jruby-runtime-min	1	>0 AND <= --jruby-runtime AND <= --jruby-runtime-max	Initial number of JRuby runtimes the container will start with.
--jruby-runtime-max	1	>0 AND >= --jruby-runtime AND >= --jruby-runtime-min	Maximum number of JRuby runtime in the pool
--show	true		Displays current settings of jruby-container

4.8. Monitoring

JRuby container monitoring is supported through the monitoring infrastructure. The JRuby Container will provide various probes and telemetry objects to emit and make the monitoring data available.

Monitoring infrastructure also needs to provide a way to convert the jruby connector exposed gfProbes as MBeans as well as exposing other MBeans coming from jruby jars in the jruby-container telemetry tree.

Refer to Monitoring onepager for details on Monitoring infrastructure.

Django/Jython is in POC phase, the monitoring details will be provided in an update to this one pager.

4.9. HA Impact

Not targetted for GlassFish v3.

4.10. I18N/L10N Impact

JRuby connector comes with the resource files for error reporting. These resource files will need to be localized.

4.11. Packaging & Delivery

- JRuby IPS package

  The package name is **jruby**. It will be hosted at the Glassfish v3 dev or final repository. It will only contain the JRuby released bundle packaged as an IPS bundle. The JRuby IPS package will depend on the Jruby-Gems IPS package. The details are below:

  - Rails IPS package

    The package name is **jruby-gems**. It will be hosted at the contrib repository. This IPS package will contain the minimum set of gems required to develop a Rails application. The gems are: Rails, Merb, and activerecord-jdbcmysql-adapter

- Grails IPS package

  The Grails IPS package name is: **grails**. It will be hosted at the contrib repository.

- Jython/Django

  Jython IPS package will be delivered on the dev repository and Django IPS package will be delivered on contrib repo.

4.12. Security Impact

None

4.13. Compatibility Impact

No requirements.

4.14. Dependencies

GlassFish API, GlassFish Embedded API, HK2 API, Grizzly API, Monitoring, JRuby API, Jython API

4.15 Architecture Review Comments

5. Reference Documents

Scripting One Pager for GlassFish v3 Prelude
GlassFish project
HK2
GlassFish Embedded API
GlassFish-Scripting project
Grizzly
JRuby

Grails

6. Schedule

Already integrated in to GlassFish v3. GlassFish v3 gem, JRuby IPS package, Grails IPS package releases will be aligned with GlassFish v3

aligned with GlassFish v3.

6.1. Projected Availability

Available with GlassFish v3 release.

7.0 ASArch comments

AsArch review March 30, 2009

Section	Comment	Resolution
4.1.2.1	Grails plugin should include glassfish distribution.	Agreed. An embedded web profile will be bundled with the GlassFish grails plugin.
4.7	Typo in '--property name:value' should be name=value	Fixed
4.5.2	Remove com.sun.enterprise.module.bootstrap.* from imported interfaces	Fixed
4.5.2	Add Rack to Imported interfaces	Fixed
4.7	--property name=value[:name=value], ':' can be a problem on windows. Admin needs to provide other separator or user needs to escape it	Email sent to ASArch

8.0 Change Log

10/01/2009

- Edited 4.7.1 to provide concrete info on <jruby-container> domain.xml element
- Added 4.7.2.1 configure-jruby-container asadmin CLI command details
- Removed monitoring data depicted in this one pager.