One Pager: Scripting support in GlassFish v3

Table of Contents

1. Introduction

Scripting languages and the associated frameworks, such as Ruby on Rails, Grails are growing in popularity. This one pager describes **Ruby on Rails** 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

1.3. Date of This Document

10/03/2008

2. Project Summary

2.1. Project Description

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

2.2. Risks and Assumptions

JRuby runtime jar is an external jar and it is used by the JRuby connector to invoke the Rails application which involves compilation of Rails libraries, Rails application code and it eventually sets some global state in the JRuby runtime and makes it non-sharable across different Rails applications deployed on GlassFish v3. Due to this limitation, the JRuby connector creates a new JRuby runtime for the each deployed Rails application.

JRuby runtime uses through JSR-223 need investigation. It needs to be seen whether JRuby runtime (without Rails) can be shared across multiple JRuby scripts using the JRuby engine through JSR-223 API. This investigation will be completed after Glassfish v3 prelude release.

3. Problem Summary

3.1. Problem Area

Deployment, administration and monitoring, HA and clustering of scripting applications , namely **JRuby on Rails** and **Grails**.

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 Prelude release:

4.1.1 Rails

JRuby v3 module is provided to enable JRuby on Rails application deployment. A Rails application can be deployed using two approaches – directory based deployment or deploying as a WAR file.

- Directory based Deployment

  In directory based deployment, a pure Rails application is simply deployed using v3's directory based deployment.

  ```
  asadmin deploy myRailsApp/
  ```

  The deployment and Rails request processing happens through integration of JRuby/Rails and v3. Rails connector code, which involves Rails Sniffer, Container, Deployer and a GrizzlyAdapter.

- WAR based deployment

  For WAR based deployment there is extra work 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 and uses JRuby APIs to integrate JRuby runtime and serves the Rails application request/response.

A Rails application has a defined directory structures such as app, config, lib, log, vendor, tmp. Warbler puts all of these inside WEB-INF directory of the war file. It also packages any jar files inside the rails application's lib sub-directory at WEB-INF/lib. It also puts 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/.

- WAR directory based deployment

  Warbler creates a WAR layout directory during WAR file creation. Users during development can deployed the WAR layout directory at tmp/war.

These are the main components for implementing Rails directory based deployment:

- GlassFish v3 Sniffer and connector

  Provides the JRuby on Rails container, deploys and serves the request

- Rails Grizzly adapter

  Receives requests/response from Grizzly layer directly and delegates these to Rails framework.

- JRuby/Rails IPS package for GlassFish Updatecenter

  The **JRuby** IPS package will contain only JRuby bundle. The Rails and other useful gems such as JDBC-MySQL gem will be available as a separate IPS package. See 4.10 for details on packaging.

- GlassFish gem

  GlassFish gem is v3 nucleus and some ruby code that bootstraps and deploys Rails application on GlassFish v3. GlassFish gem also packages asadmin command. Gem users can use asadmin command to create JDBC connection pool and JDBC resources

4.1.2 Grails

Grails application is basically a servlet application. So for deployment there is no extra work required, all it needs is the web-tier (servlet container) from inside GlassFish v3.

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

Grails IPS package includes GlassFish embedded API to allow a rapid application development experience through a groovy script. This groovy/gant script (RunApp.groovy) calls in to GlassFish embedded APIs to construct a distributed WAR and deploy/un-deploy dynamically when Grails framework requires, such as when a gsp file is modified.

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 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 Grails update center bundle provides a similar script RunApp.groovy, which translates to the same grails command **run-app** which can be used to run 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 through checking if recompile is required by calling back into Grails framework. Since we are replacing the Jetty based run-app script by the one for GlassFish, shipping jetty jars with the bundle is not required. For this reason, the bundle does not contain Jetty jars.

Following command will be used to run Grails application in embedded mode from inside Grails application directory:

```
grails run-app
```

RunAppGf.groovy uses GlassFish embedded APIs (org.glassfish.api.*) as defined in sec 4.5.2.

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 'asadmin deploy' command or using admin console's web application deployment option. The Grails war file generated this way is packaged with all the grails dependent jar files – 49 to be exact in Grails 1.0.3. The prominent ones are – Hibernate, Spring, Grails and Groovy jars.

Grails IPS package provide scripts (SharedWar.groovy) to create a smaller WAR file, it does not package any Grails dependencies. The Grails IPS package contains a wrapper jar - glassfish-grails.jar, which references all the Grails dependent jars in its manifest. To package the small jar or shared jar use **grails shared-war** command. At the time of deployment **--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, in that case error message is printed to tell users the correct command to type. Here is the deploy command that user would need to type:

Ideally we would not need specifying --libraries and expect the Grails connector to set the glassfish-grails.jar in to the classloader delegation chain or add to the DeploymentContext properties and that gets used by the web container, remember Grails application is a web web application. However this does not work currently, because there is no way to do such a thing in the current v3 infrastructure. It is expected that post GlassFish v3 prelude release, v3 infrastructure would provide some mechanism and then we would not need users to specify --libraries option during deployment.

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

Here are the main components we deliver with Grails binary distribution

- RunApp.groovy

  Used to run grails application during development using GlassFish embedded API.
  ```
  grails run-app
  ```

- glassfish-embedded-api.jar

  Glassfish embedded API. Available thru GlassFish v3.

- SharedWar.groovy

  Used to create smaller Grails application WAR file without any Grails dependent jars.

  ```
  grails shared-war
  ```

- glassfish-grails.jar

  Packaged inside $GRAILS_HOME/lib. This is a wrapper jar file that references all the Grails dependencies from it's manifest.

- gf-grails-connector.jar

  Grails connector code.

4.1.3 Alignment with JavaEE 6

JavaEE 6 would propose a way in which a JavaEE application can be composed of a non-java web application such as a Rails application. The details of what it will be and how it will be done is not known yet and will be provided when more information available. This feature is targeted for v3 Final and JavaEE6.

4.1.4 Framework to write JavaEE components using scripts

JSR-223 provides a way to execute scripts from inside Java programs. Given it is not that user friendly, a framework will be provided which will be support writing JavaEE components using the supported scripting languages such as Ruby, Groovy, Python. This feature is targeted for v3 Final or FCS.

4.2. Bug/RFE Number(s)

https://glassfish.dev.java.net/servlets/ProjectIssues, select jruby category

4.3. In Scope

Deployment of Ruby and Rails application, deployment of Grails applications, RAD support for Grails application development

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 WAR file and deploy it on any servlet container. The details of Warbler can be found here.
RunApp.groovy	Evolving		Runs a grails application on GlassFish v3.
SharedWar.groovy	Evolving		Creates Grails application WAR file without packaging the Grails dependent Jars.
asadmin	Evolving		asadmin command will be used by the GlassFish v3 gem users to create JDBC resources or to create JDBC connection pool. asadmin is an exported interface in Admin infrastructure onepager
glassfish_rails	Evolving		Glassfish v3 launcher ruby script. It launches GlassFish v3 using com.sun.enterprise.glassfish.bootstrap.ASMain(). The import dependency is mentioned in sec 4.5.2.

4.5.2 Imported interfaces

Interface	Stability	Exporting Project: Name, Specification or other Link.	Comments
org.glassfish.api.*	Contracted Project Private	GlassFish	Everything from this package
com.sun.enterprise.glassfish.bootstrap.*	Contracted Project Private	GlassFish	Everything from this package
org.jvnet.hk2.*	Contracted Project Private	HK2	Everything from this package
com.sun.enterprise.module.bootstrap.*	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.jruby.*	Committed	JRuby	Everything from this package. JRuby lead, Charles Nutter confirmed through email exchange that org.jruby.* is committed.

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 deployment on GlassFish v3 tutorial. Grails on GlassFish v3 tutorial.

4.7. Admin/Config Impact

Deploy/Un-deploy, configuration of JRuby installation, configuration of JRuby runtime pool using admin CLI as well as using admingui. Following are the requirements for rails applications administration/configuration.

4.7.1. Admin console

Admingui should provide the user interface to configure Rails applications specific properties, such as jruby installation directory, jruby runtime pool (min, max) values, RAILS_ENV value. Once user enters the specific values in the user interface, these values need to be persisted inside domain.xml and also make these available to the RailsContainer so that it can configure itself correctly. On the server restart, the v3 runtime system should make these values available to ApplicationContainer.start() or Deployer.load() methods so that the RailsContainer configures itself correctly.

Following are the configuration properties and the related details as how they would appear inside domain.xml.

- Setting JRuby install location.

  This is the entire container level property and needs to be available to all the Rails applications during deployment. RailsApplication(Container) expects a java system property: **jruby-home**

  - Possible values
    It is Rails application directory
  - Needs to be persisted in domain.xml as <jvm-options> inside <java-config> element
    For example <jvm-options>-Djruby-home=/tools/jruby</jvm-options>

- Setting RAILS_ENV (per deployed application)

  RAILS_ENV defines the environment (development, test, production) that the rails application needs to be invoked. During deployment RailsApplicationContainer sets RAILS_ENV variable for Rails framework to use and run the Rails application.

  - Possible values
    production or **development** or **test**. **development** is the default value.
  - **rails-env** is the name of the property
  - Needs to be persisted in domain.xml as

```
<applications>

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

         <engine sniffer="jruby"/>

    <property name="rails-env" value="production">
     </application>

</applications>
```

- Runtime pool (per deployed application)
  - **jruby-runtime-min** >0 (default 1)
  - **jruby-runtime-max** >=jruby-runtime-min (default 2)
  - **jruby-runtime** >0 (default 1). Initial number of runtimes to start
  - Needs to be persisted in domain.xml

```
<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-runtime" value="1">
            <property name="jruby-runtime-min" value="2">
            <property name="jruby-runtime-max" value="3">
             </application>

        </applications>
```

4.7.2. Admin CLI

There is support needed from Admin CLI to allow providing container specific options and also ways for the container to make entries in the domain.xml, for example:

- `asadmin start-domain -Djruby-home=/tools/jruby`
  The above command will result into the an entry of -Djruby-home=/tools/jruby inside jvm-options element. For example& <jvm-options>-Djruby-home=/tools/jruby</jvm-options>
- CLI deployer (asadmin deploy) should allow container specific deployment options to be passed as name value properties entries inside domain.xml, for example:
  asadmin deploy --container-properties=name:value. This syntax is just an idea. It should be defined by the admin infrastructure. These properties will be persisted inside <applications><application> element that corresponds to the Rails application being deployed as described in 4.7.1

4.8. HA Impact

Rails application HA using HADB, Im-memory replication with/without memcached.

This task will be evaluated post GlassFish v3 prelude release.

4.9. I18N/L10N Impact

4.10. Packaging & Delivery

- JRuby IPS package

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

  - Rails IPS package

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

- Grails IPS package

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

- GlassFish gem

  It is delivered to RubyForge as Ruby gem package. It is not a deliverable for GlassFish v3 prelude.

4.11. Security Impact

4.12. Compatibility Impact

No requirements.

4.13. Dependencies

GlassFish API, GlassFish Embedded API, HK2 API, Grizzly API, JRuby API

4.14 Architecture Review Comments

Following comments are from the arch review dated: 9/15/2008

Section	Review comments	Evaluation/Resolution
Section 2.2	JRuby runtime global state changes when compiling Rails application, what would it mean to using JRuby through JSR-223 APIs, can JRuby JSR 223 engine be shared across different applications accessing it through JSR 223 APIs..	This will be investigated post Glassfish v3 prelude.
Section 3.1	HA and clustering appears in the '4.8 HA Impact' but not in the 'Problem summary'.	Fixed. HA and clustering added to 3.1.
Section 4.1.1	Using Warbler, can user do directory based WAR deployment?	Yes, they can. A separate bullet is added in 4.1.1 to explain how to do it.
Section 4.1.1	How to enable uses of JDBC connection pool, JDBC resources for GlassFish gem users?	GlassFish gem ships with asadmin command available through nucleus. asadmin can be used to create the JDBC resources and JDBC connection pool.
Section 4.1.1	At present, GlassFish gem does not work as felix cache is created and if the JRuby and GlassFish gem is installed as root then users with lesser privilege can not start the gem as the install directory does not have the write permission.	Jerome is going to provide a fix
Section 4.1.1	JRuby IPS package should be split in to two. • JRuby IPS package (only JRuby no Rails or extra gems) needs to be hosted on either **dev** or **final** repository. • JRuby gems, such as Rails, ActiveRecord-JDBC, JDBC-MySQL gems will have a separate IPS package called	**Open**: Barbara to provide info to Nazrul on whether JRuby IPS package will be hosted at dev or final **Open**: Across repository dependency and install of packages needs to be tested. JRuby IPS package installation should result into install of jruby-gems IPS package too.

	package called jruby-gems and these will be hosted at the **contrib** repository.	
Section 4.1.2.1	RunApp.groovy script uses Embedded API to programmatically start,stop, deploy,undeploy GlassFish. It depends on the web container. Currently Grails IPS package contains a uber jar that contains all the v3 jars. It should not be doing it. Instead it should do one of these: 1. Either try to resolve classes from the jars available in GlassFish installation. 2. Jerome suggested using the same bootstrap mechanism ASMain.main() that is used by Glassfish gem.	Option 2 will not work due to the fact that Grails application does not have WAR directory layout, it is really scattered WAR. The other problem using Option 2 is that it does not provide programmatic way to deploy, undeploy to enable Grails iterative development. Option 1 works by resolving jars from GlassFish modules directory, hence no need to ship web-all.jar, all the classes are resolved from the GlassFish installation. Also in Option 1, Embedded API is used to create ScatteredWar and AppServer.deploy()/AppServer.undeploy() work just fine. **Resolution**: Option 1 works and will be used to support Grails application iterative development using embedded GlassFish v3
Section 4.1.2.2	1. All the Grails application dependent Jars should be created in to one OSGi bundle and installed at glassfish/applibs location.	Comment#1 needs v3 to provide applibs feature that makes the Grails OSGi jar to be available to all the deployed applications. Issue 6133 is reported to track it. Sahoo thinks this issue is invalid because there is glassfish/lib that does the same and he does not think this is any different than applibs. Assuming this is true current resolution is to install Grails OSGi jar bundle to install at `glassfish/lib`. Resolution:Creating OSGi bundle from the 49 jars is found too complext and doing it in time was risky. What was agreed on during discussion with Sahoo and Jerome that a wrapper jar that references all the Grails dependet jars thru manifest is provided and made available thru Grails connector. The web container when finally deploys the applications it finds this wrapper jar and resolves all the classes required during deployment. It was found during investigation that there is no mechanism currently in GlassFish v3 that lets you do that - either add the wrapper jar to the classloader delegation chain or add --libraries deploy option to DeploymentContext so that web container can load it. The option we provide to the users is that they provide --libraries option during deployment. Post GlassFish v3 prelude when GlassFish v3 infrastructure provides such feature then there will be no need to specify --libraries option.
Section 4.1.2.2	Grails IPS package should be hosted at **contrib** repository	Resolved.
		• Warbler is marked 'External' as it comes from external source and we

Section 4.5.1	Stability level does not appear correct	don't contribute or create it. • domain.xml dependency removed. It is not exposed to the users.
Section 4.5.2	Correct the stability levels. For external libraries, such as jruby confirmation from authors should be ok.	All the imported dependencies (except jruby.org) comes directly from GlassFish v3, hence these dependencies are marked `Contracted Project Private`. JRuby API dependency is marked `committed` after confirming this stability with Charlie.
Section 4.7.2	1. To set **jruby-home** system property, admin CLI will provide a -D option with start-domain that can be used to set Java system properties in the domain.xml as well as make this system property available to the jvm. 2. JRuby container needs deployment time properties to be passed to RailsDeployer through DeploymentContext. RailsDeployer would then persists these properties in domain.xml. `asadmin deploy` will provide an option to pass container specific properties as key value pairs.	Comment#1 needs to be implemented and the name of the option, whether it is -D or -J which can be followed by any JVM specific settings. Latter is a convention among many java based tools to set JVM specific properties. **Open**: Kedar is going to provide ability to set JVM system property while doing asadmin start-domain comment#2 **Open**: Kedar/Jerome will provide implementation of asadmin deploy option that takes name:value system property. and also the name **Open**JRuby connector needs to persist the properties it receives from DeploymentContext into domain.xml.
Section 4.7.1	admin gui is going to provide the properties set in the Admin Console and these properties will be made available to RailsDeployer via DeploymentContext. It will be responsibility of the RailsDeployer to persist these properties at the right place.	**Open**: Needs to be implemented.
Section 4.10	Provide name of the IPS packages	Fixed.

5. Reference Documents

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 v3 Prelude.

6.1. Projected Availability

Aligned with v3 Prelude.