Who We Are

> Vivek Pandey
  • GlassFish Tech lead for scripting languages support
  • http://weblogs.java.net/blog/vivekp/

> Jacob Kessler
  • GlassFish Scripting Engineer / Code Monkey
  • http://blogs.sun.com/Jacobkessler/
TS-4921: Dynamic Languages Powered By GlassFish
Agenda

> Introduction
> Dynamic Languages and the Java Virtual Machine™
> GlassFish: A Multi-Language Application Server
  • Ruby
  • Python
  • Groovy
> JavaEE™ and Dynamic Languages
> Q&A
Agenda

> Introduction
> Dynamic Languages and the Java Virtual Machine™
> GlassFish: A Multi-Language Application Server
  • Ruby
  • Python
  • Groovy
> JavaEE™ and Dynamic Languages
> Q&A
What is a Dynamic Language?

> Broadly, a language that allows a program to change its own code at runtime
  - Evaluate data as code
  - Manipulate code as data
  - Higher-order functions
  - Dynamic typing
> Often have an emphasis on readable syntax
> Lisp, Ruby, Python, Groovy, ...
What is GlassFish?

- Open Source Application Server
  - Platform for serving web applications
  - Java Enterprise Edition 5 reference implementation
    - Support for EE 6 in v3 full release
  - Modular + Extensible
  - Scalable
  - Management tools
- Traditionally for deploying Java EE applications
Why Use an Application Server?

> Serving web content well is difficult
  * Web server protocols
  * Multiple hosted sites
  * Scalability
  * Clustering/failover

> Much easier to build on an existing platform
  * Java Enterprise Edition standard
Why Use Dynamic Languages in GlassFish?

> Dynamic Languages provide advantages over Java
  
  • Bits By Friday
  • Faster development time – less overhead
  • Shorter code iterations
    • But sometimes more of them
  • Cleaner Syntax
    • Braces, Blocks, Semicolons, “fluff”

> GlassFish allows you to use those advantages while preserving Java EE features
Java vs. Ruby Code

public class Fib {
    public static void main () {
        System.out.println(fib(10));
    }
    private static int fib(int n) {
        if (n == 0) {
            return 0;
        } else if (n == 1) {
            return 1;
        } else {
            return fib(n-1)+fib(n-2);
        }
    }
}

def fib ( number )
    if number == 0
        0
    elseif number == 1
        1
    else
        fib(number - 1) +
        fib(number - 2)
    end
end
puts(fib 10)
Agenda

> Introduction
> **Dynamic Languages and the Java Virtual Machine™**
> GlassFish: A Multi-Language Application Server
  - Ruby
  - Python
  - Groovy
> JavaEE™ and Dynamic Languages
> Q&A
Dynamic Languages and the Java Virtual Machine

> Python
  • Jython interpreter provides interoperability through Java interfaces

> Ruby
  • Jruby interpreter provides automatic translation and Just-In-Time compilation

> Groovy
  • Dynamic language fully compiled to Java bytecode

> Glassfish handles all of this transparently
Advantages of Dynamic Languages

> Agile development
> More flexibility
> Metaprogramming and run-time code generation
> Domain-specific languages
Disadvantages of Dynamic Languages

> Slower execution
  * Not always an issue
  * Developer time vs. processor time

> Scalability
  * Global locks

> Maturity
Advantages of Java

- Speed
- Scalability
- Available libraries and frameworks
  - More time
  - More contributors
  - More testing
Disadvantages of Java

> Verbose syntax
  • `System.out.println("Hello world")`
  • Casting and subclasses

> Some unavailable features
  • Metaprogramming
The Right Tools for the Job

> Integration allows each language to be used for the parts of the overall project for which it is best suited

> Example: Access JMS queue from Rails

> Example: Accessing Java-based cryptography libraries from a Groovy Web application
Agenda

> Introduction
> Dynamic Languages and the Java Virtual Machine™
> **GlassFish: A Multi-Language App Server**
>   • Ruby
>   • Python
>   • Groovy
> JavaEE™ and Dynamic Languages
> Q&A
GlassFish v3
A multi-language application server

> Support for many web-frameworks
  • Grails, Rails, Merb, Sinatra, Django
>
> No prior JavaEE experience required
  • Same deploy command for all frameworks
  • No packaging (no WAR) required
  • No change in programming model
>
> Best of both worlds
  • Run dynamic language applications along side JavaEE apps
Pluggable Web Frameworks

Key
- GlassFish v3 Modules
- Web Framework Interface
- Java Framework
- Python Framework
- Ruby Framework
Embedded Interpreters

> Configured interpreter for optimized usage

> Dynamic runtime pools
  - Single threaded languages/frameworks
  - Better scaling
  - JRuby runtime pool, Jython runtime pool
Ruby scaling on native servers

Webserver/Load Balancer

- Mongrel
  - Rails

Key:
- OS Proc
- Rails
- Ruby Server

TS-4921: Dynamic Languages Powered By GlassFish
Ruby scaling on GlassFish v3

Grizzly HTTP Listener

Grizzly HTTP Listener

Grizzly HTTP Listener

JRuby Container

JRuby

JRuby

JRuby

Rails

Rails

Rails

JVM

Key

Java

Ruby
Monitoring and Administration

- GlassFish Admin console
  - Manage JavaEE as well as Ruby, Python or Groovy applications
- Or use RESTful admin services
- Transport agnostic monitoring
  - dTrace
  - JMX
  - RESTful service
- Customized monitoring
  - Write monitoring client in JavaScript and deploy as a service on GlassFish
Agenda

> Introduction
> Dynamic Languages and the Java Virtual Machine™
> GlassFish: A Multi-Language App Server
  • Ruby
  • Python
  • Groovy
> JavaEE™ and Dynamic Languages
> Q&A
JRuby

Ruby

ramaze merb Sinatra rails
Ruby support on GlassFish

> Rack compliant
  - Wires a web server to a Ruby application
  - Based on WSGI from Python
> Run any Ruby application that supports Rack

```ruby
class HelloWorld
  def call(env)
    [200, {'Content-Type' => 'text/plain'}, ['Hello World!']]
  end
end
```
Ruby support on GlassFish

> All major Ruby frameworks support Rack
  • Rails (Starting 2.3.2), Merb, Sinatra, Campsite...
> GlassFish has in built support for Rails, Merb and Sinatra
> Or you can run your own Ruby framework...
Pluggable Framework support

> Rack-up script
> `jruby.applicationType` deployment property
> GlassFish auto-detects your application
Ruby Monitoring

What can be monitored in Ruby/Rack applications?

- HTTP stats
- JRuby interpreter stats
- JRuby runtime pool stats

Monitor using

- dTrace or JMX or your custom script or REST
NetBeans support
Develop, deploy and debug

```ruby
# GET /books/1
# GET /books/1.xml
def show
  @book = Book.find(params[:id])
  respond_to do |format|
    format.html # show.html.erb
    format.xml { render :xml => @book }
  end
end
end

# GET /books/new
# GET /books/new.xml
def new
  @book = Book.new
  respond_to do |format|
    format.html # new.html.erb
    format.xml { render :xml => @book }
  end
end

# GET /books/1/edit
def edit
  @book = Book.find(params[:id])
end
```

### Watches

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>@flash</td>
<td>ActionController::Flash::FlashHash</td>
<td>Empty ActionController::Flash::FlashHash</td>
</tr>
<tr>
<td>@headers</td>
<td>Hash</td>
<td>Hash (2 elements)</td>
</tr>
<tr>
<td>@params</td>
<td>HashWithIndifferentAccess</td>
<td>HashWithIndifferentAccess</td>
</tr>
<tr>
<td>'controller'</td>
<td>String</td>
<td>books</td>
</tr>
<tr>
<td>'action'</td>
<td>String</td>
<td>new</td>
</tr>
<tr>
<td>@request</td>
<td>ActionController::CgiRequest</td>
<td>#<a href="">ActionController::CgiRequest</a></td>
</tr>
<tr>
<td>@response</td>
<td>ActionController::CgiResponse</td>
<td>#<a href="">ActionController::CgiResponse</a></td>
</tr>
<tr>
<td>CGI::Session</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GlassFish gem

> Based on GlassFish v3 Kernel
> Just about 2.5 MB in size
> Ruby friendly
> More info
  • http://glassfishgem.rubyforge.org/
Using GlassFish gem

> How simple can it get?
Using GlassFish gem

> How simple can it get?

```bash
$ gem install glassfish
```
Using GlassFish gem

> How simple can it get?

```
$ gem install glassfish

$ glassfish -e production
```
Demo
Ruby RESTful service on GlassFish
Agenda

> Introduction
> Dynamic Languages and the Java Virtual Machine™
> GlassFish: A Multi-Language App Server
  • Ruby
  • Python
  • Groovy
> JavaEE™ and Dynamic Languages
> Q&A
Python
(J)Python and Django on GlassFish

> Jython as Python interpreter
> Jython container
  • Like JRuby, extends GlassFish
  • Jython Grizzly Adapter for HTTP processing
  • WSGI support makes other Python frameworks simple to plugin
> Simple deployment

$ asadmin deploy myDjangoApp/
Demo
Django application deployment on GlassFish v3
Agenda

> Introduction
> Dynamic Languages and the Java Virtual Machine™
> GlassFish: A Multi-Language App Server
  • Ruby
  • Python
  • Groovy
> JavaEE™ and Dynamic Languages
> Q&A
Groovy and Grails
Grails Application Deployment

> WAR deployment
  • grails war
    • Creates grailsApp.war (with ~48 jars and ~18MB in size)

> Efficient deployment
  • grails shared-war
    • Creates grailsApp.war but only ~200KB
    • Available through Grails package on GlassFish v3 update center
GlassFish Grails Plugin

> Embedded GlassFish for Grails
> Same server for development and production
> Supports Grails run-app and run-war command

> Install GlassFish as Grails plugin

  $ grails install-plugin glassfish --global

> Now run your application on GlassFish

  $ grails run-app
Agenda

- Introduction
- Dynamic Languages and the Java Virtual Machine™
- GlassFish: A Multi-Language App Server
  - Ruby
  - Python
  - Groovy
- JavaEE™ and Dynamic Languages
- Q&A
JavaEE and Dynamic Languages

> JavaEE is a mature and robust enterprise platform
  
  • Servlet, EJB, JMS, JAX-RS, JAX-WS...

> Hybrid Rails and Servlet/JSP application
  
  • Forward to and from Rails to Servlet/JSP
  • HttpSession as Rails session
  • ServletRequest/Response API available to Rails

> JDBC connection pool
  
  • Efficient Database access

> JMS
  
  • Async messaging in Rails
Servlet 3.0 pluggability

> JavaEE 6 and Dynamic Languages
  - No boilerplate web.xml
    - Dynamically add ServletFilter and ServletContextListner

> Simple packaging
  - No need to package as WAR
  - Hence no Warbler needed
  - Place Rails application inside WEB-INF directory
Demo: Rails and JavaEE
Scripting Sessions and BOF

- PAN-5348: Script Bowl 2009: A Scripting Languages Shootout
- TS-5413: JRuby on Rails in Production: Lessons Learned from Operating a Live, Real-World Site
- TS-5033: Scripting Java™ Technology with JRuby
- BOF-4434: Hacking JRuby
- TS-5216: Toward a Renaissance VM
- TS-4955: Comparing Groovy and JRuby
- TS-5015: Welcome to Ruby
- BOF-5058: JRuby Experiences in the Real World
- Scripting POD#566
Q&A
Resources

- http://glassfish-scripting.dev.java.net/
- http://glassfishgem.rubyforge.org/
- http://glassfish.dev.java.net/
- http://jruby.org
- http://jython.org
- http://grails.org

- Mailing List
  - users@glassfish.dev.java.net

- Issue Tracker
  - https://glassfish.dev.java.net/servlets/ProjectIssues
Thank You

Vivek Pandey
Vivek.Pandey@Sun.COM

Jacob Kessler
Jacob.Kessler@Sun.COM
GlassFish Community
Open Source and Enterprise Ready

- **GlassFish V3 Preview Available now!**
  - Java EE 6 reference implementation
  - Modular OSGi architecture – easy to develop & deploy
  - Runs in-process and easy to extend
  - Support for Ruby-on-Rails, Groovy and Grails, Python and Django
- **GlassFish V2 – Production Ready**
  - Best price/performance open source App server with Clustering, High Availability, Load Balancing
  - Secure, Reliable, Transactional, .NET-interop Web svcs
  - Support for Ajax and Comet
- **GlassFish ESB**
  - SOA and Business Integration platform
- **GlassFish Communications App Server**
  - SIP servlet technology for converged services

- 24x7 Enterprise and Mission Critical Support
  - sun.com/appserver
- **Tools Integration**
  - NetBeans and Eclipse
- **Pavilion booth numbers: 550, 566, 567**
- **Meet Java EE spec leads and experts at Ancillary Event & Booth**

glassfish.org