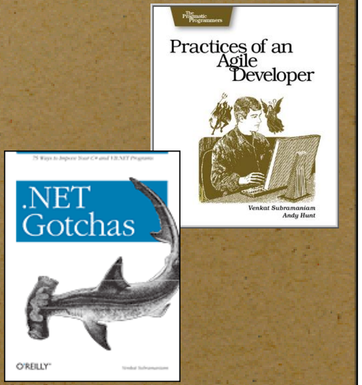
Spring into Groovy

```
spkr.name = 'Venkat Subramaniam'
spkr.company = 'Agile Developer, Inc.'
spkr.credentials = %w{Programmer Trainer Author}
spkr.blog = 'agiledeveloper.com/blog'
spkr.email = 'venkats@agiledeveloper.com'
```



Abstract

- What do you get when you mix an agile, object-oriented, dynamic language with a lightweight, flexible, and extensible framework? You get a *groovier Spring*! Spring allows you to develop using Groovy as much as Java. Groovy brings some neat concepts to the Java Platform that is hard to realize directly through the Java language. Using these capabilities can lead to elegant and easier Spring development.

- In this presentation we will cover topics including: Strengths of Groovy, Using Groovy in Spring, Configuration, Bean Development, Deployment, How it fits in

Agenda

- Spring in a minute
- Groovy in a minute
- Dynamic Languages
- Groovy in Spring
- Configuring
- Refreshable
- Embedding
- Conclusion

Spring in a Minute

- Spring is a light-weight ~~*Java*~~ framework
- Supports Inversion of Control—Dependency Injection
- Component development and integration
- Rich class library
- Enterprise Application development
- Widely used to ease development effort

It’s all about Bean

- Spring Bean is
  - a ~~Java~~ class
  - has properties
  - behavior may be specified declaratively

Agenda

- Spring in a minute
- Groovy in a minute
- Dynamic Languages
- Groovy in Spring

- Configuring
- Refreshable
- Embedding
- Conclusion

Groovy in a Minute

- Dictionary says: Marvelous, Excellent, Hip
- Power of Scripting language on JVM
  - Dynamic, agile, OO
  - Derives strengths from Smalltalk, Pyton, Ruby
- Alternate to Java for small to medium size projects
- Great to write unit tests
- Can generate Byte code—You can mix with Java

What makes Groovy Attractive?

- Groovy is sometimes called the glue language
- Useful for
  - writing unit tests
  - rapid prototyping
  - small to medium sized projects
  - expressing business rules
  - ...

Why not mix a light-weight language
with a light-weight framework?

Agenda

- Spring in a minute
- Groovy in a minute
- Dynamic Languages
- Groovy in Spring
- Configuring
- Refreshable
- Embedding
- Conclusion

Spring 2.0 Breaks Language Barrier

- You can implement your component in any language that runs on JVM
- JSR 223 has provided facility to make Java platform truly One VM—Multiple Languages
- This allows you to choose language that makes most sense for your need on Java 6
- Spring can already do that in 2.0

Spring 2.0 Dynamic Languages

- Currently supports
  - Groovy
  - JRuby
  - BeanShell
- Is Extensible to use other languages
- Constructor based injection not possible currently
- Only setter based injection is allowed (mostly this is preferred anyway)

Agenda

- Spring in a minute
- Groovy in a minute
- Dynamic Languages
- Groovy in Spring

- Configuring
- Refreshable
- Embedding
- Conclusion

How to use Groovy?

- Compile Groovy and use it like you would a Java Bean
  - 🛑 Stop revolting, there are other interesting ways!
- OK, you can enjoy the Groovy script directly with no compilation
  - Provides some interesting capabilities

What does it take?

- Very little!
- Choose to implement any Java interface
  - Serves as contract between consumers and the bean
- Bean definition conceals the implementation language

Beans and Scripts

- At least one bean definition per dynamic language source file
- Multiple bean definitions may refer to the same dynamic language source file

Scripting Factory

- Spring Beans may be written in different languages
- org.springframework.scripting package allows for different scripts to be plugged in
- org.springframework.scripting.groovy's GroovyScriptFactory is pertinent to Groovy
- You can configure Spring factory to use Groovy
- You can configure your bean to use a Groovy object

Classpath

- Dependencies
- Needs the following jars in classpath
    - groovy-1.0-jsr-04.jar
    - asm-2.2.2.jar
    - antlr-2.7.6.jar
- Use appropriate current version as you need

Agenda

- Spring in a minute
- Groovy in a minute
- Dynamic Languages
- Groovy in Spring

- Configuring
- Refreshable
- Embedding
- Conclusion

Configuring

- Quite a few options for configuring
  - Good old Bean Definition
    - ❖ Once you compile Groovy,...
  - GroovyScriptFactory may define the source and location
  - Use <lang:groovy>
    - or <lang:jruby> or <lang:bsh>

Configuring using <bean>

```
package com.agiledeveloper;

public class GroovyGreeter implements Greeter
{
    def String greetings

    def String getMessage()
    {
        greetings + " from Groovy";
    }
}
```

Compiled into
GroovyGreeter.class

```
package com.agiledeveloper;

public interface Greeter
{
    public String getMessage();
}
```

Configuring using <bean>

```xml
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:tx="http://www.springframework.org/schema/tx"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
         http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
          http://www.springframework.org/schema/aop
         http://www.springframework.org/schema/aop/spring-aop-2.0.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-2.0.xsd">

     <bean id="javagreeter" class="com.agiledeveloper.JavaGreeter">
        <property name="greetings" value="Hello" />
     </bean>

      <bean id="groovygreeter" class="com.agiledeveloper.GroovyGreeter">
        <property name="greetings" value="Howdy" />
      </bean>
</beans>
```

Using the Groovy Bean

```
package com.agiledeveloper;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class UseGreeter
{
    public static void main(String[] args)
    {
        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationcontext.xml");

        useGreeter(context, "javagreeter");
        useGreeter(context, "groovygreeter");
    }

    private static void useGreeter(ApplicationContext context, String beanID)
    {
        Greeter service = (Greeter) context.getBean(beanID);
        System.out.println(service.getMessage());
    }
```

Configuring using GroovyScriptFactory

```
...
   <bean
    class="org.springframework.scripting.support.ScriptFactoryPostProcessor" />

   <bean id="groovygreeter_viafactory"
        class="org.springframework.scripting.groovy.GroovyScriptFactory">
            <constructor-arg
                    value="classpath:com/agiledeveloper/GroovyGreeter2.groovy" />
      <property name="greetings" value="Hi" />
   </bean>
</beans>
```

Ensures dynamic
bean definitions
processed at startup

scripts directory that contains the subdirectories com/agiledeveloper with groovy script is in classpath

Configuring using <lang:groovy>

```
...<beans ...
        xmlns:lang="http://www.springframework.org/schema/lang"
        xsi:schemaLocation="...Rest of the Schema Junk Goes Here...
         http://www.springframework.org/schema/lang http://
        www.springframework.org/schema/lang/spring-lang-2.0.xsd" >


...
   <lang:groovy id="grooviergreeter"
        script-source="classpath:com/agiledeveloper/GroovierGreeter.groovy">
            <lang:property name="greetings" value="Groovy" />
   </lang:groovy>
</beans>
```

Agenda

- Spring in a minute
- Groovy in a minute
- Dynamic Languages
- Groovy in Spring
- Configuring
- ✹ Refreshable
- Embedding
- Conclusion

Refreshable Beans

- A scripted bean may be monitored for change and reloaded
- No need to restart or redeploy your application
- Spring creates a proxy for the target object
- Spring AOP TargetSource gets instance before each call
- `Refreshable` interface facilitates programmatic refresh

Refreshable Beans

- Turned off by default
- <lang:groovy>'s refresh-check-delay attribute does the trick
  - defines a delay, in milliseconds, before which the bean is loaded
- Change is picked up only when a method is called on the bean (not right after the change)
- If you mess up the bean, it will result in fatal error
- Not applicable to in-lined scripts

Configuring Refresh

```
<lang:groovy id="grooviergreeter" refresh-check-delay="2"
        script-source="classpath:com/agiledeveloper/GroovierGreeter.groovy">
    <lang:property name="greetings" value="Groovy" />
</lang:groovy>
```

Agenda

- Spring in a minute
- Groovy in a minute
- Dynamic Languages
- Groovy in Spring
- Configuring
- Refreshable
- Embedding
- Conclusion

Embedded Beans

- You may embed your bean in the configuration if you like
- May not desirable in general
- May come in handy to throw in a little validator, for example
- You may want to place your inline script in a CDATA section if it contains "weird" symbols (symbols that will cause allergy for XML parser)

Configuring Inline

```
<lang:groovy id="embeddedgreeter">
    <lang:inline-script>
        package com.agiledeveloper;

        public class EmbeddedGreeter implements Greeter
        {
            def String greetings

             def String getMessage()
             {
                 greetings + " from EmbeddedGroovy"
             }
        }
    </lang:inline-script>
      <lang:property name="greetings" value="Hey" />
 </lang:groovy>
```

With XML comes...

```
<lang:groovy id="embeddedgreeter">
  <lang:inline-script>
    package com.agiledeveloper;

    public class EmbeddedGreeter implements Greeter
    {
      def String greetings

       def String getMessage()
       {
         if (greetings.compareTo("Hello there") < 0)
         {
           greetings = "Hello there!"
         }

         greetings + " from EmbeddedGroovy"
       }
    }
  </lang:inline-script>
   <lang:property name="greetings" value="Hey" />
</lang:groovy>
```

SAXParseException

Concealed Groovy...

```
<lang:groovy id="embeddedgreeter">
   <lang:inline-script><![CDATA[
      package com.agiledeveloper;

      public class EmbeddedGreeter implements Greeter
      {
         def String greetings

          def String getMessage()
          {
             if (greetings.compareTo("Hello there") < 0)
             {
               greetings = "Hello there!"
             }

             greetings + " from EmbeddedGroovy"
          }
      }
   ]]></lang:inline-script>
     <lang:property name="greetings" value="Hey" />
</lang:groovy>
```

Agenda

- Spring in a minute
- Groovy in a minute
- Dynamic Languages
- Groovy in Spring
- Configuring
- Refreshable
- Embedding
- Conclusion

Quiz Time

References

- http://www.springframework.org/docs/reference/dynamic-language.html
- http://groovy.codehaus.org

Thank You!

http://www.agiledeveloper.com — download