Groovy And Grails

An Overview

Groovy

What Is Groovy?

- Groovy…
  - Is A Dynamic Language For The Java Virtual Machine (JVM)
  - Takes inspiration from Smalltalk, Python and Ruby (etc...)
  - Integrates with the Java language and platform at every level

Sugar In Your Java

- Groovy is Java-like
  - Easy to learn for a Java developer

    → **flat learning curve**
  - Simpler than Java for beginners and **subject matter experts**

- Seamless integration with Java
  - You can **mix Groovy and Java** objects together
    - Groovy class extending Java class implementing Groovy interface, and vice versa…
  - Same strings, regex, APIs, OO, threads, security

Groovy Class

```
class GroovyPerson {

    // dynamically typed property
    def age

    // statically typed property
    String name

    def printName() {
        println name
    }

    static void main(String[] args) {
        def person = new GroovyPerson(age: 8,
                                      name: 'Jake')
        person.printName()
    }
}
```

Groovy Beans

- Groovy Beans / POGOs
- Similar To POJOs
  - ...but groovier
- Eliminates Boilerplate Code
- Simple To Achieve JavaBean Compliance

POJO

```
public class Person {

    private String firstName;
    private String lastName;

    public Person() {
    }

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }
```

POJO

```
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getFirstName() {
        return firstName;
    }
}
```

POJO

- Modern Java IDEs Generate Most Of That Code
  - developer declares fields
  - IDE generates constructors
  - IDE generates getters/setters

If the IDE can generate all of that code, why can't the compiler or the runtime?

Groovy Beans

- Groovy Beans Eliminate All Of The Boilerplate Code
- No Need To Write Getters/Setters
- Seldom Need To Write Constructors

Groovy Beans

```
class BaseballTeam {
    def cityName
    def teamName
}
```

```
def myTeam = new BaseballTeam(teamName: 'Cardinals',
                              cityName: 'St. Louis')

println myTeam.teamName
println myTeam.cityName
```

Groovy Beans

- Property Access Looks Like Field

```
def myTeam = new BaseballTeam()

// myTeam.setTeamName('Cardinals')
myTeam.teamName = 'Cardinals'

// myTeam.setCityName('St. Louis')
myTeam.cityName = 'St. Louis'

// println myTeam.getTeamName()
println myTeam.teamName
```

Print Independence Day

```
// PrintIndependenceDay.java
import java.util.Calendar;
import java.util.Date;

public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.clear();
        calendar.set(Calendar.MONTH, Calendar.JULY);
        calendar.set(Calendar.DATE, 4);
        calendar.set(Calendar.YEAR, 1776);
        Date time = calendar.getTime();
        System.out.println(time);
    }
}
```

Print Independence Day

```
// PrintIndependenceDay.groovy
import java.util.Calendar;
import java.util.Date;

public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.clear();
        calendar.set(Calendar.MONTH, Calendar.JULY);
        calendar.set(Calendar.DATE, 4);
        calendar.set(Calendar.YEAR, 1776);
        Date time = calendar.getTime();
        System.out.println(time);
    }
}
```

No Utility Imports...

```
// PrintIndependenceDay.groovy

public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance();
        calendar.clear();
        calendar.set(Calendar.MONTH, Calendar.JULY);
        calendar.set(Calendar.DATE, 4);
        calendar.set(Calendar.YEAR, 1776);
        Date time = calendar.getTime();
        System.out.println(time);
    }
}
```

No Semicolons...

```
// PrintIndependenceDay.groovy
public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.getInstance()
        calendar.clear()
        calendar.set(Calendar.MONTH, Calendar.JULY)
        calendar.set(Calendar.DATE, 4)
        calendar.set(Calendar.YEAR, 1776)
        Date time = calendar.getTime()
        System.out.println(time)
    }
}
```

No Getters...

```
// PrintIndependenceDay.groovy
public class PrintIndependenceDay {

    public static void main(String[] args) {
        Calendar calendar = Calendar.instance
        calendar.clear()
        calendar.set(Calendar.MONTH, Calendar.JULY)
        calendar.set(Calendar.DATE, 4)
        calendar.set(Calendar.YEAR, 1776)
        Date time = calendar.time
        System.out.println(time)
    }
}
```

No Static Typing...

```
// PrintIndependenceDay.groovy
public class PrintIndependenceDay {

    public static void main(String[] args) {
        def calendar = Calendar.instance
        calendar.clear()
        calendar.set(Calendar.MONTH, Calendar.JULY)
        calendar.set(Calendar.DATE, 4)
        calendar.set(Calendar.YEAR, 1776)
        def time = calendar.time
        System.out.println(time)
    }
}
```

No System.out.blah.blah...

```
// PrintIndependenceDay.groovy
public class PrintIndependenceDay {

    public static void main(String[] args) {
        def calendar = Calendar.instance
        calendar.clear()
        calendar.set(Calendar.MONTH, Calendar.JULY)
        calendar.set(Calendar.DATE, 4)
        calendar.set(Calendar.YEAR, 1776)
        def time = calendar.time
        println(time)
    }
}
```

No Class...

```groovy
// PrintIndependenceDay.groovy

def calendar = Calendar.instance
calendar.clear()
calendar.set(Calendar.MONTH, Calendar.JULY)
calendar.set(Calendar.DATE, 4)
calendar.set(Calendar.YEAR, 1776)

def time = calendar.time

println(time)
```

Optional Parens...

```
// PrintIndependenceDay.groovy

def calendar = Calendar.instance
calendar.clear()
calendar.set Calendar.MONTH, Calendar.JULY
calendar.set Calendar.DATE, 4
calendar.set Calendar.YEAR, 1776

def time = calendar.time

println time
```

Lets Go Meta...

```groovy
// PrintIndependenceDay.groovy

def calendar = Calendar.instance
calendar.with {
    clear()
    set MONTH, JULY
    set DATE, 4
    set YEAR, 1776
    println time
}
```

Lets Compare...

```java
// PrintIndependenceDay.java
import java.util.Calendar;
import java.util.Date;

public class PrintIndependenceDay {

  public static void main(String[] args) {
    Calendar calendar = Calendar.getInstance();
    calendar.clear();
    calendar.set(Calendar.MONTH, Calendar.JULY);
    calendar.set(Calendar.DATE, 4);
    calendar.set(Calendar.YEAR, 1776);
    Date time = calendar.getTime();
    System.out.println(time);
  }
}
```

```groovy
// PrintIndependenceDay.groovy

def calendar = Calendar.instance
calendar.with {
    clear()
    set MONTH, JULY
    set DATE, 4
    set YEAR, 1776
    println time
}
```

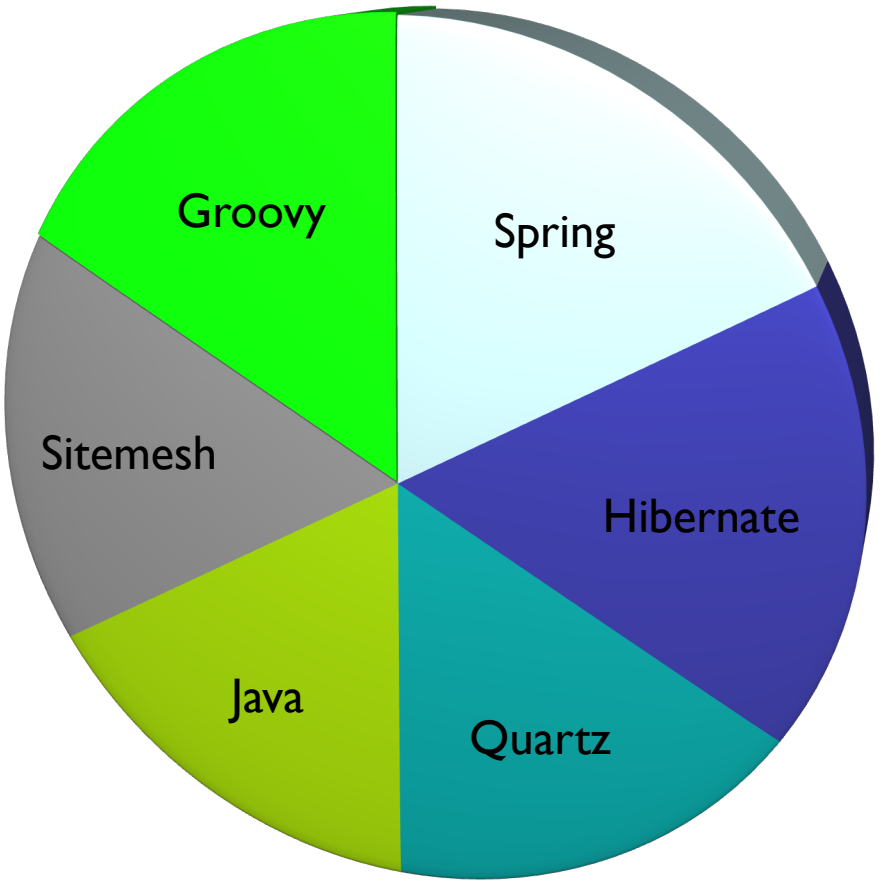
Grails

What Is Grails?

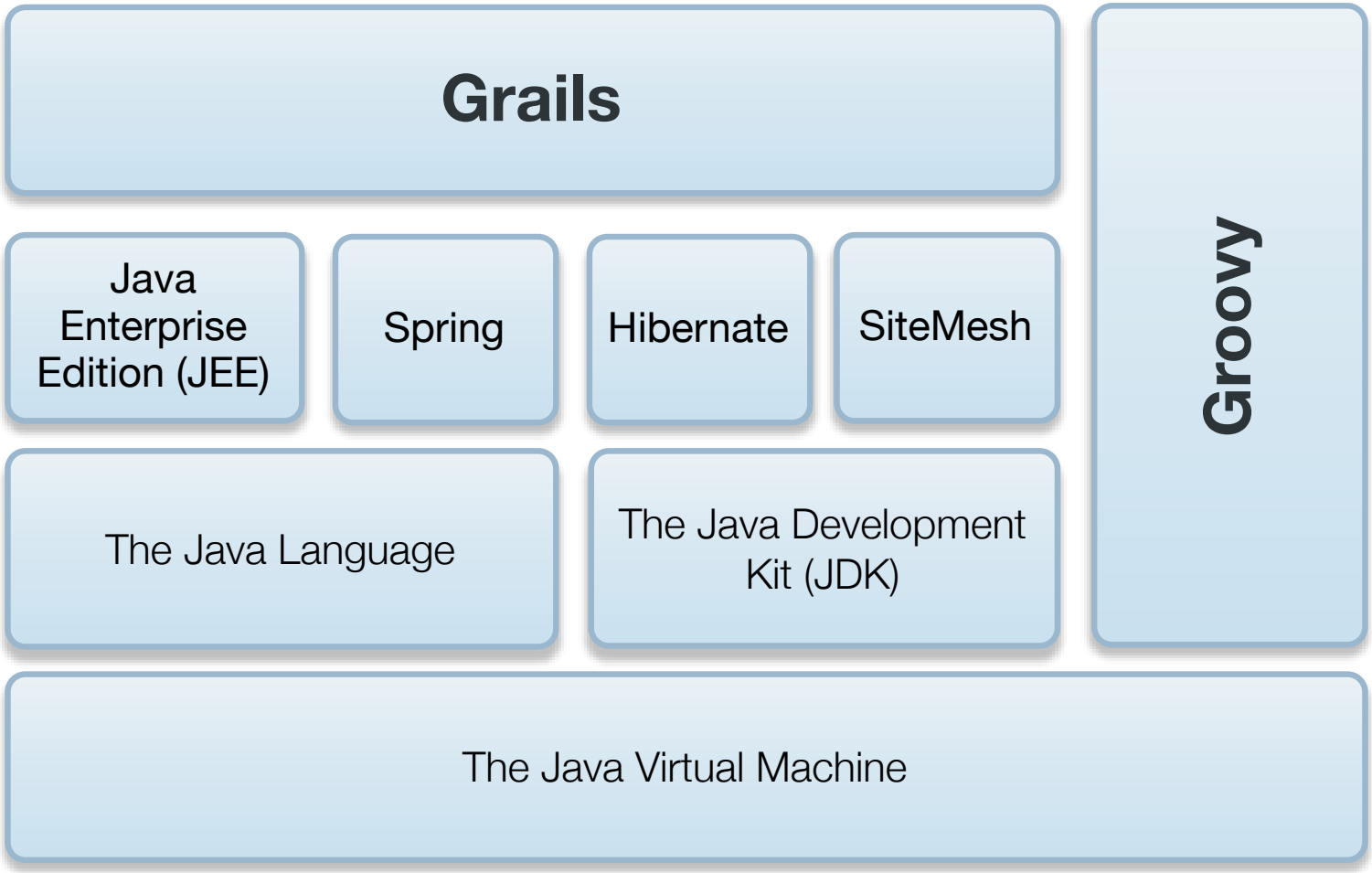
- A Web platform that implements the full stack from build system down to ORM layer
- Leverages existing technologies like Spring, Hibernate, Quartz etc. avoiding re-inventing the wheel
- Features and extensible plug-in system and an environment for runtime configuration built on Spring

Best Of Breed

- Spring
- Hibernate
- Groovy
- Quartz
- Sitemesh
- Tomcat
- Ant



The Grails Stack



More than Just a Web Framework

- Grails delivers more than your regular web framework - a full stack.
- Grails aims to ease development of every tier and features
  - An integrated Groovy build system
  - An incredibly simple ORM layer built on Hibernate
  - An amazing Groovy-based view technology called GSP

Sensible Defaults

- Quickly get started
  - An in-memory HSQLDB
  - A built-in Tomcat servlet container
  - The ability to generate a WAR file out of the box
  - A built-in interactive console and shell
  - An ant build.xml file with useful targets like war, test etc.
  - IDE project files

Querying

- GORM supports a number of ways to query including:
  - Dynamic Finders
  - Criteria
  - Query-by-example
  - HQL

Dynamic Finders

- Automatically translate the properties of the class into "method expressions" - at runtime!
- Uses the Hibernate Criteria API underneath
- Rich and expressive way to query

Dynamic Finders

```
def all = Bookmark.list()

// user like expressions
def grailsBookmarks =
   Bookmark.findAllByTitleLike("%Grails%")

// query between two values
def now = new Date()
def lastWeeks =
    Bookmark.findByCreatedDateBetween(now-7, now)

// query associations
def bookmark = Bookmark.get(34)
def comments =  Comment.findAllByBookmark(bookmark)
```

Querying With Criteria

```
// returns first 10 users who have an active
// account that has been created in the last
// 30 days and that have Grails-like
// bookmarks created in the last 7 days
def now = new Date()
def users = Bookmark.withCriteria {
    comments {
        like("text","%Grails%")
        between("dateCreated", now-7, now)
    }
    between("dateCreated",now-30, now)
    maxResults(10)
}
```

Querying with HQL

- If all else fails, there is always HQL!:

```
// Query for Bookmark instances
def bookmarks =
 Bookmark.findAll("from Bookmark b where b.title like ?", ["%Grails%"] )

// select only the Bookmark titles
def titles =
  Bookmark.executeQuery(
    "select b.title from Bookmark b where b.title like ?",
    ["%Groovy%"] )
```

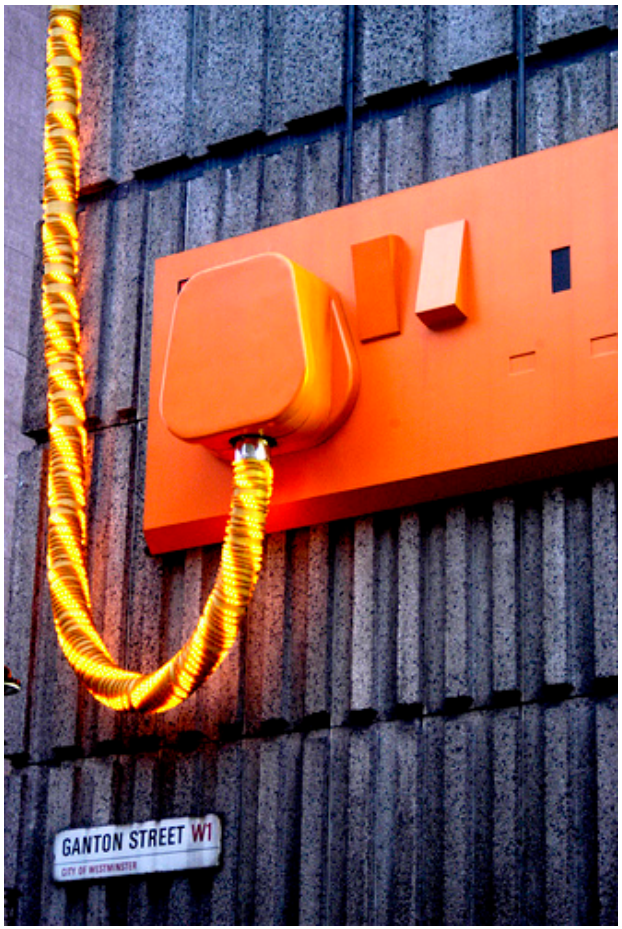
The Grails Plugin System

The Background

- Grails is designed to wire together different libraries and make them easy to use
- In this sense it can be seen as a "platform for runtime configuration"
- De-coupling those components was hard without a well defined system

The Extension Points

- The Build System
- Spring Application Context
- Dynamic method registration
- Auto Reloading
- Container Config



What can a Plug-in do?

- Add new methods, constructors, properties etc. to any class at runtime
- Perform runtime Spring configuration
- Modify web.xml on the fly
- Add new controllers, tag libraries etc.

A Look Ahead

- 1.2 Is Just Around The Corner
- Performance, Stability, New Features Etc...
- Focus On Plugins
- Cloud Foundry Etc...

Q & A