

Building Grails Applications with PostgreSQL

Brent Baxter and Ken Rimple
PostgreSQL East - March 25, 2010

About Us

- Brent Baxter (bbaxter@chariotsolutions.com)
 - ▶ Consultant and Applications Architect
 - ▶ Grails, Java, and Spring developer
- Ken Rimple (krimple@chariotsolutions.com)
 - ▶ Head of Education Services
 - ▶ Host, Chariot TechCast
(techcast.chariotsolutions.com)

Goals

- Provide a basic overview of Groovy and the Grails framework
- Review some of the features of Grails Object Relational Modeling (GORM)
- Demonstrate GORM with PostgreSQL

Agenda

- Introduction to Groovy
- Introduction to Grails
 - ▶ Components of a Grails application
- GORM
 - ▶ How to use GORM
 - ▶ GORM with PostgreSQL

Introduction to Groovy



Groovy

- A dynamic language built for the Java Virtual Machine
 - ▶ Compiles to native ByteCode
 - ▶ Can inherit Groovy classes from Java classes (and vice versa)
 - ▶ Native access to any Java library



Benefits of Groovy

- Adds "Convention over Configuration"
 - ▶ Groovy beans are VERY simple
 - ▶ Assumes methods are public, member variables are private
 - ▶ Constructors not required
 - ▶ Easy collections – Use the [] syntax for lists and maps

Groovy APIs

- xml – Powerful XML Parsing, Creation
- gsql – Simplified JDBC API
- Groovy Testing– simple testing API, mocking with closures, etc...
- Collections – lists, maps and ranges
- builders – a set of repetitive building tasks can be automated using various Groovy Builders
- Swing – the Groovy SwingBuilder makes UI easy. Also Griffon is a Groovy-based MVC framework for Swing (like Grails for Web)

Java Person Class

```
public class Person {  
    private String ssn;  
    private String lastName;  
    private String firstName;  
  
    public Voter(String ssn, String firstName,  
                String lastname) {  
        ...  
    }  
  
    public void setSsn() { ...}  
    public String getSsn() { ...}  
  
    // etc...  
}
```

Groovy Person Class

Groovy Person Class

```
class Person {  
    String ssn  
    String lastName  
    String firstName  
}
```

Constructors and Lists

Constructors and Lists

```
// Groovy provides constructors for free
def p = new Person(ssn:"123-45-6789",
    firstName:"Jack",
    lastName:"Beanstalk")
def p2 = new Person(ssn:"123-45-6789")
```


Constructors and Lists

```
// Groovy provides constructors for free
def p = new Person(ssn:"123-45-6789",
    firstName:"Jack",
    lastName:"Beanstalk")
def p2 = new Person(ssn:"123-45-6789")
```

```
// ArrayLists are simple
def persons = [ new Person(ssn:"234..."),
    new Person(ssn:"235...") ]

persons += new Person(ssn:"234-333-4444")
```

Groovy SQL Support

Groovy SQL Support

```
// Using JDBC
```

```
def sql = Sql.newInstance(url, usr, pwd, driver)
sql.execute("insert into table values ($foo, $bar)")
sql.execute("insert into table values (?,?)", [a,b])
sql.eachRow("select * from USER") {print it.name}
def list = sql.rows("select * from USER")
```

Introduction to Grails



Grails Framework

- A web application framework
 - ▶ Three-tiered Model View Controller (MVC)
- Convention based
 - ▶ Sensible default configuration or use any of a number of simple DSLs
- Extendable with a rich set of plug-ins

Grails Is Groovy

- Leverages the Groovy programming language
 - ▶ Groovy Services and Beans
- Configuration DSLs
 - ▶ Logging, data source configuration, dependency resolution, ...
- Web tier Groovy Server Pages (GSP)

Grails Is Java

- Framework backed by best of breed industry standards
 - ▶ Spring, Hibernate, SiteMesh, Web Flow, etc.
- Leverage any Java library, access from Groovy or Java classes
- Can be deployed as a Java web application (WAR) to any Servlet container

Grails Technology Stack

Web UI / Web Services

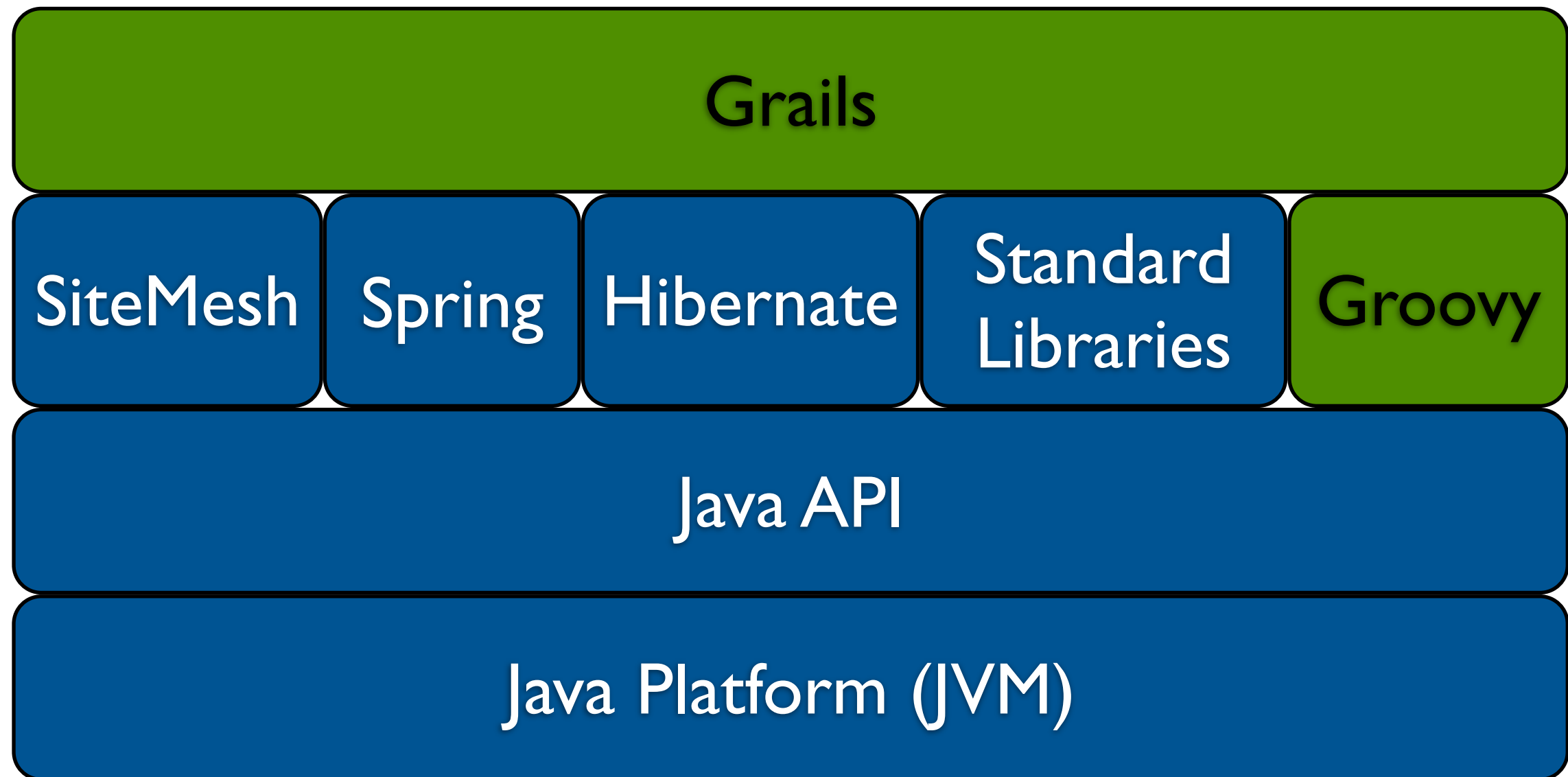
SiteMesh

Spring

Hibernate

Standard
Libraries

Grails Technology Stack



Key Grails Components

- Domain Class - A class representing an object in your domain (database table)
- Controller - A class that operates on URLs submitted to the web site
- View - A Groovy Server Page (GSP) designed to render the content based on a specific request

Domain Driven Design with Grails

- Focus on the complexities of the domain first
- Create domain objects and their relationships first
- Dynamically 'scaffold' controllers and views for each domain class, validate domain objects and relationships
- When ready, finish by generating or coding all views, controllers, and tests

Grails Scaffolding

- Dynamically generated views based on default templates
- No need to define a page before completing the data model

```
// Domain  
class Person {  
    String firstName  
    String lastName  
}
```

```
// Dynamic Scaffold Controller  
class PersonController {  
    def scaffold = true  
}
```

More than a web framework...

- Eases development of every tier
 - ▶ Integrated Groovy build system
 - ▶ Integrated Groovy unit and integration test mechanism
 - ▶ Simple ORM based on Hibernate
 - ▶ Extensible plug-in system built on Spring
 - ▶ Also simplifies Web MVC

Grails Object Relational Mapping (GORM)

GORM

- **G**rails **O**bject **R**elational **M**apping API
 - ▶ Uses domain classes written in Groovy
 - ▶ Injects methods into domain classes for load, save, update, and query data
 - ▶ Backed by Hibernate and Spring
 - ▶ Write Hibernate objects without all of the messy XML!

GORM Benefits

- Write your domain classes as POGOs
- Define validation via constraints
- Define relationships via simple belongsTo, hasMany mappings
- Easy to test framework, can use grails console or grails shell, even integration tests
- Finders make it easy to locate data

Grails Domain Classes

- Created with create-domain-class
- Simple Groovy Beans
 - ▶ Each member variable represents a column in a table
 - ▶ Implied version, primary key fields
 - ▶ Can define relationships with other domains, validation constraints

Connecting to PostgreSQL

- Grails defines database connections for various environments (Development, Test, Production)
- Steps to configure your database
 - ▶ Install your database driver
 - ▶ Modify `grails-app/config/DataSource.groovy`
 - ▶ Boot grails with `grails run-app`

Defining Constraints

- Checked automatically when saved or when `validate()` is called
- Tip: scaffold configuration follows constraint order for field order

```
class Customer {  
  
    static constraints = {  
        firstName(maxLength: 15, blank: false)  
        lastName(maxLength: 20, blank: false)  
        registrationDate(nullable: false)  
        preferredCustomer(default: true)  
        awardsBalance(range: 0.0..5000.0)  
    }  
    String firstName  
    String lastName  
    Date registrationDate  
    Boolean preferredCustomer  
    BigDecimal awardsBalance  
}
```

Entity Relationships

- Grails supports all Hibernate relation mappings
 - ▶ One to One
 - ▶ One to Many
 - ▶ Many to Many
 - ▶ Parent/Child (Inheritance)

One to many relationship

- Customer 'owns' accounts (cascade deletes, updates, saves)

```
class Customer {  
  
    String firstName  
    String lastName  
    ...  
  
    static hasMany = [accounts:Account]  
}
```

```
class Account {  
  
    static belongsTo =  
        [customer:Customer]  
  
    String accountNumber  
}
```

```
// arbitrary but working example... flush for testing only...  
def customer = new Customer(firstName: "Dude", lastName: "WheresMyCar")  
def account = new Account(accountNumber:"1234", customer:customer)  
customer.accounts = [account]  
def result = customer.save(flush:true)  
  
def account2 = Account.findById(account.id)  
println account2.customer  
  
def customer2 = Customer.findById(customer.id)  
println customer2.accounts
```

Physical DB Settings - mapping

- Changing the table, versioning on/off, column mappings, indexing, multi-column-index, etc...

```
class Person {  
    String firstName  
    String address  
    static mapping = {  
        table 'people'  
        version false  
        id column: 'person_id'  
        firstName column: 'First_Name', index: 'Name_Idx'  
        address column: 'Address', index: 'Name_Idx, Address_Index'  
    }  
}
```

Stored Procedures

- Several approaches
 - ▶ Spring's JdbcTemplate
 - ▶ Spring's StoredProcedure class
 - ▶ Spring's SqlFunction
 - ▶ Groovy SQL
- Demonstration

Where to find the code

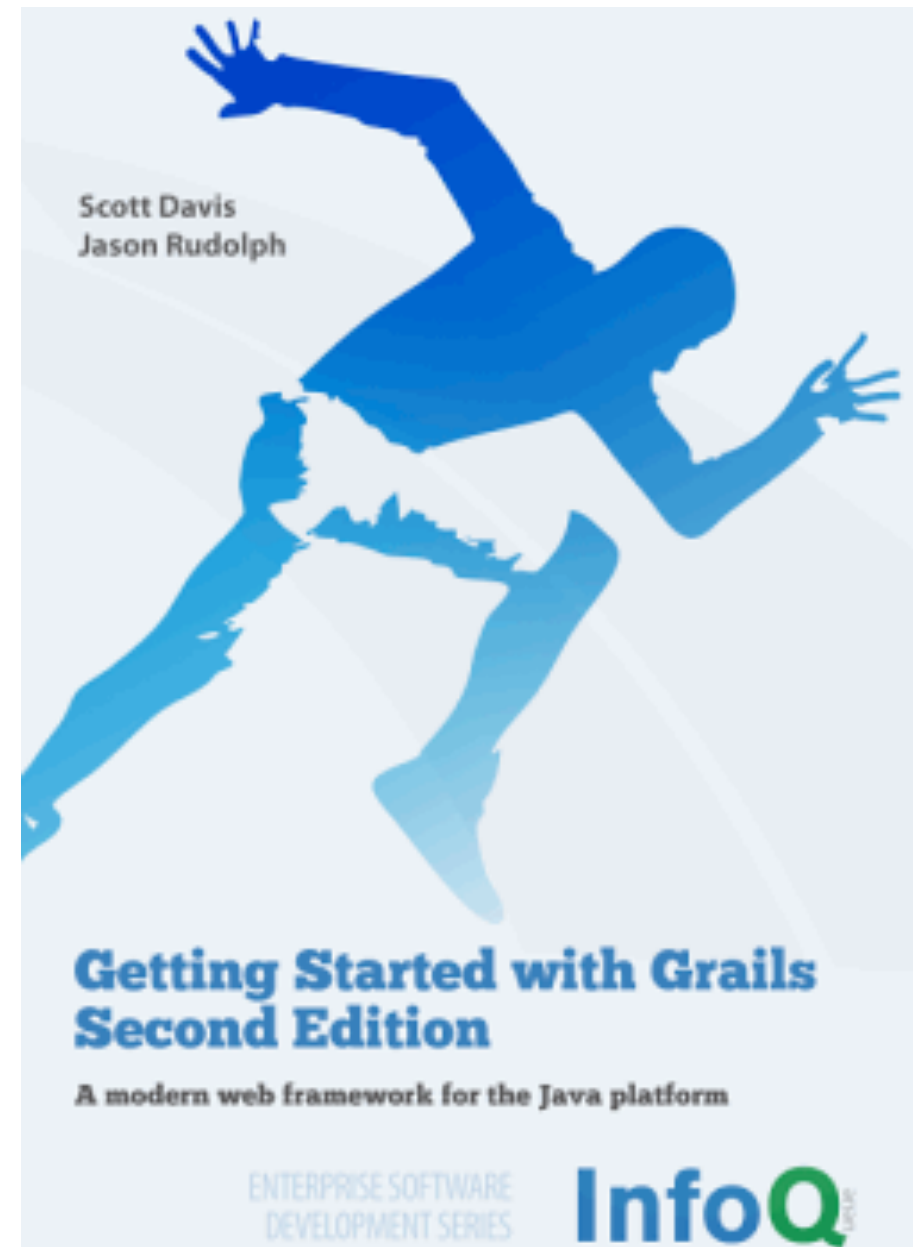
- Hit our GitHub repository (you can just ask to download the files)
- GitHub repo for our samples
 - ▶ [github.com/krimple/Grails-Demos-PostgreSQL-East-2010.git](https://github.com/krimple/Grails-Demos-PostgreSQL-East-2010)

Resources

- Grails Web Site
 - ▶ <http://www.grails.org>
- Grails User Email List
 - ▶ <http://n4.nabble.com/Grails-f1312388.html>
- Groovy Web Site
 - ▶ <http://groovy.codehaus.org/>

Free InfoQ E-Book

- Getting Started with Grails, Second Edition
 - ▶ Scott Davis
 - ▶ Jason Rudolph
- <http://www.infoq.com/minibooks/grails-getting-started>



Thank you!

Questions ... ?