Viva la evolución

Peter Backlund

CITERUS

Grails

- Java- och Groovybaserat
- Treskiktade webbapplikationer med databas
- Bygger på Spring, Hibernate och Sitemesh
- Groovy

CITERUS

Likheter med Rails

- Konventioner, inte konfiguration
- Profiler – dev, test, prod
- Scaffolding
- Levereras som en sammansatt stack

Skillnader

- **Domänmodellen drivs från Groovy (eller Java)**
- **Servicelager med transaktioner en naturlig del**
- **Statisk *och* dynamisk typning**
- **Trivialt att integrera med Java EE**

Var börjar vi?


```
$ grails create-app jfokus

jfokus
    grails-app
        domain
        service
        controllers
        views
    src
        java
        groovy
    test
        integration
        unit
```

Domänmodellen

- Allt under domain/ är persistent, mappas automatiskt
- GORM == Hibernate++
- Kan blandas med Javaklasser & manuell mappning

Minimalt exempel


```
class Person {
  String name
}
```

- Long id, version mixas in
- Groovy har inbyggda JavaBean-properties och implicita konstruktorer
- Hibernate defaults, som vanligt

Relationer


```
class Person {
  Address homeAddress                  // 1-1, inbäddad
  Person bestFriend                    // 1-1
  static hasMany = [pets:Animal]       // 1-*
  static embedded = ['homeAddress']
}
```

- **Särskilda statiska fält:**
  `hasMany, belongsTo, mappedBy, embedded, fetchMode, ...`
- **Set är default**

Validering


```
class Person {
    String name
    Person bestFriend
    static constraints = {
     name(range:4..100,match:"^[A-Z][a-z]*$")
     bestFriend(nullable:true)
   }
}
```

- Relationer är ej-null default
- Massor av inbyggda regler, går att skapa godtyckliga
- Automatiskt vid save(), finns validate()

CRUD


```
def me = Person.findByFirstNameAndLastName(
         “Peter”, “Backlund”)
me.addToPets(new Cat(name:”Snowball”,color:WHITE))
me.save()
```

- **Active Record**
- **Dynamiska finders:**
  `“from Person where firstName = ? and lastname = ?”, “Peter”, “Backlund”`
- **HQL, Criteria eller SQL går också bra**

Transaktioner

- Spring
- Servicelagret tillgängligt för Dependency Injection

```
class PersonService {
  static transactional = true
  void marry(Person one, Person theOther) {...}
}
```

- Även:

```
Person.withTransaction { ts ->
  ...
}
```

Controllers

- **Spring MVC**
- **Massor av mixins:** `request, session, params, redirect, chain, ...`
- **REST-likt: /customer/show/1**
- **Scaffolding – on-the-fly eller till fil**

Konventioner


```
CustomerController {
  def scaffold = true

  def overview = {
    def c = Customer.get(params.id)
    def recentOrders = Order.findRecentOrders(c)
    [customer:c, orders:recentOrders]
  }
}
```

- Vynamn blir “customer/overview”
- Returnernar ModelAndView eller Map
- Scaffold ger CRUD-hantering

Mer

- **Flash scope: lever till nästa request**
  `flash.message = “Stored user ${user.name}”`
- **Ny instans varje request, så medlemsvariabler är tillgängliga i vyn**
- **Binding: kommandoobjekt, `bindData()` eller**
  `aPerson.properties = params`

Render-metoden

- render “OK”
- render(view:”foo”,model:[bar:”baz”])
- render(contentType:”text/xml”) {
  customer {
  name c.name
  recentOrders {
  Order.findRecent(c).each { o ->
  order(date:o.date) {
  itemName o.item.name
  ...
  }
  }
  }
  }
  }
- render Person.list() as JSON

Resultatet


```
<?xml version="1.0"?>

<customer>
    <name>Peter Backlund</name>
    <recentOrders>
        <order date="2008-01-15">
            <itemName>Macbook Air</itemName>
            ...
        </order>
        ...
    </recentOrders>
</customer>
```

Vyer

- GSP – Groovy Server Pages
- Taglibs

```
<g:if test="${...}">...</g:if>
```

- Groovy-scriptlets/GString:

```
<p>
  Welcome back,
    ${customer.female ? "Miss":"Mr"}
${customer.lastName}
</p>
```

CITERUS

Testning

- test/unit: enhetstester
- Mocka med GroovyMock eller ExpandoMetaClass
- test/integration: context + mixins
- Canoo WebTest

CITERUS

Verktyg

- IntelliJ 7 + JetGroovy är bäst ;-)
- Debugger, korskompilering, refaktorering, körbara mål
- Eclipse, TextMate

Drift & deploy

- **Jetty, Hypersonic out-of-the-box under utveckling**

```
grails run-app
```

- **Packas till .war med kompilerad bytekod med JNDI-datakälla för drift**

```
grails war
```

- **“Äkta” Java EE – samexisterar med existerande infrastruktur**

Svagheter

- 1.0 RC4 – kan vara lite oslipat
- Ingen delning av kontexten vid integrationstester
- Enorma stacktrace-ar
- Prestanda

USP

- Hög produktivitet – låg LoC
- Språkkompetens: Java ~ Groovy
- Ramverkskompetens: Spring, Hibernate
- Evolution, inte revolution!

Tack!

Agile Alliance Corporate Member

CITERUS

Hjälper företag att lyckas med mjukvaruutveckling