A real-world JEE Application written in Scala

Christian Hapke
christian.hapke@oximity.com

Like and follow us @ www.facebook.com/oximity

Agenda

- About Me + About Oximity
- Motivation
- App Overview
- Database + App Server
- JPA, EJBs, JSF, CDI
- Build
- Let's Explore the Code ...
- Lessons Learned
- Discussion

About Me

- Christian Hapke, Dipl.-Ing. Technical Computer Science and (Vordiplom) Mathematics
- Worked for GMD FOKUS/Fraunhofer, vectos, iLove/Jamba, Questico/adviqo and others
- Development and management: building Java backend software, software roll-out, new technologies and processes
- Founder of open source project silvertunnel.org
- Now: CTO and Co-Founder of start-up Oximity

About Oximity

- Redefinition of the entire News Media industry
- Transform how news is sourced, organised and consumed using the power of the crowd
- Bottom-up instead of top-down
- Public platform launch in summer 2013
- Currently offering a Junior Scala Developer position to students and marketing positions
- More full-time Scala Developer jobs in late summer

Motivation

- Java EE 6 architecture allows development of powerful and mature online applications
- Java EE 6 is usable (unlike Java 2 EE before v 5!!!)
- Java libraries for almost everything are available
- Scala allows cleaner, more concise and more expressive code than Java
- Very good interoperability between Scala and Java
- Why not combining best of everything?

App Overview (1/2)

- Demo application: show **Motto of the Minute**
- Based on technology stack of our real online platform
- Technology stack of the app:
    - JSF + Primefaces
    - CDI
    - EJB (session beans and scheduled jobs)
    - JPA
- Sources: github.com/oximity/motto

App Overview (2/2)

- Runtime Enviroment:
  - Scala 2.10 with Java 7
  - App server (example): JBoss 7.1
  - Database (example): MySQL 5.5 database
- Tools (not discussed here):
  - Build tool: gradle
  - IDE: Eclipse or IntelliJ IDEA
  - Testing: JUnit + Mockito + Selenium
  - Continous Integration/Deployment: Jenkins + Chef

Database (MySQL)

- SQL:

```
CREATE TABLE motto (
    motto_id BIGINT NOT NULL
      PRIMARY KEY AUTO_INCREMENT,
    content VARCHAR(255),
    author VARCHAR(255)
    ...);
```

- Char set: `utf8mb4` instead of `utf8`
- Full SQL inclusive test data: src/main/sql/motto.sql

App Server (JBoss)

- Configuration in jboss/standalone/configuration/standalone.xml
- Configuration of:
  - JDBC connection inclusive encoding stuff
  - Datasource name used by JPA
  - URI encoding UTF-8
  - Ports and root path
- Example: src/main/jboss/jboss-as-7.1/standalone.xml

JPA Configuration

- Configuration in src/main/resources/META-INF/persistence.xml
- Configuration of
  - Datasource
    - As configured for app server
  - Persistence unit name
    - Referenced in Scala code
  - Optional JPA/SQL logging

JPA Class–Table Mapping

- Model class:

```
import java.lang.{Long => Jlong}
…
@Entity
@Table(name="motto")
class Motto {
  @Id @GeneratedValue(strategy=GenerationType.IDENTITY)
  @Column(name="motto_id") @BeanProperty
  var mottoId: JLong = _

  @Column @BeanProperty
  var content: String = _

  @Column @BeanProperty
  var author: String = _
  …
}
```

JPA Database Access

- In EJB class:

```
@Stateless
@LocalBean
class MottoDBService {
  @PersistenceContext(unitName = "dbMotto")
  var em: EntityManager = _
  …
  def getMottoById(mId: Long): Option[Motto] = {
    try {
      Option(em.find(classOf[Motto], mottoId))
    } catch {
      case ex: NoResultException => { None }
    }
  }...
```

Service Layer with EJBs

- Highest layer of frontend-independent business logic
- By default: @TransactionAttribute(REQUIRED)
- Example service:

```
@Stateless
@LocalBean
class MottoService {
  @EJB
  var mottoDb: MottoDBService = _
  …
  def getRandomMotto(): Motto = {
    val maxId = mottoDb.getMaxMottoId()
    val randomId = (Math.random()*(maxId+1)).toLong
    mottoDb.getMottoById(randomId) match {
      case Some(motto) => motto
      case _           => getDefaultMotto()
    }
  } ...
```

Scheduled Jobs with EJBs

- Example:

```
@Singleton
@LocalBean
class MottoChangerJobService {
  …
  @Schedule(persistent=false,
    second="0", minute="*",
    hour="*", dayOfMonth="*",
    month="*", year="*")
  def setMottoOfTheMinute() {...}
```

- Crontab-like timing pattern

JSF Overview

- JSF pages
  - src/main/webapp/
- JSF components
  - Standard components + Primefaces extension
- JSF composite components
  - src/main/webapp/resources/jsf-components
- JSF expressions to access objects
  - #{myBean.propertyOrMethod}

JSF Pages

- Example form (mottoEdit.xhtml):

```
<h:form...>
  ...
  <p:inputText value="#{mottoEditPage.content}"...
  <p:inputText value="#{mottoEditPage.author}"...
  <p:commandButton
       action="#{mottoEditPage.createNewMotto}"...
  ...
</h:form>
```

- Maps HTML fields to fields in JSF backing bean
- Maps button to method in JSF backing bean

JSF Backing Beans (1/2)

- Example (MottoEditPage.scala):

```
@Named
@RequestScoped
class MottoEditPage {
  @Inject /* CDI injection of other CDI bean or EJB */
  var mottoService: MottoService = _
  ...
  @TextSingleLine @Size(...) @BeanProperty
  var content: String = _
  @TextSingleLine @Size(...) @BeanProperty
  var author: String = _
  …
  def createNewMotto(): String = { … }
```

JSF Backing Beans (2/2)

- `mottoEditPage` of type `MottoEditPage` is automatically available in JSF expression
- Same with fields if Java getters/setters are defined - in Scala generated with `@BeanProperty`
- Field with validators (`@TextSingleLine @Size`)
- Naming conventions simplify live: `mottoEdit.xhtml` – `MottoEditPage.scala`

JSF Composite Components (1/2)

- Usage with parameters:

```
<jsfcomp:mottoShowBox
   motto="#{mottoShowPage.motto}"
   title="This is the title"/>
```

- Parameters can be complex objects
- Objects need getters/setters to access data `(@BeanProperty)`

JSF Composite Components (2/2)

- Definition in mottoShowBox.xhtml:

```
<composite:interface>
  <composite:attribute
   name="motto" type="d.m.m.c.Motto".../>
  <composite:attribute
   name="title" type="String".../>
</composite:interface>

<composite:implementation>
  ...#{cc.attrs.title}...
  ...#{cc.attrs.motto.content}...
</composite:implementation>
```

CDI Beans (1/2)

- Example (MottoEditPage.scala):

```
@Named
@RequestScoped
class MottoEditPage {
  @Inject /* other CDI Bean */
  var msg: Messages = _
  @Inject /* EJB */
  var mottoService: MottoService = _
  @Inject /* dynamically produced bean */
  var log: Logger = _
```

- Injections are by default based on field type

CDI Beans (2/2)

- Possible injections:
  - CDI beans
  - EJBs
  - Dynamically produced beans (e.g. Logger)
- Different scopes (lifetimes) of CDI beans:
  - `@RequestScoped`
  - `@ConversationScoped`
  - `@SessionScoped`
  - `@ApplicationScoped`

Build

- Get the code
    - `git clone git@github.com:oximity/motto.git`
- Build the code
    - `gradle clean war`
- Configure database and app server
- Deploy war and start app server

Let's Explore the Code ...

Lessons Learned (1/3)

- Scala and Java EE APIs interact without problems
  - All Java EE annotations work with Scala
- Scala code much better to read than Java
- Functional programming used only when appropriate
- In contrast: Scala trainings often suggest that most problems should be solved in a functional way

Lessons Learned (2/3)

- Most problems with EE, not with Scala
  - JSF notably hard to debug
- JPA classes and JSF backing beans
  - Need getters/setters: with `@BeanProperty`
  - These objects are mutable
  - Types must by Java-compatible – watch:
    - primitve types vs. objects, e.g. `scala.Long` vs. `java.lang.Long`)
    - Collections

Lessons Learned (3/3)

- Scala compiler is quite slow compared to Java
    - Incremental builds are essential for developers
- Limited tool support for Scala, e.g. in IDEs
    - Eclipse with limitations
    - IntelliJ IDEA better

Discussion

- Questions
- Answers
- Comments

We love Scala!

github.com/oximity/motto

www.oximity.com/jobs/
christian.hapke@oximity.com

Like and follow us @ www.facebook.com/oximity