From the Spring Framework to Java EE 7
Ivar Grimstad
About the Speaker

Ivar Grimstad
Software Architect at Cybercom Sweden
“[…] is so much better than […]!”
What this presentation is NOT about

“[…] rocks and […] sucks!”
What this presentation IS about

• Feature comparison
• Migration from Spring Framework to Java EE
Content

• Background
• Comparison of Spring Framework and Java EE
• Convert Demo application to Java EE
  – Highlight differences and similarities
• Lessons learned
History

1998
• JPE

1999
• J2EE 1.2

2001
• J2EE 1.3

2002
• Spring

2003
• J2EE 1.4

2004
• Spring 1

2006
• Java EE 5
• Spring 2

2009
• Java EE 6
• Spring 3

2013
• Java EE 7
• Spring 4
Spring Framework

- Flexible
- Lightweight
- Modular
- Extensible
Java EE 7
## Comparison of Selected Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Spring 3.2.x</th>
<th>Java EE 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependency Injection</td>
<td>Spring IoC</td>
<td>CDI</td>
</tr>
<tr>
<td>Web Framework</td>
<td>Spring MVC</td>
<td>JSF</td>
</tr>
<tr>
<td>REST</td>
<td>Spring MVC</td>
<td>JAX-RS</td>
</tr>
<tr>
<td>Transactions</td>
<td>Annotations, AOP</td>
<td>EJB, JTA</td>
</tr>
<tr>
<td>Persistence</td>
<td>JDBC Templates, Spring Data</td>
<td>JPA</td>
</tr>
<tr>
<td>Batch</td>
<td>Spring Batch</td>
<td>Batch Applications for Java Platform 1.0</td>
</tr>
<tr>
<td>WebSockets</td>
<td>-</td>
<td>Java API for WebSockets 1.0</td>
</tr>
<tr>
<td>Validation</td>
<td>Spring Validation API</td>
<td>Bean Validation API</td>
</tr>
<tr>
<td>Security</td>
<td>Spring Security</td>
<td>Java EE Security</td>
</tr>
<tr>
<td>Messaging</td>
<td>JMS</td>
<td>JMS</td>
</tr>
</tbody>
</table>
## Comparison of Selected Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Spring 3.2.x</th>
<th>Java EE 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependency Injection</td>
<td>Spring IoC</td>
<td>CDI</td>
</tr>
<tr>
<td>Web Framework</td>
<td>Spring MVC</td>
<td>JSF</td>
</tr>
<tr>
<td>REST</td>
<td>Spring MVC</td>
<td>JAX-RS</td>
</tr>
<tr>
<td>Transactions</td>
<td>Annotations, AOP</td>
<td>EJB, JTA</td>
</tr>
<tr>
<td>Persistence</td>
<td>JDBC Templates, Spring Data</td>
<td>JPA</td>
</tr>
<tr>
<td>Batch</td>
<td>Spring Batch</td>
<td>Batch Applications for Java Platform 1.0</td>
</tr>
<tr>
<td>WebSockets</td>
<td>-</td>
<td>Java API for WebSockets 1.0</td>
</tr>
<tr>
<td>Validation</td>
<td>Spring Validation API</td>
<td>Bean Validation API</td>
</tr>
<tr>
<td>Security</td>
<td>Spring Security</td>
<td>Java EE Security</td>
</tr>
<tr>
<td>Messaging</td>
<td>JMS</td>
<td>JMS</td>
</tr>
</tbody>
</table>
Demo
Demo Application

• Online Cook Book
  – Web UI
  – REST API
Demo Application Data Model

CookBookUser -> CookBook -> Recipe
Demo Application Architecture
Spring Implementation
Demo Application Characteristics

- WEB Application
- Based on Spring Framework
- Mix of XML and Annotations
Migration Steps

1. Add Java EE dependency
2. Layer for layer
   1. Add layer specific Java EE configuration
      (web.xml, persistense.xml, faces-config.xml)
   2. Add Java EE annotations
   3. Replace Spring specific implementation with Java EE
   4. Replace Spring injections with CDI
3. Remove Spring configuration
4. Remove Spring Dependencies
Layer for Layer Walkthrough
Preparations
## Dependency Injection

<table>
<thead>
<tr>
<th>Spring IoC</th>
<th>CDI</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Autowired XML</td>
<td>@Inject</td>
</tr>
</tbody>
</table>

@ivar_grimstad
Persistence Layer

- REST API
- Web UI
- Services
- Persistence
- DB
## Persistence Layer

<table>
<thead>
<tr>
<th>Spring JDBC</th>
<th>JPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Repository JdbcTemplate</td>
<td>@PersistenceContext</td>
</tr>
<tr>
<td></td>
<td>@Entity</td>
</tr>
<tr>
<td></td>
<td>@OneToMany, @ManyToOne, @ManyToMany</td>
</tr>
<tr>
<td></td>
<td>persistence.xml</td>
</tr>
</tbody>
</table>
protected JdbcTemplate jdbcTemplate;

@Autowired
defined void setDataSource(DataSource dataSource) {
    this.jdbcTemplate = new JdbcTemplate(dataSource);
}

@Override
defined List<CookBook> findAll() {
    List<CookBook> cookBooks = jdbcTemplate.query("select * from COOKBOOK", new CookBookMapper());
    for (CookBook cookBook : cookBooks) {
        cookBook.setOwner(cookBookUserDao.find(cookBook.getOwnerId()));
    }
    return cookBooks;
}
@PersistenceContext(unitName = "cookBookPU")
private EntityManager em;

public List<T> findAll(Class<T> entityClass) {
    CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
    cq.select(cq.from(entityClass));
    return em.createQuery(cq).getResultList();
}
Service Layer

- REST API
- Web UI
- Services
- Persistence
- DB
### Service Layer

<table>
<thead>
<tr>
<th>Spring POJO</th>
<th>EJB</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Service</td>
<td>@Local</td>
</tr>
<tr>
<td>@.Transactional</td>
<td>@Stateless</td>
</tr>
</tbody>
</table>

JDD 2013
public class CookBookServiceBean implements CookBookService {

    @Autowired
    private CookBookDao cookBookDao;

```
EJB

```java
/** ... */
@Local
public interface CookBookService {

/** ... */
@Stateless
public class CookBookServiceBean implements CookBookService {
    @Inject
    private CookBookDao cookBookDao;
```
REST API
## REST API

<table>
<thead>
<tr>
<th>Spring MVC</th>
<th>JAX-RS</th>
</tr>
</thead>
<tbody>
<tr>
<td>@Controller</td>
<td>@Path</td>
</tr>
<tr>
<td>@RequestMapping</td>
<td>@GET, @POST,</td>
</tr>
<tr>
<td>@ResponseBody</td>
<td>@PUT, @DELETE</td>
</tr>
<tr>
<td>@RequestBody</td>
<td>@Consumes,</td>
</tr>
<tr>
<td>@PathVariable</td>
<td>@Produces,</td>
</tr>
<tr>
<td>@RequestParam</td>
<td>@Produces</td>
</tr>
<tr>
<td>@RequestParam</td>
<td>@PathParam</td>
</tr>
<tr>
<td></td>
<td>@QueryParam</td>
</tr>
</tbody>
</table>
**Spring MVC**

```java
/**...*/
@Controller
@RequestMapping("cookbooks")
public class CookBookResource {

    @Autowired
    private CookBookService cookBookService;

    @RequestMapping(value = "/{id}", method = RequestMethod.GET, produces = APPLICATION_JSON_VALUE)
    @ResponseBody
    public CookBook find(@PathVariable("id") Long id) {
        return cookBookService.find(id);
    }
}
```
```java
/** ...
 */
@Path("cookbooks")
public class CookBookResource {

    @EJB
    private CookBookService cookBookService;

    @GET
    @Path("{id}")
    @Produces(APPLICATION_JSON)
    public CookBook find(@PathParam("id") Long id) {
        return cookBookService.find(id);
    }
```
Web UI

Diagram showing the relationship between Web UI, REST API, Services, Persistence, and DB.
## Web UI

<table>
<thead>
<tr>
<th>Spring MVC</th>
<th>Java ServerFaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSP</td>
<td>XHTML</td>
</tr>
<tr>
<td>@Controller</td>
<td>@Named</td>
</tr>
<tr>
<td>@RequestMapping</td>
<td></td>
</tr>
</tbody>
</table>
```java
/**...
 * @Controller
 * @RequestMapping("/cookBook")
 * public class CookBookController {

     @Autowired
     private CookBookService cookBookService;
```
public class CookBookController implements Serializable {

    private CookBookService cookBookService;

    @EJB

Spring resources (i18n)

**applicationContext.xml**

```xml
<br:bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">
  <br:property name="basename">
    <value>translations</value>
  </br:property>
</br:bean>
```

**Create.jsp**

```html
	<tbody>
		<tr>
			<td><br:label for="name"><br:message code="CreateCookBookLabel_name"/></br:message></br:label></td>
			<td><br:input id="name" type="text" name="name" title="<br:message code="CreateCookBookTitle_name"/>"></br:input></td>
		</tr>
	</tbody>
```
JSF resources (i18n)

```xml
faces-config.xml

  <application>
    <resource-bundle>
      <base-name>/translations</base-name>
      <var>bundle</var>
    </resource-bundle>
  </application>

Create.xhtml

  <h:outputLabel value="#{bundle.CreateCookBookLabel_name}" for="name" />
  <h:inputText id="name" value="#{cookBookController.selected.name}" title="#{bundle.CreateCookBookTitle_name}" />
```
Remove Spring
Application Architecture - Spring
Application Architecture – Java EE
Migration Steps – what we did

1. Added Java EE dependency
2. Layer for layer
   1. Added layer specific Java EE configuration
      (web.xml, persistense.xml, faces-config.xml)
   2. Added Java EE annotations
3. Replaced Spring specific implementation with Java EE
4. Replaced Spring injections with CDI
3. Removed Spring configuration
4. Removed Spring Dependencies
Dependencies (Spring)
WAR File (Spring)
WAR File (Java EE)
What about testing?

• Spring Framework has excellent support for testing
• No direct support for testing in Java EE 7
• Tools like Arquillian is an option
Not Covered Here

• Simplified JMS
• Java API for WebSocket 1.0
• JSON Processing 1.0
• Batch Applications for Java Platform 1.0
Why Migrate?

- Spring IS a proprietary framework
- Java EE is a standard
- Many Spring applications run in a Java EE container anyway
When NOT to Migrate?

• If your target platform is prior to Java EE 6
• If your developers are experts on Spring Framework
• If your application is heavily based on AOP
Lessons Learned

• Migration is not that hard
• How hard it is depends on application structure
• Migration from newer Spring version is easier
  – may even be required to upgrade first
• Not all applications can be migrated
Wrap Up