CYBERCOM GROUP

# From the Spring Framework to Java EE 7

Ivar Grimstad

## About the Speaker

Ivar Grimstad

Software Architect at Cybercom Sweden

## What this presentation is NOT about

*“ […] 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

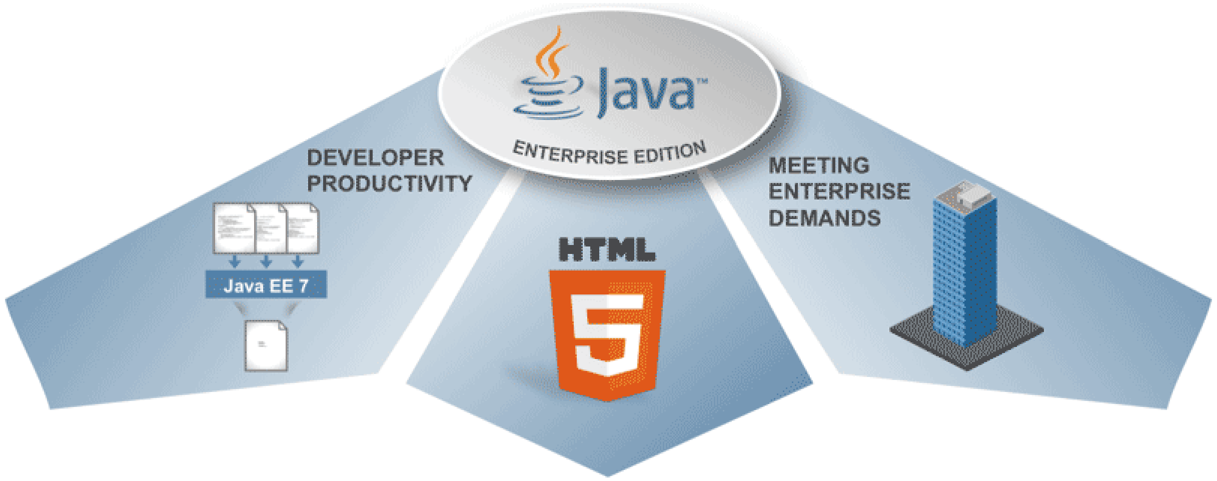
# History

- 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


Java
ENTERPRISE EDITION
DEVELOPER PRODUCTIVITY
Java EE 7
HTML
MEETING ENTERPRISE DEMANDS

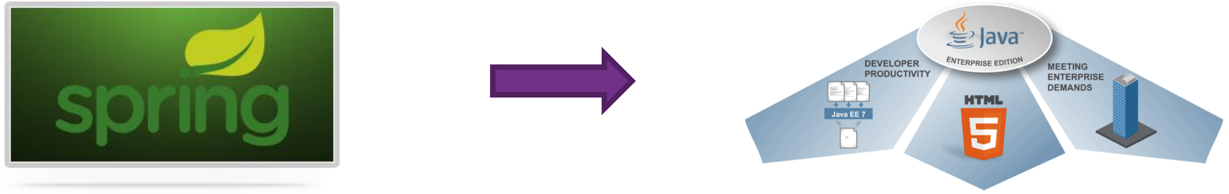
## Comparison of Selected Features

| Feature | Spring 3.2.x | Java EE 7 |
|---|---|---|
| Dependency Injection | Spring IoC | CDI |
| Web Framework | Spring MVC | JSF |
| REST | Spring MVC | JAX-RS |
| Transactions | Annotations, AOP | EJB, JTA |
| Persistence | JDBC Templates, Spring Data | JPA |
| Batch | Spring Batch | Batch Applications for Java Platform 1.0 |
| WebSockets | - | Java API for WebSockets 1.0 |
| Validation | Spring Validation API | Bean Validation API |
| Security | Spring Security | Java EE Security |
| Messaging | JMS | JMS |

## Comparison of Selected Features

| Feature | Spring 3.2.x | Java EE 7 |
|---|---|---|
| Dependency Injection | Spring IoC | CDI |
| Web Framework | Spring MVC | JSF |
| REST | Spring MVC | JAX-RS |
| Transactions | Annotations, AOP | EJB, JTA |
| Persistence | JDBC Templates, Spring Data | JPA |
| Batch | Spring Batch | Batch Applications for Java Platform 1.0 |
| WebSockets | - | Java API for WebSockets 1.0 |
| Validation | Spring Validation API | Bean Validation API |
| Security | Spring Security | Java EE Security |
| Messaging | JMS | JMS |

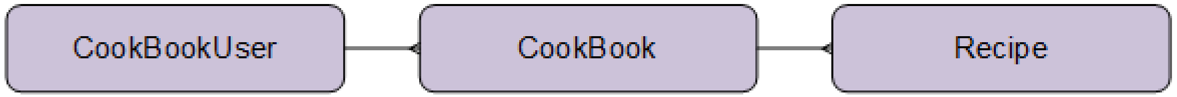
## Demo


spring
Java
ENTERPRISE EDITION
DEVELOPER PRODUCTIVITY
Java EE 7
HTML 5
MEETING ENTERPRISE DEMANDS

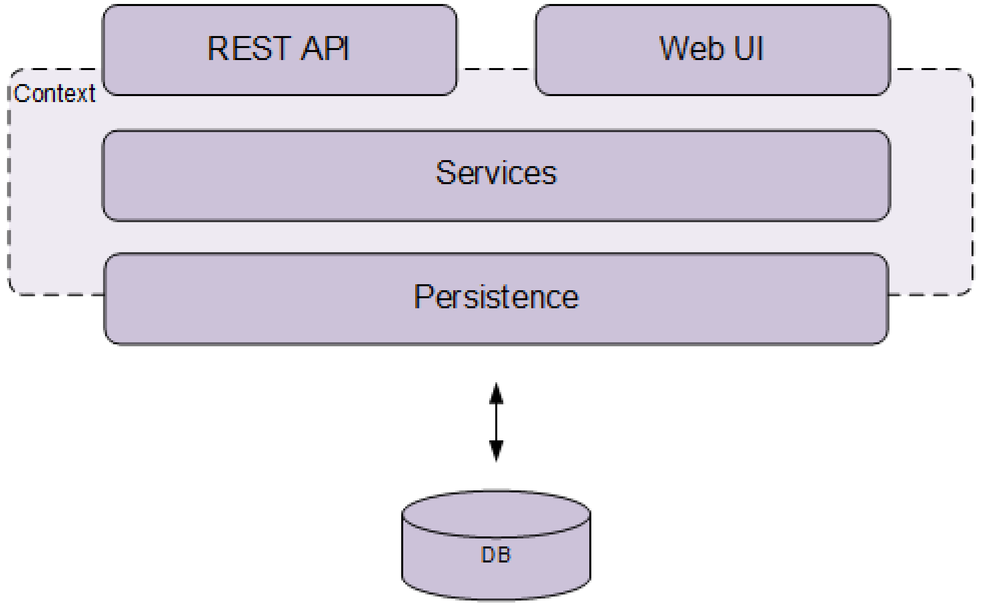
## Demo Application

- Online Cook Book
  - Web UI
  - REST API

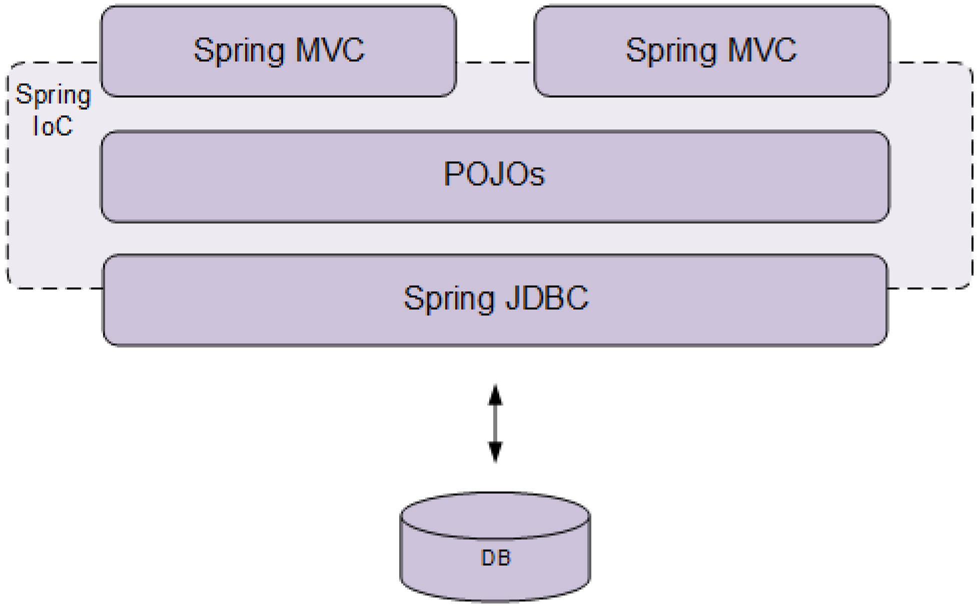
## Demo Application Data Model

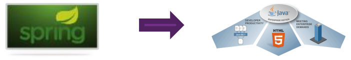


## Demo Application Architecture



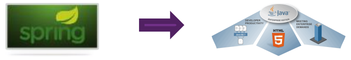
# Spring Implementation





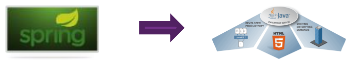
## Demo Application Characteristics

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

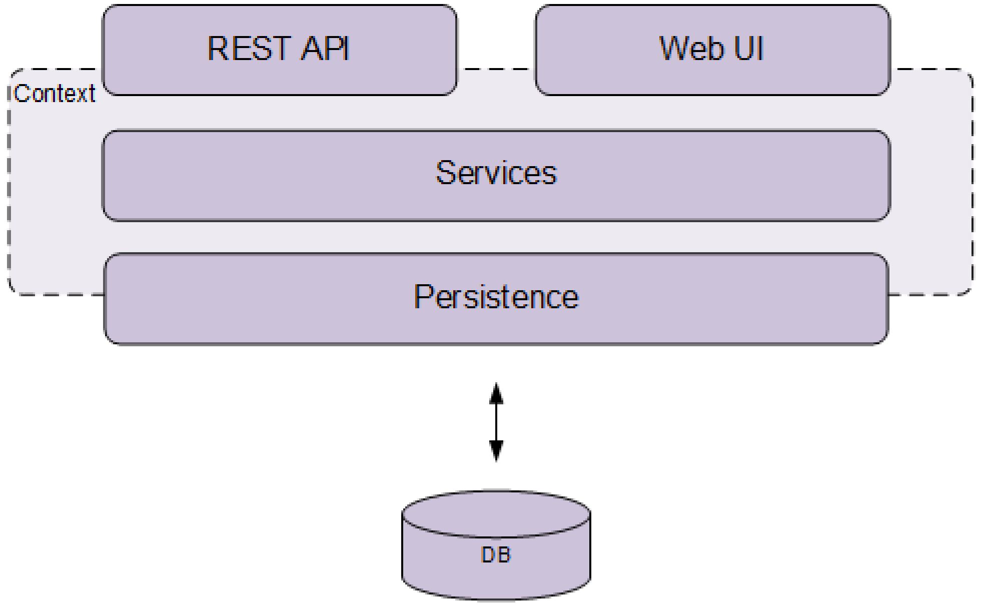


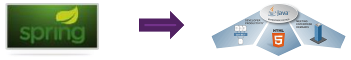
## Migration Steps

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

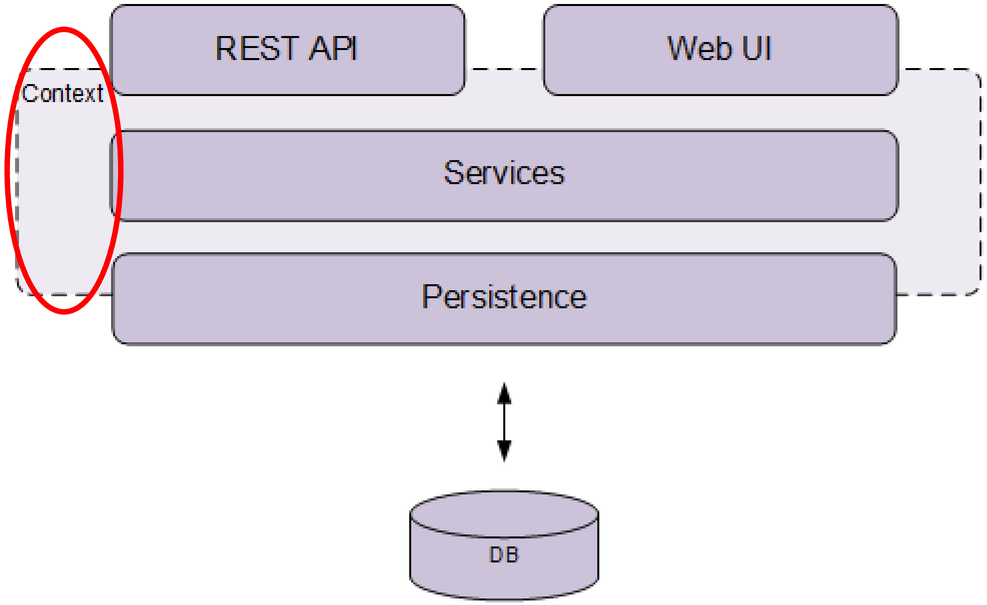


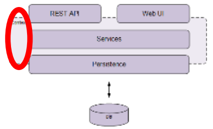
## Layer for Layer Walkthrough

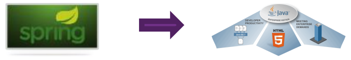




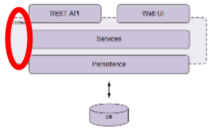
## Preparations



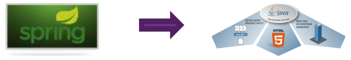




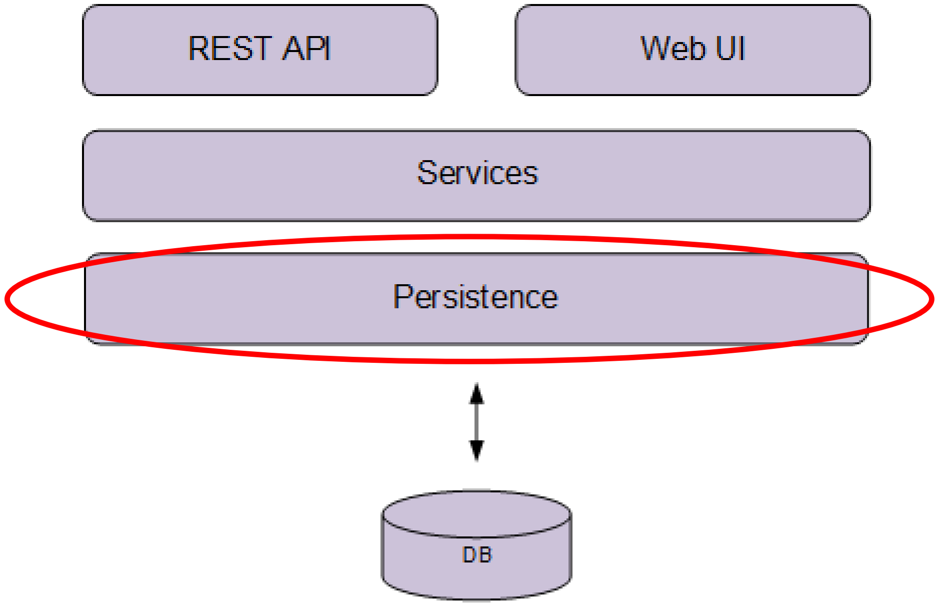
## Dependency Injection

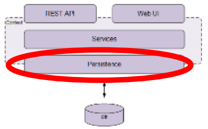


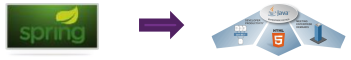
| Spring IoC | CDI |
|---|---|
| @Autowired<br>XML | @Inject |



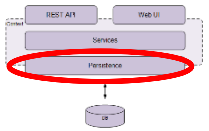
# Persistence Layer



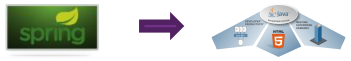




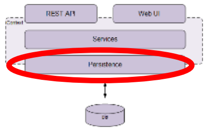
## Persistence Layer

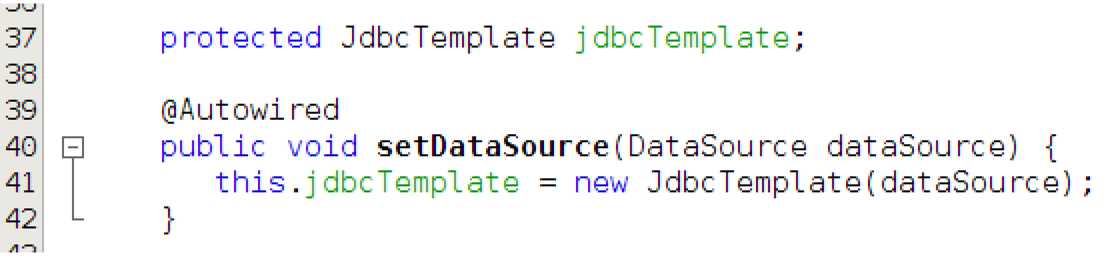


| Spring JDBC | JPA |
|---|---|
| @Repository<br>JdbcTemplate | @PersistenceContext<br>@Entity<br>@OneToMany,<br>@ManyToOne,<br>@ManyToMany |
|  | persistence.xml |

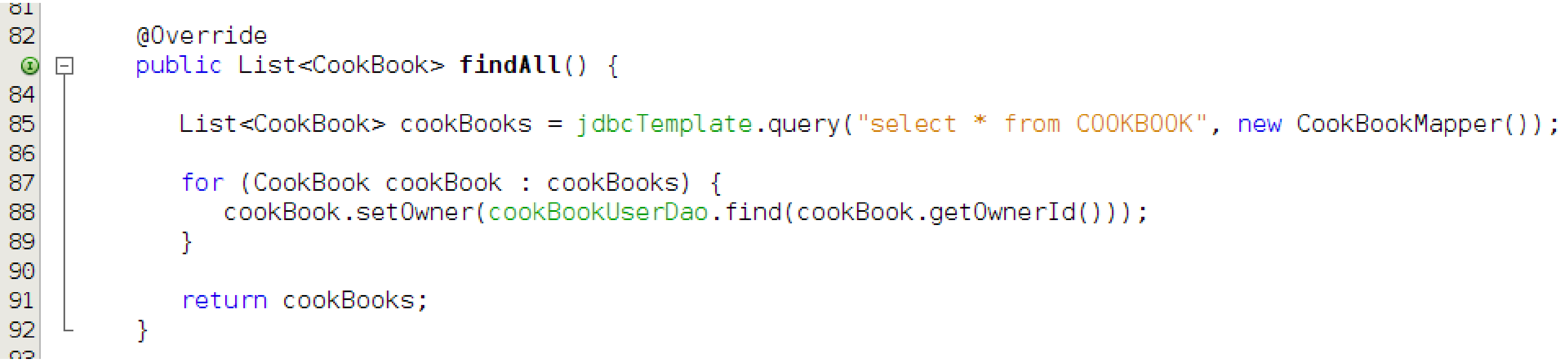


# Spring JDBC



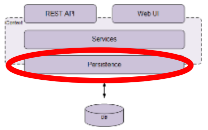


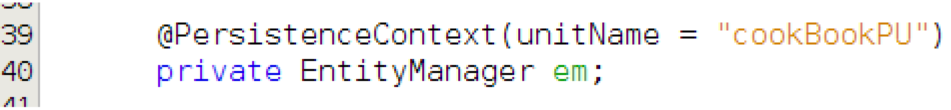
```java
    protected JdbcTemplate jdbcTemplate;

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



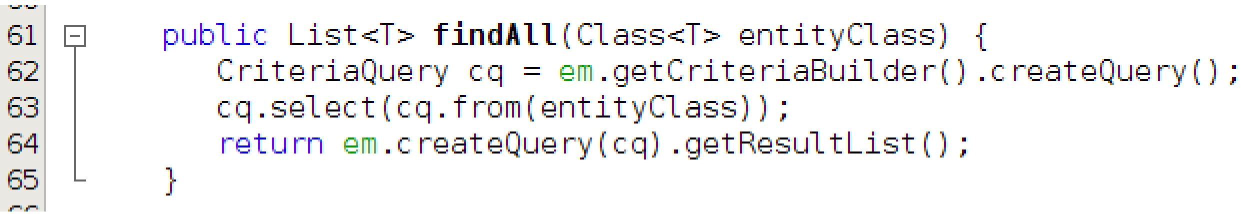
```java
    @Override
    public List<CookBook> findAll() {

        List<CookBook> cookBooks = jdbcTemplate.query("select * from COOKBOOK", new CookBookMapper());

        for (CookBook cookBook : cookBooks) {
            cookBook.setOwner(cookBookUserDao.find(cookBook.getOwnerId()));
        }

        return cookBooks;
    }
```

# JPA



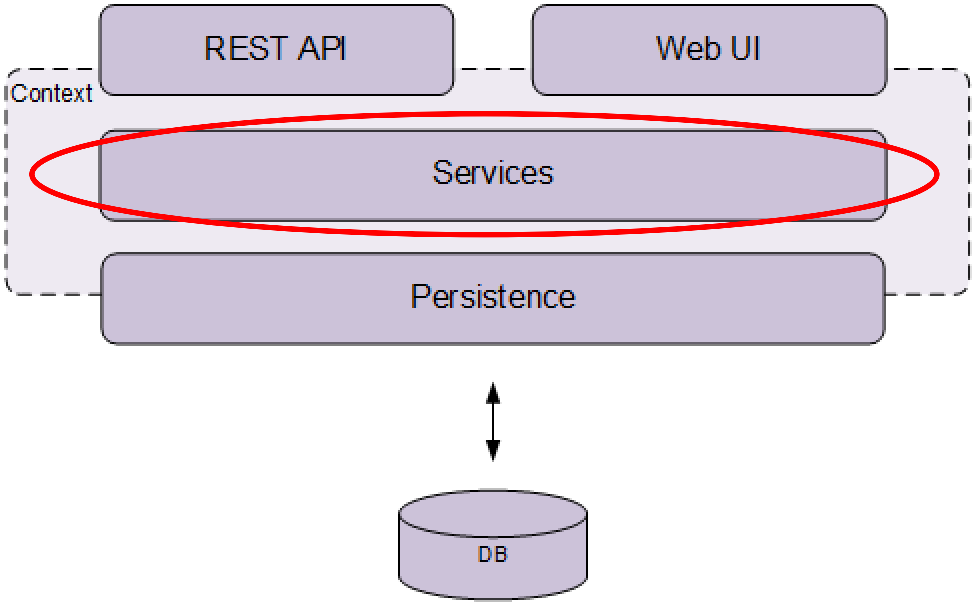


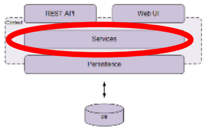
```java
@PersistenceContext(unitName = "cookBookPU")
private EntityManager em;
```

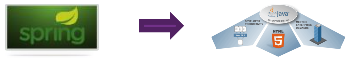


```java
public List<T> findAll(Class<T> entityClass) {
    CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
    cq.select(cq.from(entityClass));
    return em.createQuery(cq).getResultList();
}
```

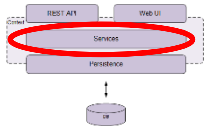
# Service Layer



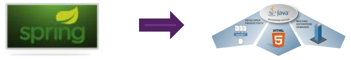




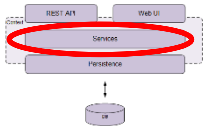
## Service Layer

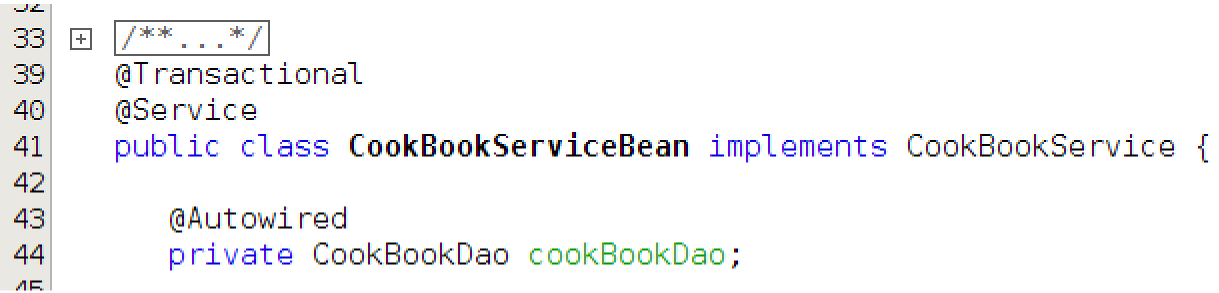


| Spring POJO | EJB |
|---|---|
| @Service<br>@Transactional | @Local<br>@Stateless |

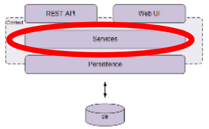


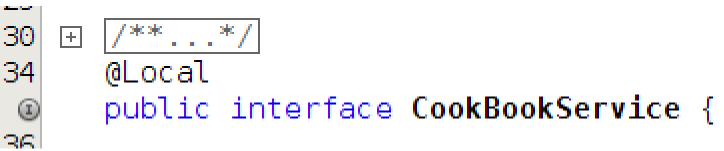
## Spring



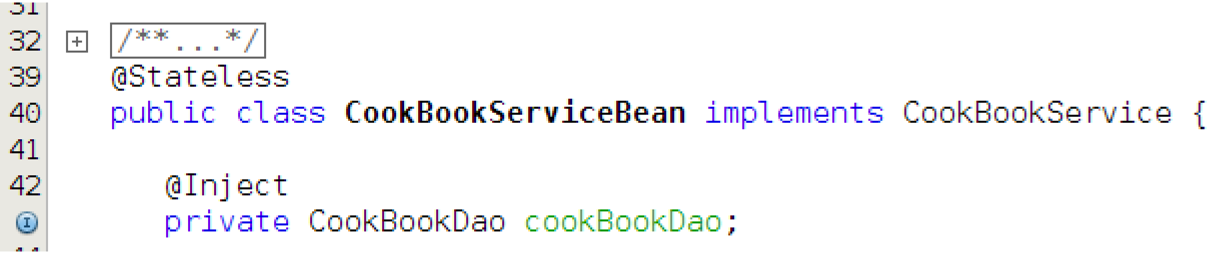

```
/**...*/
@Transactional
@Service
public class CookBookServiceBean implements CookBookService {

    @Autowired
    private CookBookDao cookBookDao;
```

# EJB



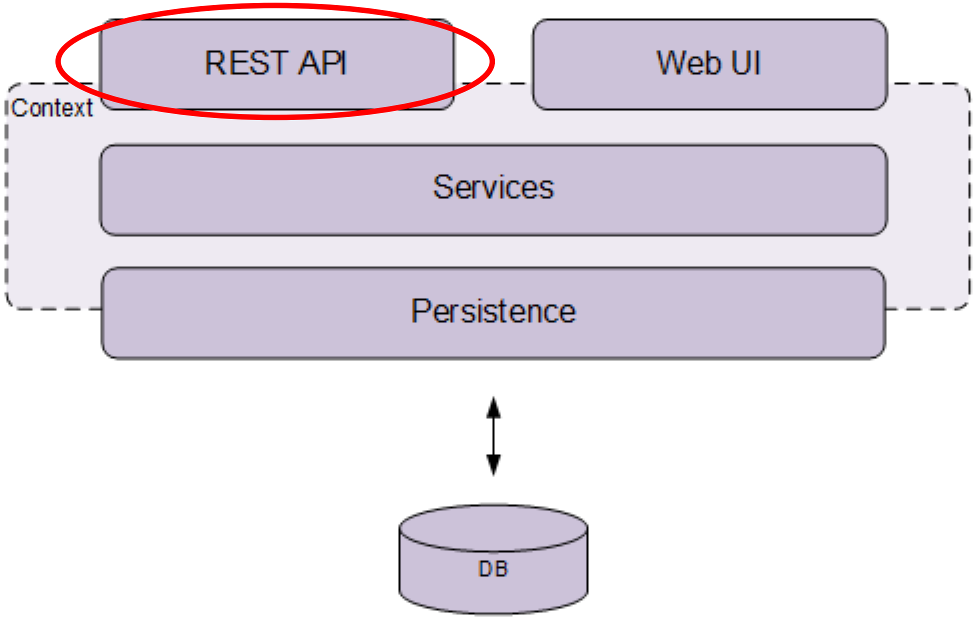


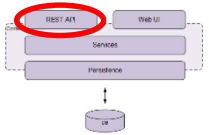
```java
/**...*/
@Local
public interface CookBookService {
```

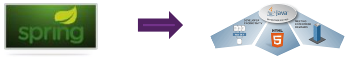


```java
/**...*/
@Stateless
public class CookBookServiceBean implements CookBookService {

    @Inject
    private CookBookDao cookBookDao;
```

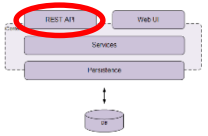
# REST API



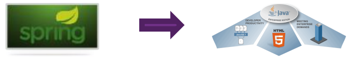




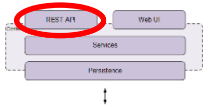
# REST API

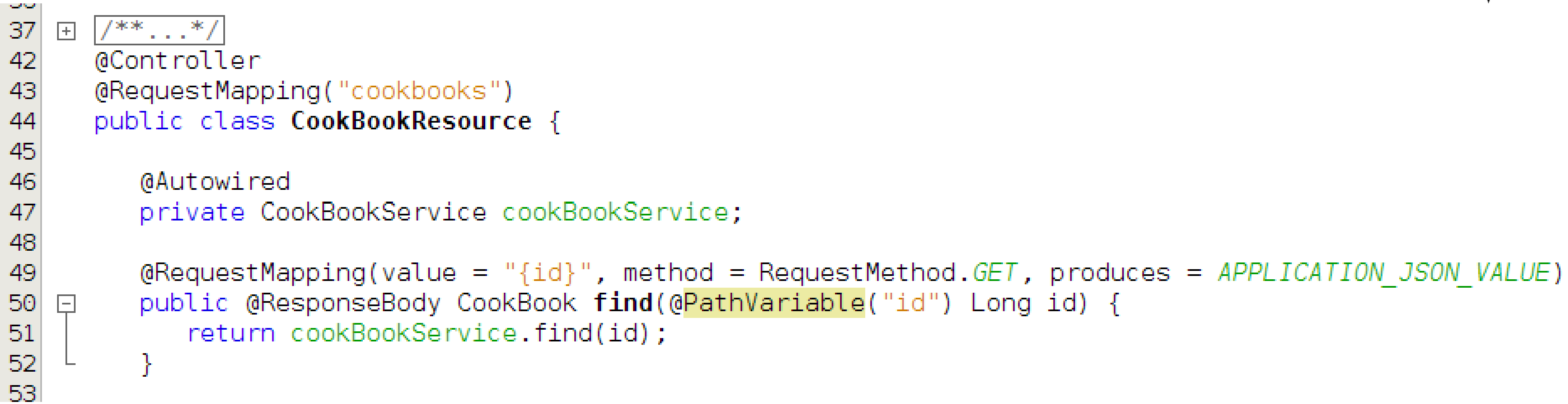


| Spring MVC | JAX-RS |
|---|---|
| @Controller<br>@RequestMapping<br>@ResponseBody<br>@RequestBody<br>@PathVariable<br>@RequestParam | @Path<br>@GET, @POST,<br>@PUT, @DELETE<br>@Consumes,<br>@Produces<br>@PathParam<br>@QueryParam |



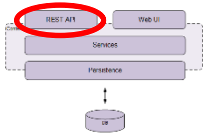
# Spring MVC

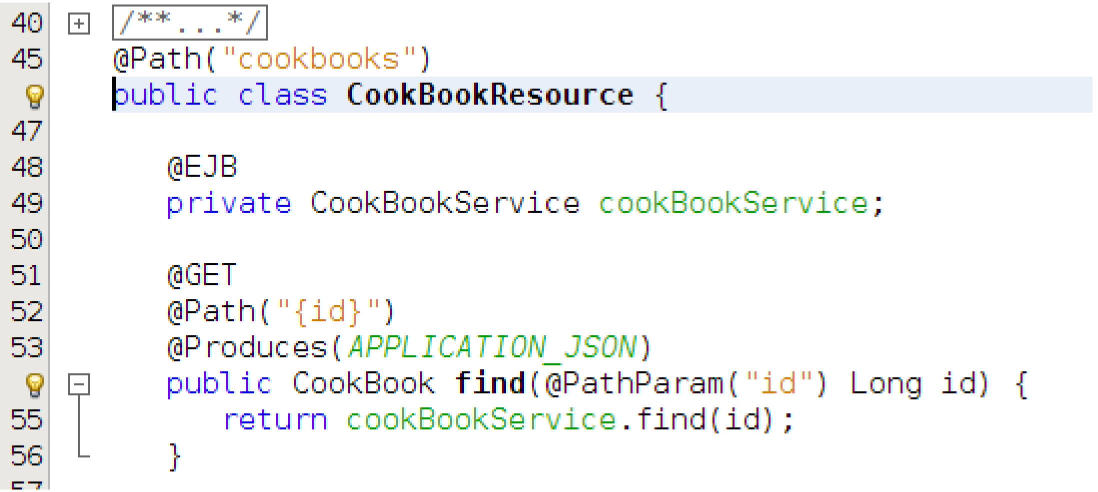




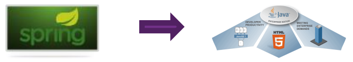
```java
/**...*/
@Controller
@RequestMapping("cookbooks")
public class CookBookResource {

    @Autowired
    private CookBookService cookBookService;

    @RequestMapping(value = "{id}", method = RequestMethod.GET, produces = APPLICATION_JSON_VALUE)
    public @ResponseBody CookBook find(@PathVariable("id") Long id) {
        return cookBookService.find(id);
    }
```

# JAX-RS

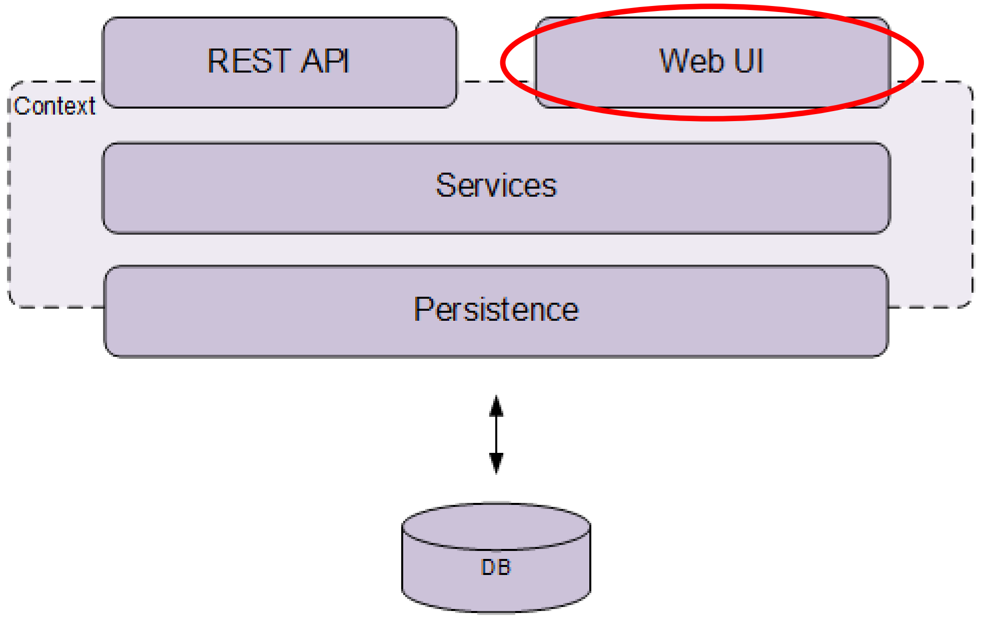


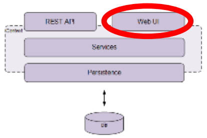


```
/**...*/
@Path("cookbooks")
public class CookBookResource {

    @EJB
    private CookBookService cookBookService;

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

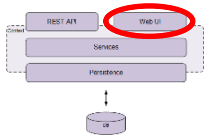


# Web UI


REST API
Web UI
Context
Services
Persistence
DB

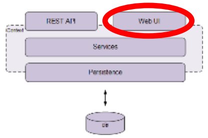

REST API
Web UI
Services
Persistence

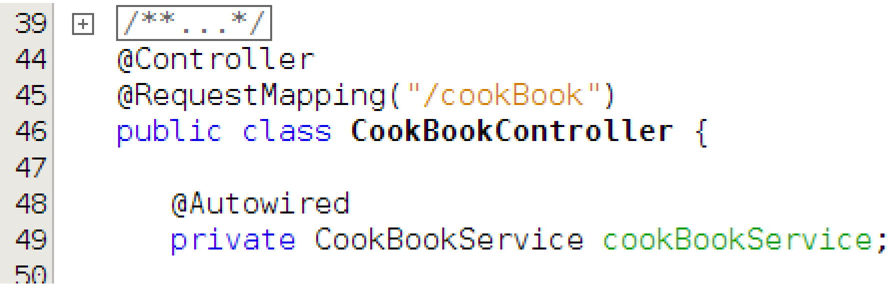
# Web UI



| Spring MVC | Java ServerFaces |
|---|---|
| JSP | XHTML |
| @Controller<br>@RequestMapping | @Named |

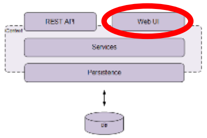
# Spring MVC Controller

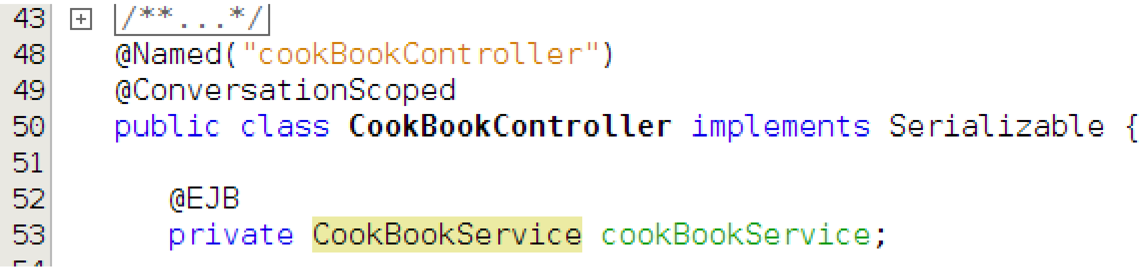




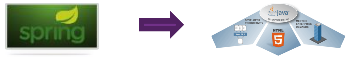
```
/**...*/
@Controller
@RequestMapping("/cookBook")
public class CookBookController {

    @Autowired
    private CookBookService cookBookService;

```

# JSF Controller

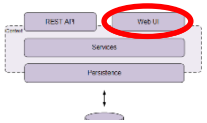




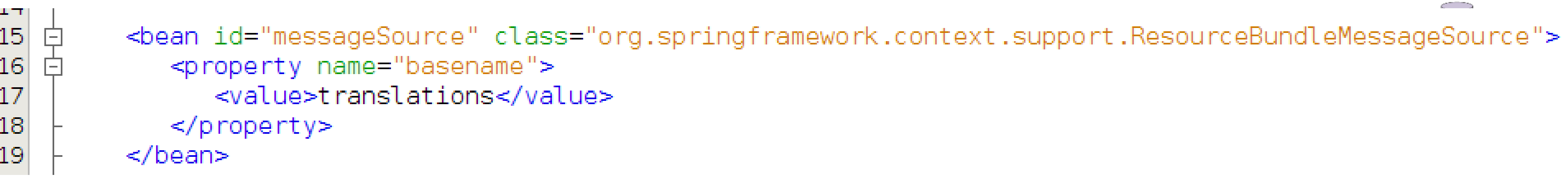
```
/**...*/
@Named("cookBookController")
@ConversationScoped
public class CookBookController implements Serializable {

    @EJB
    private CookBookService cookBookService;
```



# Spring resources (i18n)

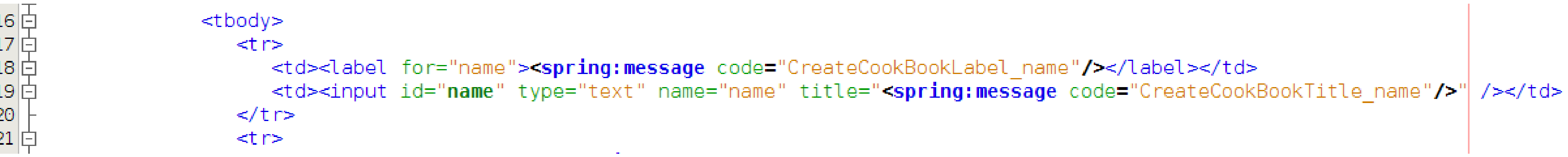


applicatonContext.xml



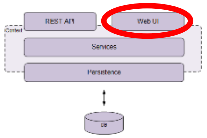
```
    <bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">
        <property name="basename">
            <value>translations</value>
        </property>
    </bean>
```

Create.jsp

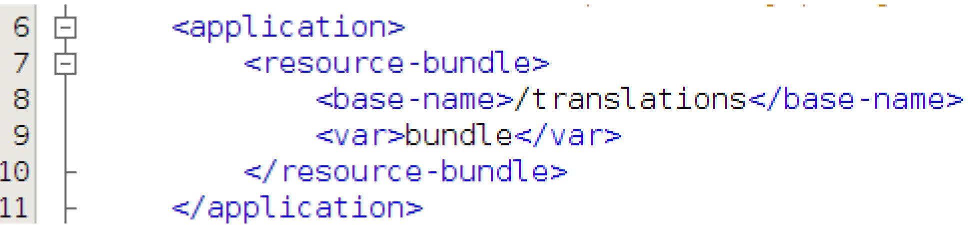


```
            <tbody>
                <tr>
                    <td><label for="name"><spring:message code="CreateCookBookLabel_name"/></label></td>
                    <td><input id="name" type="text" name="name" title="<spring:message code="CreateCookBookTitle_name"/>" /></td>
                </tr>
                <tr>
```

# JSF resources (i18n)

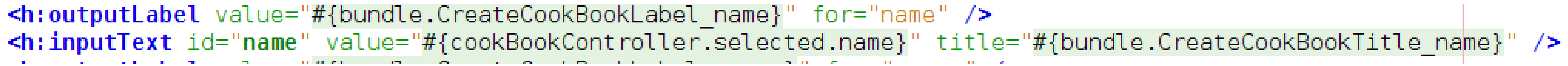


faces-config.xml



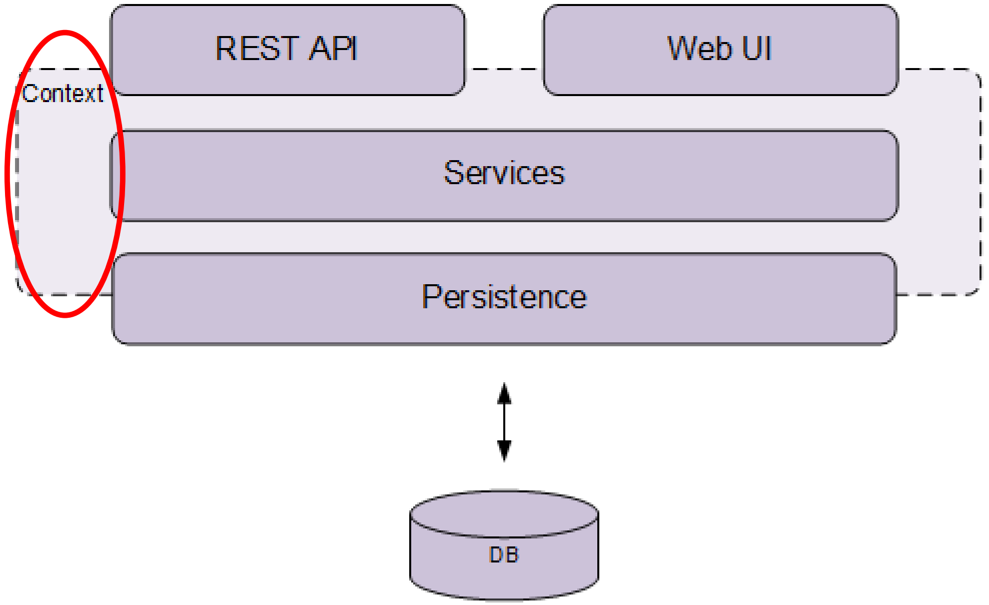
```
<application>
    <resource-bundle>
        <base-name>/translations</base-name>
        <var>bundle</var>
    </resource-bundle>
</application>
```

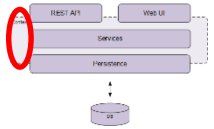
Create.xhtml

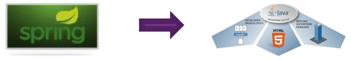


```
<h:outputLabel value="#{bundle.CreateCookBookLabel_name}" for="name" />
<h:inputText id="name" value="#{cookBookController.selected.name}" title="#{bundle.CreateCookBookTitle_name}" />
```

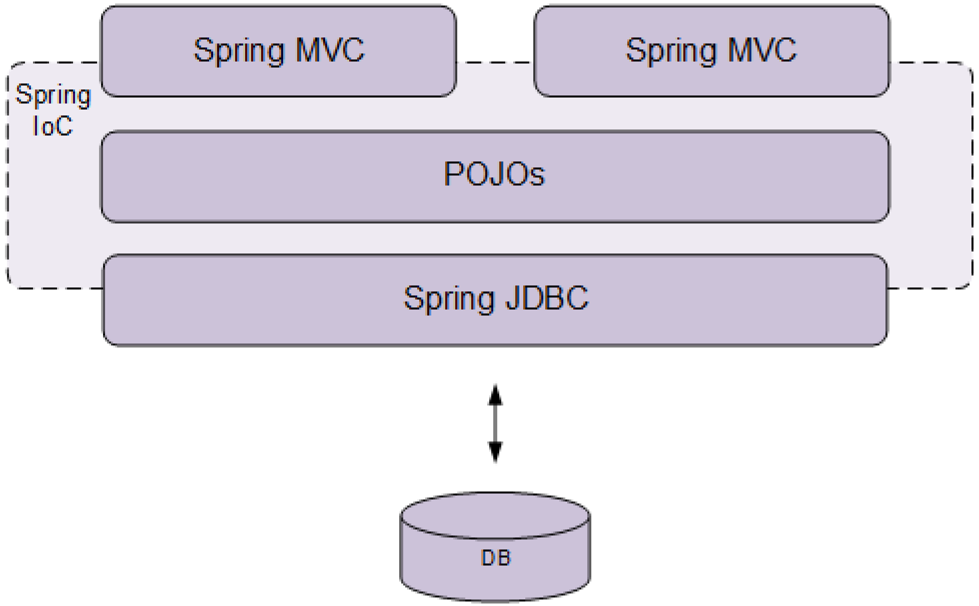
# Remove Spring



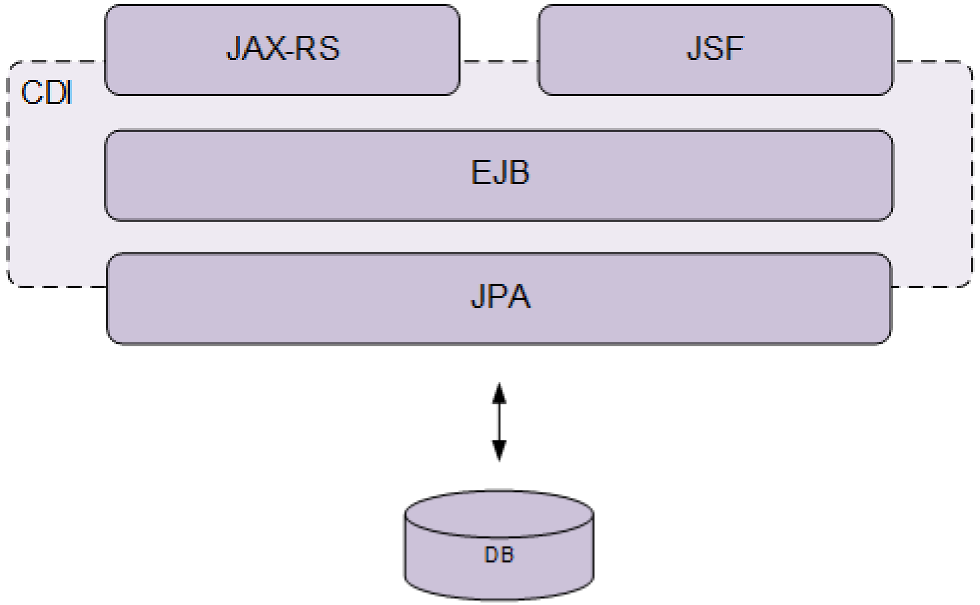


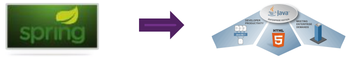


# Application Architecture - Spring



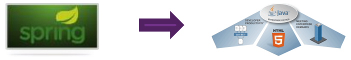
## Application Architecture – Java EE



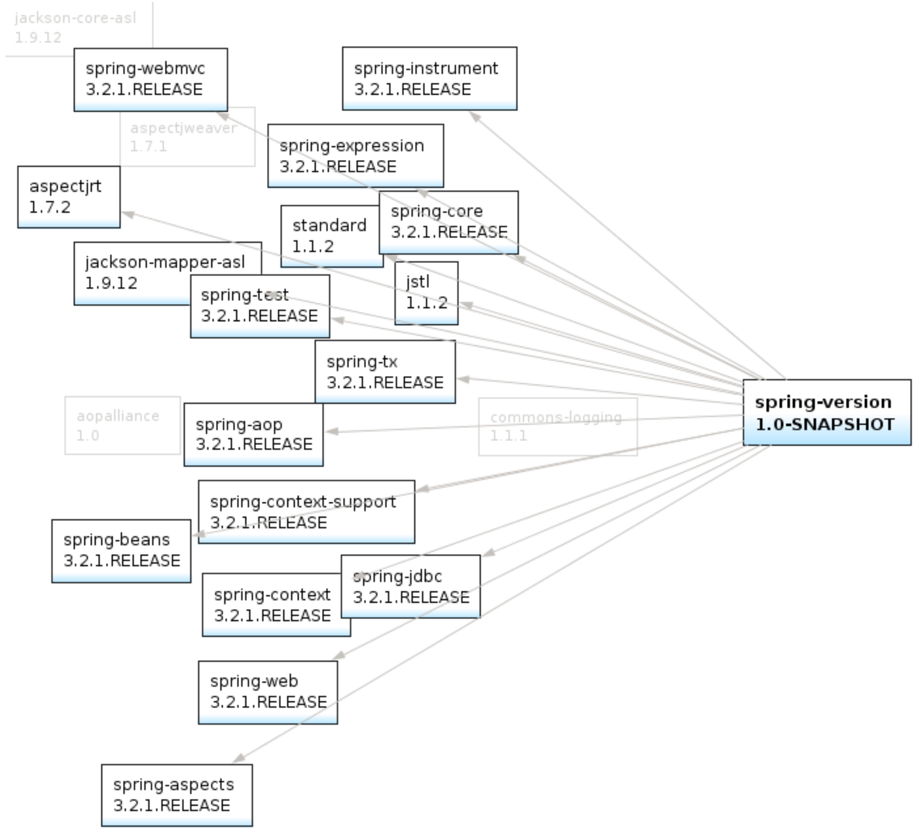


## Migration Steps – what we did

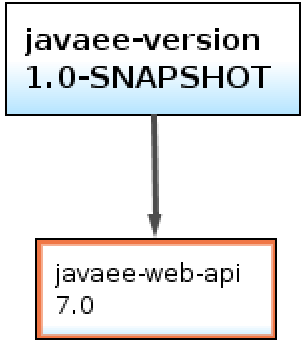
1. Added Java EE dependency
2. Layer for layer
    1. Added layer specific Java EE configuration
       (web.xml, persistense.xml, faces-config.xml)
    2. Added Java EE annotations
    3. Replaced Spring specific implementation with Java EE
    4. Replaced Spring injections with CDI
3. Removed Spring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

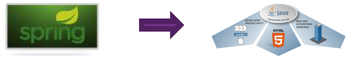


# Dependencies (Spring)

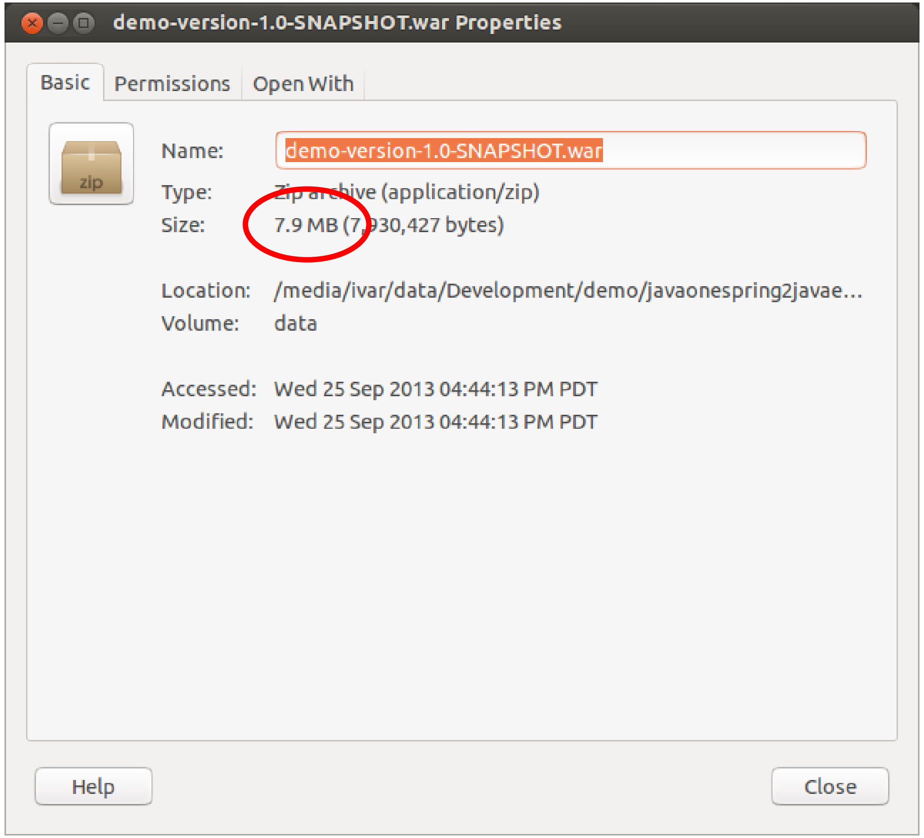

jackson-core-asl 1.9.12
spring-webmvc 3.2.1.RELEASE
spring-instrument 3.2.1.RELEASE
aspectjweaver 1.7.1
spring-expression 3.2.1.RELEASE
aspectjrt 1.7.2
standard 1.1.2
spring-core 3.2.1.RELEASE
jackson-mapper-asl 1.9.12
spring-test 3.2.1.RELEASE
jstl 1.1.2
spring-tx 3.2.1.RELEASE
aopalliance 1.0
spring-aop 3.2.1.RELEASE
commons-logging 1.1.1
spring-version 1.0-SNAPSHOT
spring-context-support 3.2.1.RELEASE
spring-beans 3.2.1.RELEASE
spring-jdbc 3.2.1.RELEASE
spring-context 3.2.1.RELEASE
spring-web 3.2.1.RELEASE
spring-aspects 3.2.1.RELEASE

## Dependencies (Java EE)





# WAR File (Spring)



demo-version-1.0-SNAPSHOT.war Properties

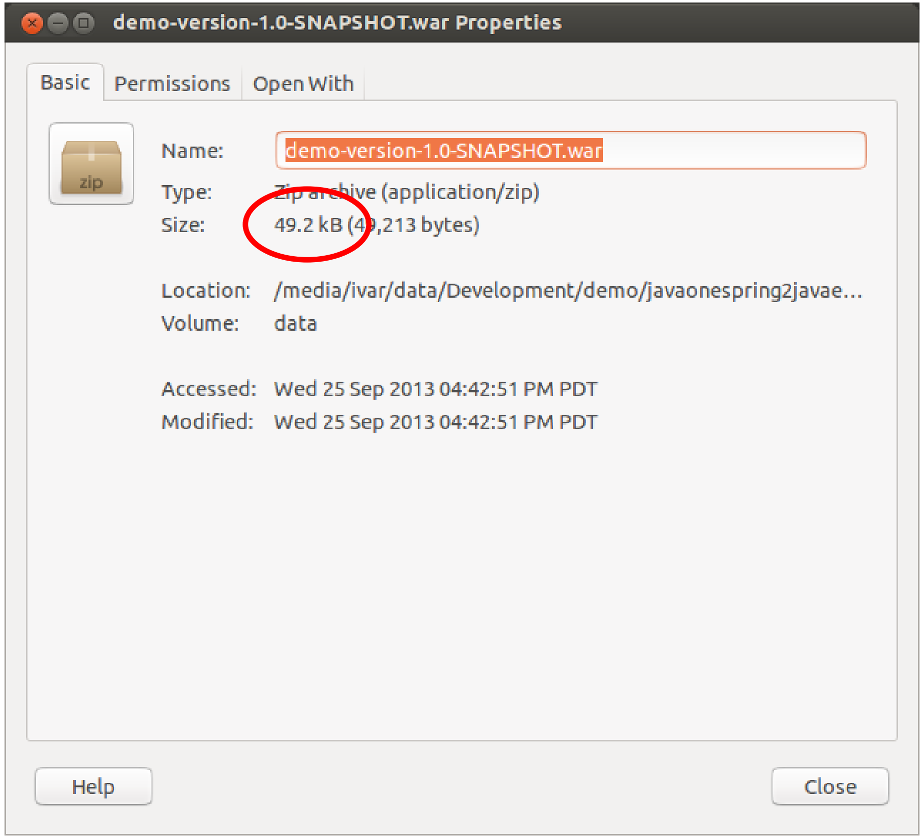
Basic | Permissions | Open With

Name: demo-version-1.0-SNAPSHOT.war
Type: Zip archive (application/zip)
Size: 7.9 MB (7,930,427 bytes)

Location: /media/ivar/data/Development/demo/javaonespring2javae...
Volume: data

Accessed: Wed 25 Sep 2013 04:44:13 PM PDT
Modified: Wed 25 Sep 2013 04:44:13 PM PDT

# WAR File (Java EE)


demo-version-1.0-SNAPSHOT.war Properties

Basic | Permissions | Open With

Name: demo-version-1.0-SNAPSHOT.war
Type: Zip archive (application/zip)
Size: 49.2 kB (49,213 bytes)

Location: /media/ivar/data/Development/demo/javaonespring2javae...
Volume: data

Accessed: Wed 25 Sep 2013 04:42:51 PM PDT
Modified: Wed 25 Sep 2013 04:42:51 PM PDT

## 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

spring

Java
ENTERPRISE EDITION
DEVELOPER PRODUCTIVITY
Java EE 7
HTML
MEETING ENTERPRISE DEMANDS

CYBERCOM GROUP

cybercom.com