Java EE vs. Spring

SOGETI

What we learned from Open Source

Sogeti Java

Erwin de Gier, Amsterdam, mei 2014

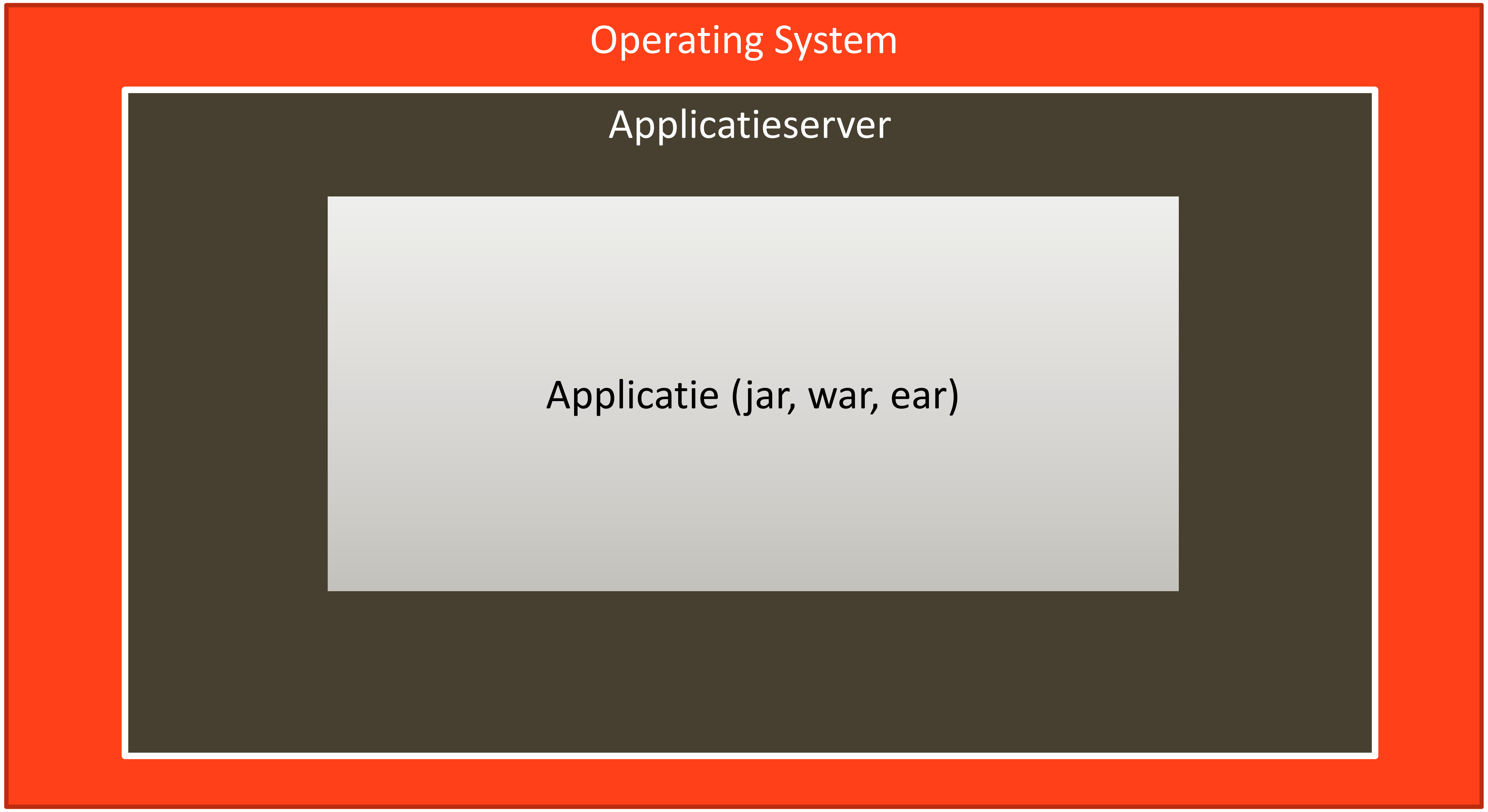
Contents

Java Enterprise applications

- (Web) Server applications
- Shared requirements
  - http(s), security, database, mail
- Open source libraries
- Dependencies
- Application Servers

SOGETI

Containers

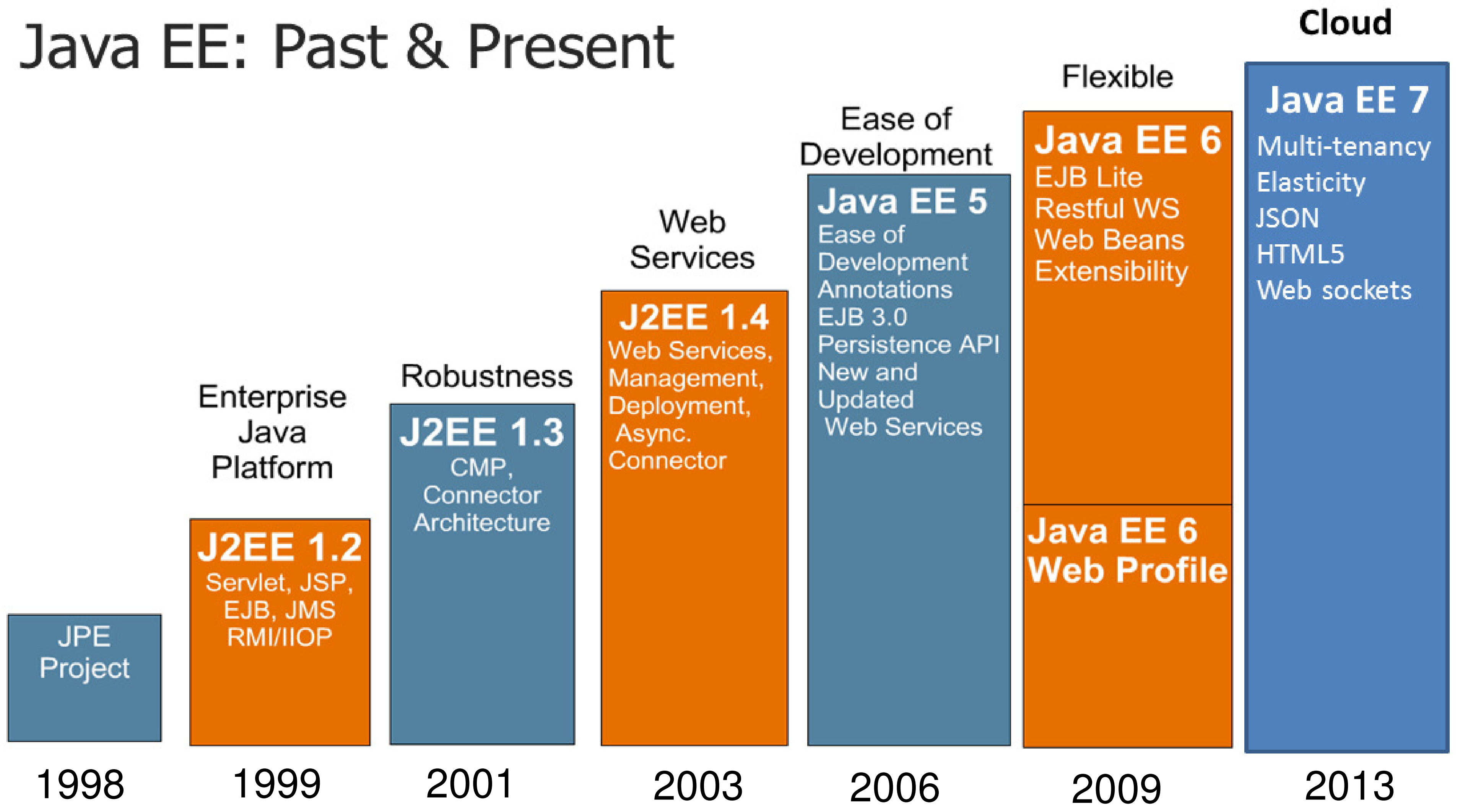

Operating System
Applicatieserver
Applicatie (jar, war, ear)

Web vs. JEE Containers

Web profile	Full profile
Servlet / JSP /JSF	Servlet / JSP /JSF
JPA	JPA
JAX-RS	JAX-RS
CDI	CDI
	JAX-WS
	EJB Full
	JMS
	JavaMail
	JAAS
	JAX-B
	JCA

SOGETI

Java EE history



Java EE: Past & Present
Cloud
Flexible
Ease of Development
Web Services
Robustness
Enterprise Java Platform
JPE Project
J2EE 1.2
Servlet, JSP, EJB, JMS RMI/IIOP
J2EE 1.3
CMP, Connector Architecture
J2EE 1.4
Web Services, Management, Deployment, Async. Connector
Java EE 5
Ease of Development Annotations EJB 3.0 Persistence API New and Updated Web Services
Java EE 6
EJB Lite Restful WS Web Beans Extensibility
Java EE 6 Web Profile
Java EE 7
Multi-tenancy Elasticity JSON HTML5 Web sockets
1998
1999
2001
2003
2006
2009
2013

Java EE early 2000's

- Complex
- Heavy weight
- Hard for developers
- Required expensive middleware
- Associated with vendor lock in

SOGETI

EJB 2 Example

```
package nl.sogeti;
import java.rmi.RemoteException;
import javax.ejb.*;

public class HelloBean implements SessionBean {
   private SessionContext sessionContext;
   public void ejbCreate() { }
   public void ejbRemove() { }
   public void ejbActivate() { }
   public void ejbPassivate() { }
   public void setSessionContext(SessionContext
    sessionContext) {
       this.sessionContext = sessionContext;
   }
   public String sayHello() throws
    java.rmi.RemoteException {
       return "Hello World!!!!!";
   }
}
```

EJB 2 Example

```java
package org.acme;
import java.rmi.*;
import javax.ejb.*;
import java.util.*;
public interface HelloHome extends EJBHome {
    public HelloObject create() throws RemoteException,
    CreateException;
}

package org.acme;
import java.rmi.*;
import javax.ejb.*;
import java.util.*;
public interface HelloObject extends EJBObject {
    public String sayHello() throws RemoteException;
}
```

ejb-jar.xml

```
<ejb-jar>
   <enterprise-beans>
      <session>
         <ejb-name>Hello</ejb-name>
         <home>nl.sogeti.HelloHome</home>
         <remote>nl.sogeti.HelloObject</remote>
         <ejb-class>nl.sogeti.HelloBean</ejb-class>
         <session-type>Stateless</session-type>
         <transaction-type>Container</transaction-type>
      </session>
   </enterprise-beans>
<assembly-descriptor> <container-transaction> <method> <ejb-
name>Hello</ejb-name> <method-name>*</method-name> </method>
<trans-attribute>Required</trans-attribute> </container-
transaction>
</assembly-descriptor>
</ejb-jar>
```

SOGETI

EJB 2 Example

```
Properties p = new Properties();
p.put("java.naming.factory.initial",
"org.openejb.client.RemoteInitialContextFactory");
p.put("java.naming.provider.url", "127.0.0.1:4201");
p.put("java.naming.security.principal", "myuser");
p.put("java.naming.security.credentials", "mypass");
InitialContext ctx = new InitialContext( p );
Object obj = ctx.lookup("/Hello");
HelloHome ejbHome = (HelloHome)
PortableRemoteObject.narrow(obj,HelloHome.class);
HelloObject ejbObject = ejbHome.create();
String message = ejbObject.sayHello();
```

SOGETI

How Spring saved Java EE

- Alternative to Java EE / EJB
- Spring is a Container
- Include Spring JAR’s as dependency in WAR
- Runs on servlet container (Tomcat, Jetty)

Spring injection example

```
<bean id="helloBean" class="nl.sogeti.HelloBean" />
<bean id="test" class="nl.sogeti.Test">
    <property name="helloBean" ref=" helloBean"/>
</bean>

public class Test {
HelloBean helloBean;

    public String processRequest(){
        return helloBean.sayHello();
    }
}
```

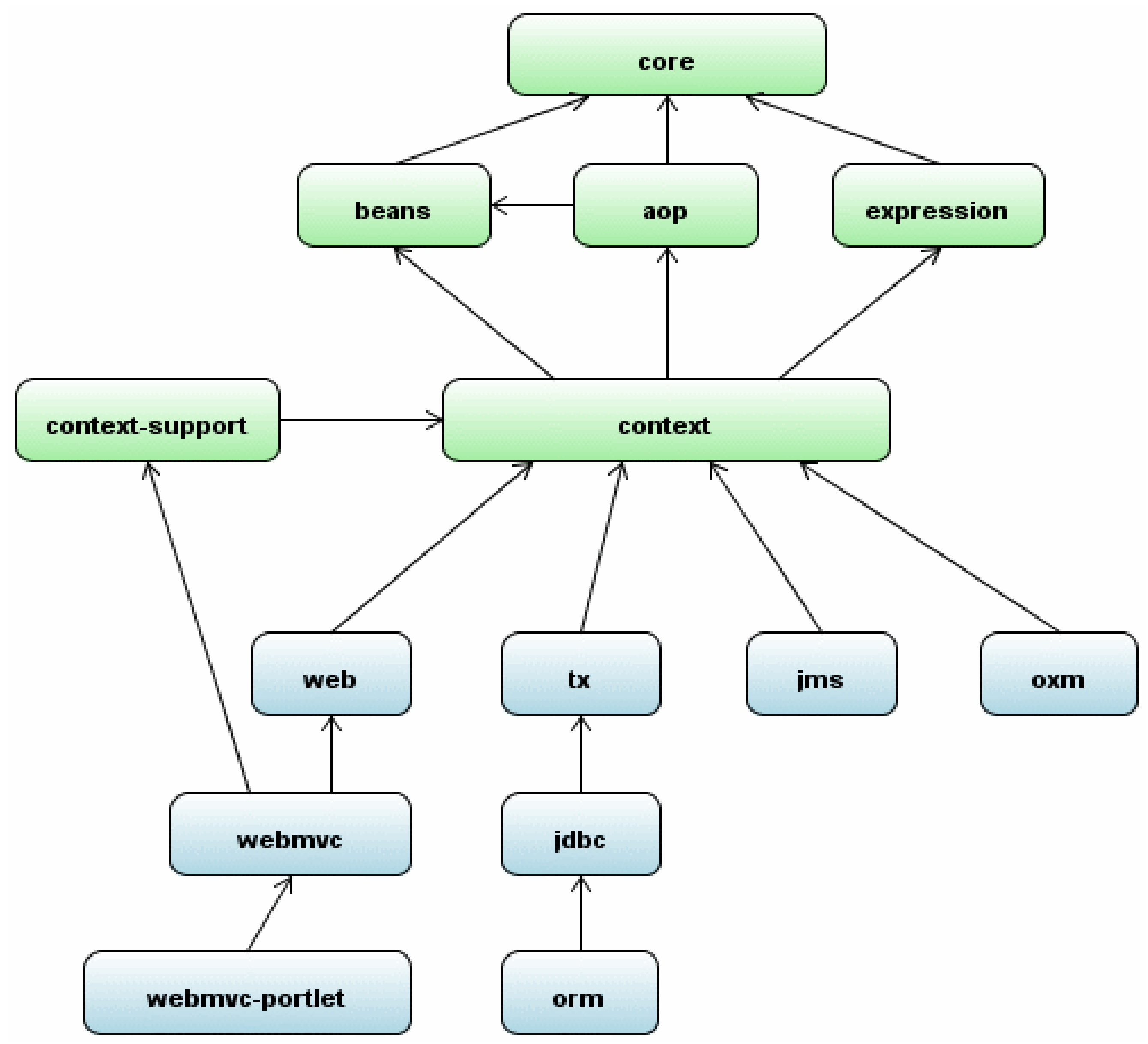
SOGETI

How Spring saved Java EE

- POJO based development
- Separation between application and configuration
- Light weight
- Open source
- Developer friendly
- Lots of functionality

SOGETI

Spring modules



Spring Core

- Inversion of Control (IoC)
- Inject object instances (wiring)
- Managing lifecycles

➢ *JEE alternative: CDI / EJB*

Spring injection example @

```
<context:annotation-config/>

<bean id=“helloBean“ class=“nl.sogeti.HelloBean“ />


public class Test {
@Autowired
HelloBean helloBean;

    public String processRequest(){
        return helloBean.sayHello();
    }
}
```

SOGETI

Spring configuration - Java

```
package nl.sogeti;
import org.springframework.context.annotation.*;

@Configuration
public class HelloWorldConfig {

    @Bean
    public HelloBean helloBean() {
        return new HelloBean();
    }
}
public class Test {
@Autowired
HelloBean helloBean;

    public String processRequest(){
        return helloBean.sayHello();
    }
}
```

Spring injection- @

```
package nl.sogeti;
import org.springframework.context.annotation.*;

@Component
public class HelloBean {
   public String sayHello() {
    return “hello world”;
   }
}
public class Test {
@Autowired
HelloBean helloBean;

   public String processRequest(){
      return helloBean.sayHello();
   }
}
```

SOGETI

JEE CDI Injection

```
package nl.sogeti;
import org.springframework.context.annotation.*;

@Named
public class HelloBean {
    public String sayHello() {
     return “hello world”;
    }
}
public class Test {
@Inject
HelloBean helloBean;

    public String processRequest(){
        return helloBean.sayHello();
    }
}
```

JEE vs. Spring: Bootstrapping

- Spring
  - *Servlet in web.xml*
  - *Java or XML configuration*
- JEE
  - *empty beans.xml*

SOGETI

JEE vs. Spring: core injection

- Spring
  - *Configuration required*
  - *Multiple styles*
- JEE
  - *No configuration*
  - *One consistent style*

Spring Transactions (tx)

- working with local and global transactions (without application server)
- working with nested transactions
- working with savepoints
- working in almost all environments of the Java platform

➢*JEE alternative: EJB / JTA*

SOGETI

Spring Transactions (tx)

```
<bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTrans
actionManager"/>
<tx:annotation-driven transaction-manager="txManager"/>


@Component
public class HelloWorld {

    @Transactional
    public String sayHelloWorld() {
        return "hello world";
    }
}
```

SOGETI

JEE Transactions

```
@Stateless
public class HelloWorld {

    public String sayHelloWorld() {
        return “hello world”;
    }
}
```

- Optional: @TransactionAttribute

Spring WebMVC

- Frontend framework
- Model View Controller
- Request based (**@RequestMapping**)
- Front controller
- Form Binding
- Validation

➢*JEE alternative: JSF / JAX-RS*

Spring WebMVC

```
@Controller
public class HelloWorldController {
@RequestMapping("/hello")
    public String hello(@RequestParam(value="name“) String
    name, Model model) {
        model.addAttribute("name", name);
        return "helloworld";
    }
}

<html><head>head>
<body>
    <h1>Hello : ${name}</h1>
</body>
</html
```

JEE JSF

```
@ManagedBean
public class HelloBean {
    @ManagedProperty(value = "#{param.name}")
    private String name;

    public String getName() {
     return name;
    }
}

<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h=http://java.sun.com/jsf/html>
    <h:head>
        <title>JSF 2.0 Hello World</title>
    </h:head>
    <h:body>
        <h1>Hello #{helloBean.name}</h1>
    </h:body>
</html>
```

JEE vs. Spring packaging

- Both use WAR / EAR file
- Both use web.xml (optional)
- Both use XML configuration (optional)
- Java EE does not need any packaged JAR files
- Spring needs a number of JAR files in WEB-INF/lib

Java EE vs. Spring libraries

Spring	JEE6
./WEB-INF/lib/aopalliance-1.0.jar ./WEB-INF/lib/aspectjrt-1.6.10.jar ./WEB-INF/lib/commons-fileupload-1.2.2.jar ./WEB-INF/lib/commons-io-2.0.1.jar ./WEB-INF/lib/el-api-2.2.jar ./WEB-INF/lib/hibernate-validator-4.1.0.Final.jar ./WEB-INF/lib/jackson-core-asl-1.8.1.jar ./WEB-INF/lib/jackson-mapper-asl-1.8.1.jar ./WEB-INF/lib/javax.inject-1.jar ./WEB-INF/lib/jcl-over-slf4j-1.6.1.jar ./WEB-INF/lib/jdom-1.0.jar ./WEB-INF/lib/joda-time-1.6.2.jar ./WEB-INF/lib/jstl-api-1.2.jar ./WEB-INF/lib/jstl-impl-1.2.jar ./WEB-INF/lib/log4j-1.2.16.jar ./WEB-INF/lib/rome-1.0.0.jar ./WEB-INF/lib/slf4j-api-1.6.1.jar ./WEB-INF/lib/slf4j-log4j12-1.6.1.jar ./WEB-INF/lib/spring-aop-3.1.0.RELEASE.jar ./WEB-INF/lib/spring-asm-3.1.0.RELEASE.jar ./WEB-INF/lib/spring-beans-3.1.0.RELEASE.jar ./WEB-INF/lib/spring-context-3.1.0.RELEASE.jar ./WEB-INF/lib/spring-context-support-3.1.0.RELEASE.jar ./WEB-INF/lib/spring-core-3.1.0.RELEASE.jar ./WEB-INF/lib/spring-expression-3.1.0.RELEASE.jar ./WEB-INF/lib/spring-web-3.1.0.RELEASE.jar ./WEB-INF/lib/spring-webmvc-3.1.0.RELEASE.jar ./WEB-INF/lib/validation-api-1.0.0.GA.jar	**None!**

Java EE vs. Spring artefacts

	Java EE 6	Spring
WAR File Size	0.021030 MB	10.87 MB (~516x)
Number of files	20	53 (> 2.5x)
Bundled libraries	0	36
Total size of libraries	0	12.1 MB
XML files	3	5
LoC in XML files	50 (11 + 15 + 24)	129 (27 + 46 + 16 + 11 + 19) (~ 2.5x)
Total .properties files	1 Bundle.properties	2 spring.properties, log4j.properties
Cold Deploy	5,339 ms	11,724 ms
Second Deploy	481 ms	6,261 ms
Third Deploy	528 ms	5,484 ms
Fourth Deploy	484 ms	5,576 ms
Runtime memory	~73 MB	~101 MB

	Java EE 6 Application Server	Spring Stack
Web Container		53 MB (tcServer 2.6.3)
Security		12 MB (Spring Security 3.1)
Persistence		6.3 MB (Hibernate 4.1)
Dependency Injection		5.3 MB (Framework)
Web Services		800 KB (Spring WS 2.0.4)
Messaging		3.4 MB (RabittMQ) 900 KB (Java client)
OSGI		1.4 MB (Spring OSGi 1.2.1)
Total	GlassFish (starting at 33 MB) JBoss 7 (starting at 65 MB) WildFly (starting at 15 MB)	83 MB

Java EE vs. Spring startup

Jboss EAP 6 with application ~ 2 seconds
GlassFish 3 with application ~ 4 seconds
Tomcat 6 + Spring application ~ 4 seconds

Java EE vs. Spring

- Spring was a great alternative for JEE
- Java EE >= 5 makes core Spring features obsolete
- Java EE requires less configuration
- Java EE requires less libraries

Java EE vs. Spring

- Freedom to choose container
- Vendor choice
- Production support
- Maintainability

Spring relevance

- Support for Java EE API’s in Spring (@Inject)
- Interoperability with EJB
- Innovation
- Runs on non – JEE containers

Spring innovation beyond EE

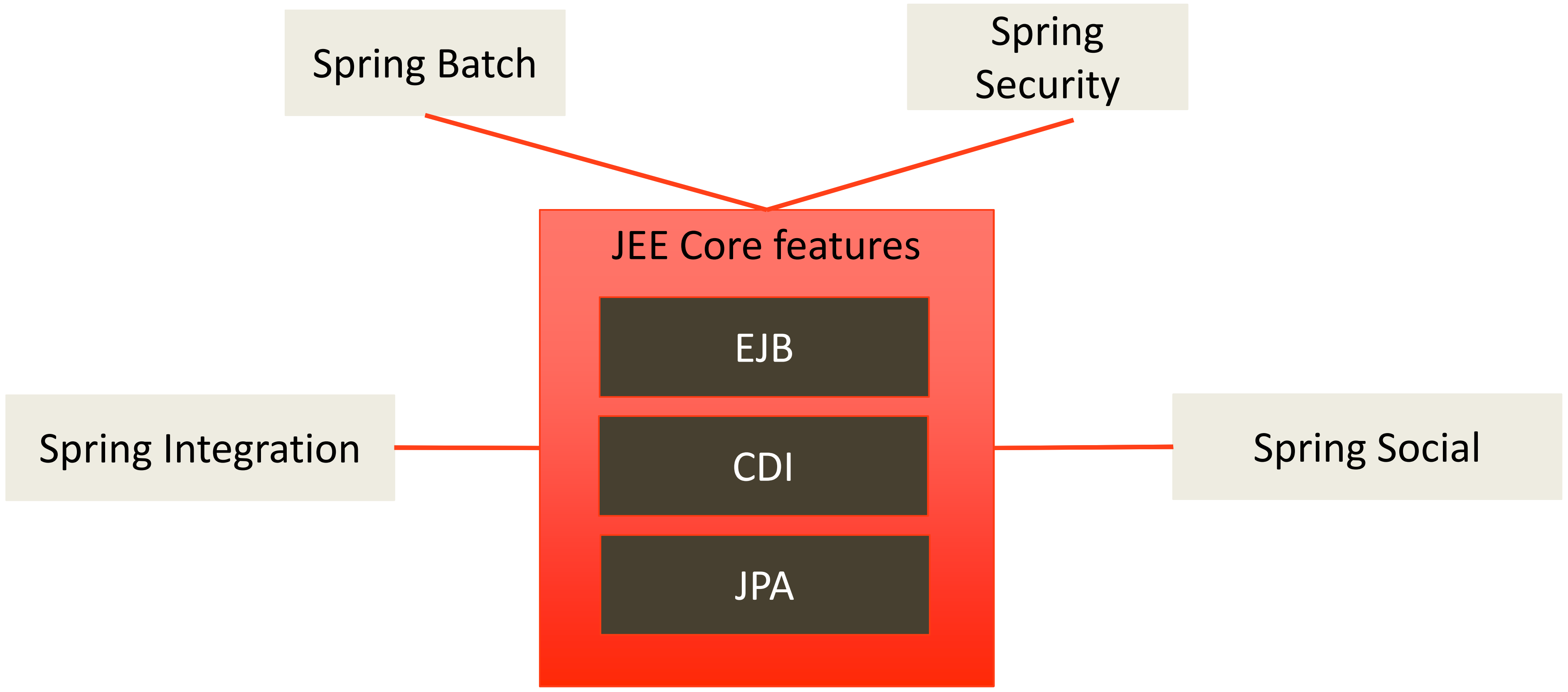
- Spring Batch DSL in Java EE 7 (JSR-352)
- Spring Integration
- Spring Social
- Spring Security
- Spring for Android
- Spring Data

http://spring.io/projects

When to use JEE / Spring

- Java EE is the standard
- Java EE does not require any libraries
- Always start with JEE on a Full JEE container
- Only use specific Spring features when required

JEE / Spring interoperability



JEE / Spring interoperability

- Inject EJB into Spring Managed Bean

```
<bean
    class="org.springframework.context.annotation.Common
AnnotationBeanPostProcessor">
    <property name="alwaysUseJndiLookup" value="true" />
 </bean>

@Resource(mappedName = HelloEJB.JNDI_NAME)
private HelloEJB helloEJB;
```

SOGETI

Spring Batch

- Framework for creating batch applications
- Create jobs with steps
- Read / Process/ Write

Spring Batch

```
<job id="ioSampleJob">
    <step name="step1">
        <tasklet>
        <chunk reader="fooReader" processor="fooProcessor"
            writer="compositeItemWriter" commit-
        interval="2">    </chunk>
        </tasklet>
    </step>
</job>
<bean id="compositeItemWriter"
class="...CustomCompositeItemWriter">
    <property name="delegate" ref="barWriter" />
</bean>
<bean id="barWriter" class="...BarWriter" />
```

SOGETI

Spring Social

- Connect with social networks
- e.g. Facebook, twitter, linkedin
- Login with social login
- Post tweets / messages
- Browse connections
- …

Spring Social

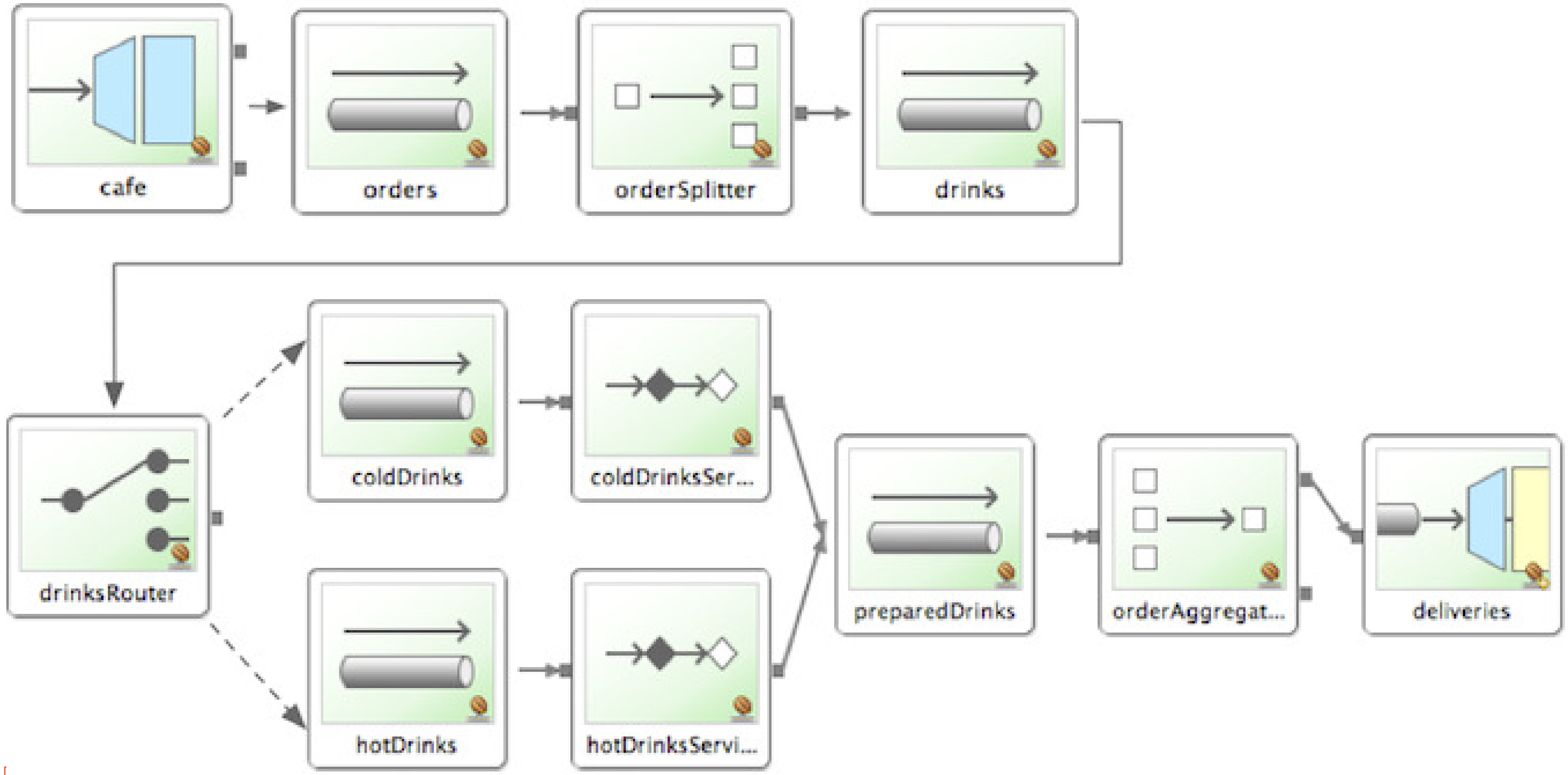
```
@Controller
public class HomeController {
   @Inject
   private final Facebook facebook;

   @RequestMapping(value = "/", method =
   RequestMethod.GET)
   public String home(Model model) {
      }  List<Reference> friends =
      facebook.friendOperations().getFriends();
      model.addAttribute("friends", friends);
      return "home";
    }
}
```

Spring Integration

- Framework for implementing messaging patterns
- Receive / route / send messages
- Connecting to ftp, http, email, twitter, queues, database, files, etc.

SOGETI

Spring Integration



Spring Integration

```
<beans>
    <int:gateway id="cafe" service-interface="o.s.i.samples.cafe.Cafe"/>
    <int:channel id="orders"/>
    <int:splitter input-channel="orders" ref="orderSplitter"
    method="split" output-channel="drinks"/>
    <int:channel id="drinks"/>
    <int:router input-channel="drinks" ref="drinkRouter"
    method="resolveOrderItemChannel"/>
    <int:channel id="coldDrinks"><int:queue capacity="10"/></int:channel>
    <int:service-activator input-channel="coldDrinks" ref="barista"
    method="prepareColdDrink" output-channel="preparedDrinks"/>
    <int:channel id="hotDrinks">
        <int:queue capacity="10"/>
    </int:channel>
    <int:service-activator input-channel="hotDrinks" ref="barista"
    method="prepareHotDrink" output-channel="preparedDrinks"/>
    <int:channel id="preparedDrinks"/> <int:aggregator input-
    channel="preparedDrinks" ref="waiter" method="prepareDelivery" output-
    channel="deliveries"/>
    <int-stream:stdout-channel-adapter id="deliveries"/>
</beans>
```

Summary

- Java EE is a standard for common enterprise functionalities
- Java EE implemented in application server
- Spring is an alternative, implementation as dependecies (jars)
- Java EE works out-of-the-box
- Spring requires configuration
- Spring scope is larger then JEE

Java EE, unless...

- you have a specific functionality that is not part of the JEE standard
- you don’t have a JEE server available

More information

Why is Java EE 6 better than Spring? – Arun Gupta
https://blogs.oracle.com/arungupta/entry/why_java_ee_6_is

Migrating Spring to Java EE6 – Bert Ertman, Paul Pakker
http://www.parleys.com/#st=5&id=2749&sl=1

Spring Framework
http://projects.spring.io/spring-framework/

Sogeti Java blog
https://java.sogeti.nl

SOGETI