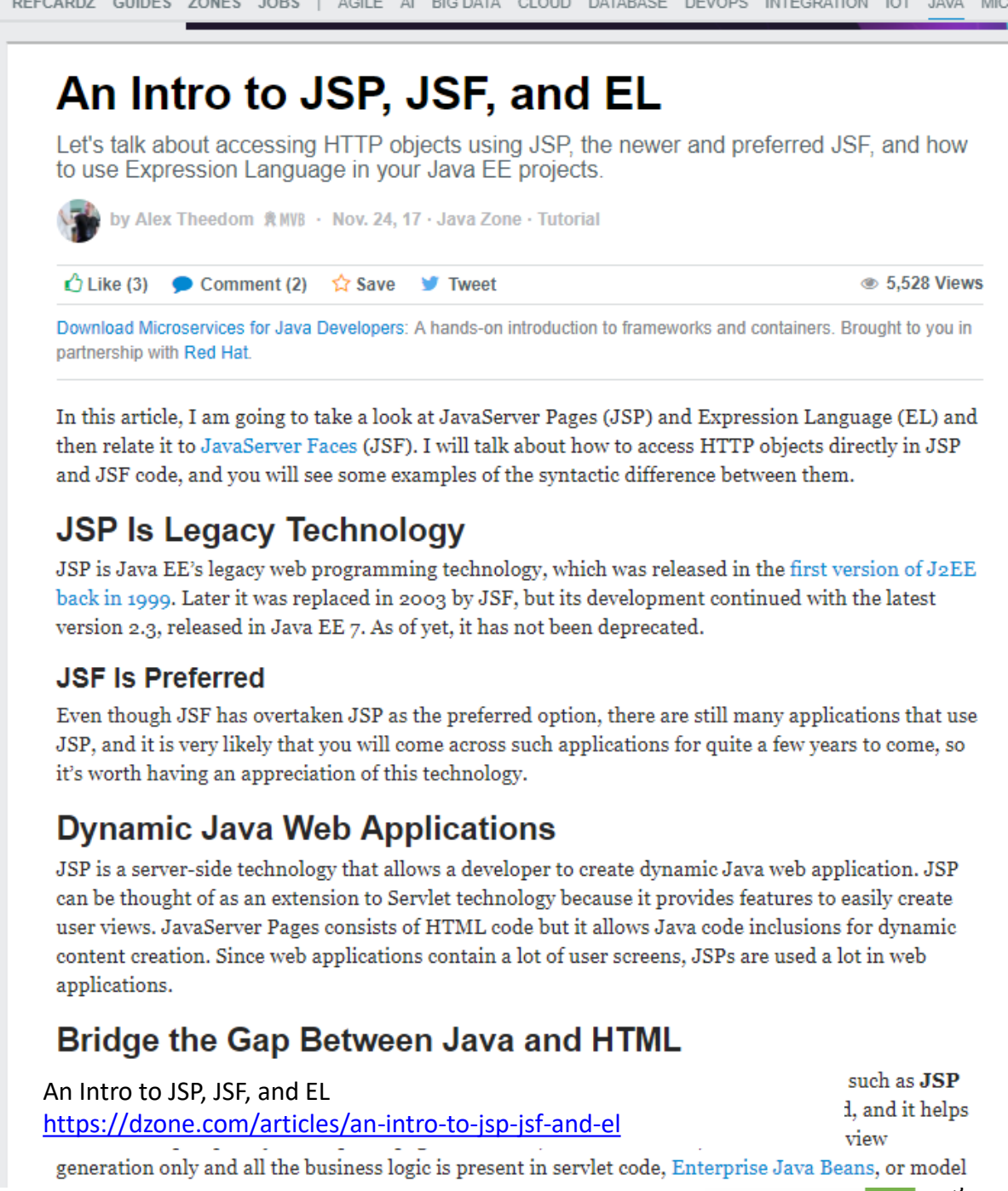
Java Server Faces

Java Server Faces (JSF)

- Building Web applications.
- Java Community Process (JCP)
- **Depends on CDI**
- Provides a component-centric approach
  - "Drag and drop" UI design
  - Collection of components
    - Rendering + specialized tags + event model (+ data model)
- Model-View-Controller (MVC) design pattern
- Runtime manages the dependencies & events

Java Se

- Building Web app
- Java Community
- **Depends on CDI**
- Provides a compo
  - "Drag and drop"
  - Collection of con
    - Rendering + spe
- Model-View-Con
- Runtime manage



REFCARDZ GUIDES ZONES JOBS | AGILE AI BIG DATA CLOUD DATABASE DEVOPS INTEGRATION IOT JAVA MO

An Intro to JSP, JSF, and EL

Let's talk about accessing HTTP objects using JSP, the newer and preferred JSF, and how to use Expression Language in your Java EE projects.

by Alex Theedom MVB · Nov. 24, 17 · Java Zone · Tutorial

Like (3) Comment (2) Save Tweet 5,528 Views

Download Microservices for Java Developers: A hands-on introduction to frameworks and containers. Brought to you in partnership with Red Hat.

In this article, I am going to take a look at JavaServer Pages (JSP) and Expression Language (EL) and then relate it to JavaServer Faces (JSF). I will talk about how to access HTTP objects directly in JSP and JSF code, and you will see some examples of the syntactic difference between them.

JSP Is Legacy Technology

JSP is Java EE's legacy web programming technology, which was released in the first version of J2EE back in 1999. Later it was replaced in 2003 by JSF, but its development continued with the latest version 2.3, released in Java EE 7. As of yet, it has not been deprecated.

JSF Is Preferred

Even though JSF has overtaken JSP as the preferred option, there are still many applications that use JSP, and it is very likely that you will come across such applications for quite a few years to come, so it's worth having an appreciation of this technology.

Dynamic Java Web Applications

JSP is a server-side technology that allows a developer to create dynamic Java web application. JSP can be thought of as an extension to Servlet technology because it provides features to easily create user views. JavaServer Pages consists of HTML code but it allows Java code inclusions for dynamic content creation. Since web applications contain a lot of user screens, JSPs are used a lot in web applications.

Bridge the Gap Between Java and HTML

such as **JSP**
l, and it helps
view
generation only and all the business logic is present in servlet code, Enterprise Java Beans, or model

An Intro to JSP, JSF, and EL
https://dzone.com/articles/an-intro-to-jsp-jsf-and-el

JSF: supported on Servlets

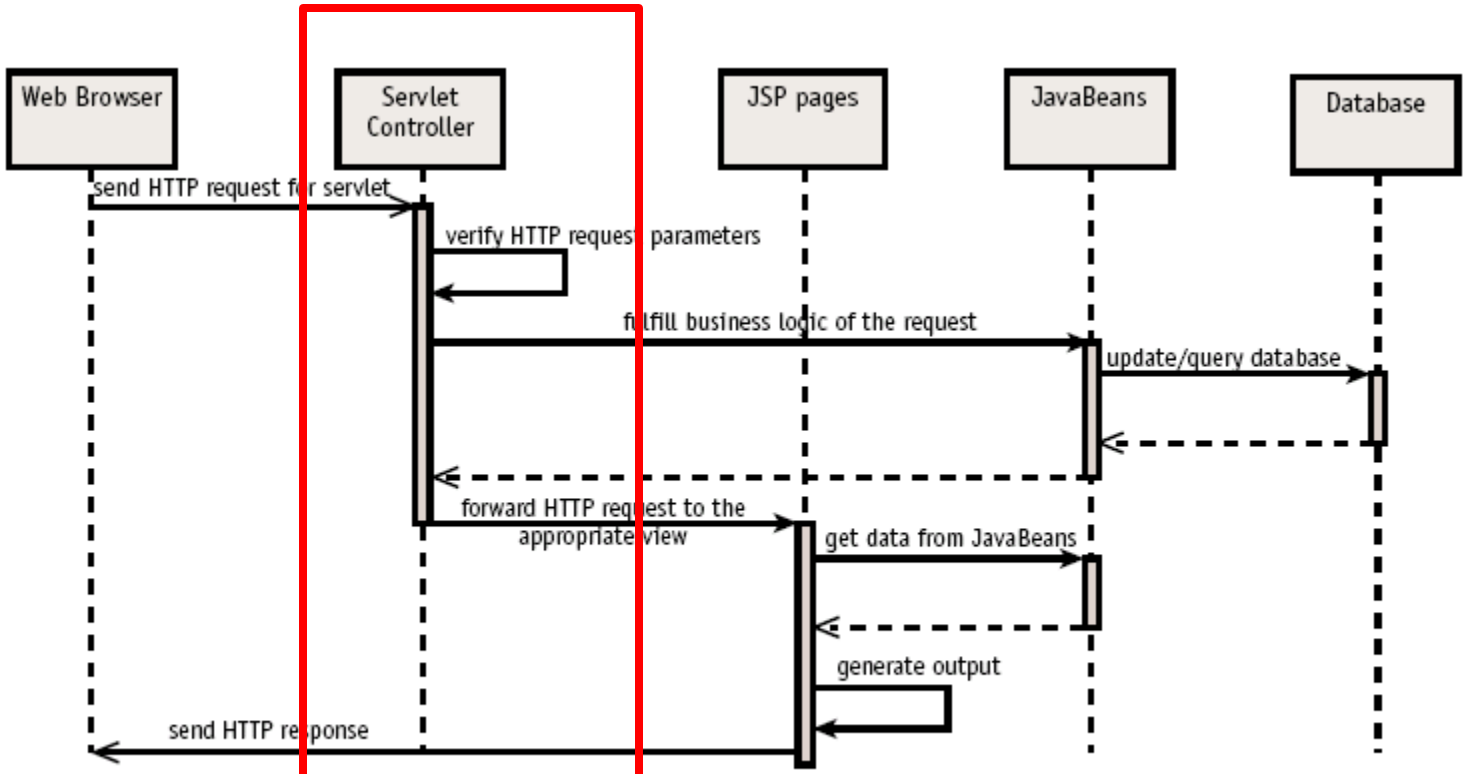
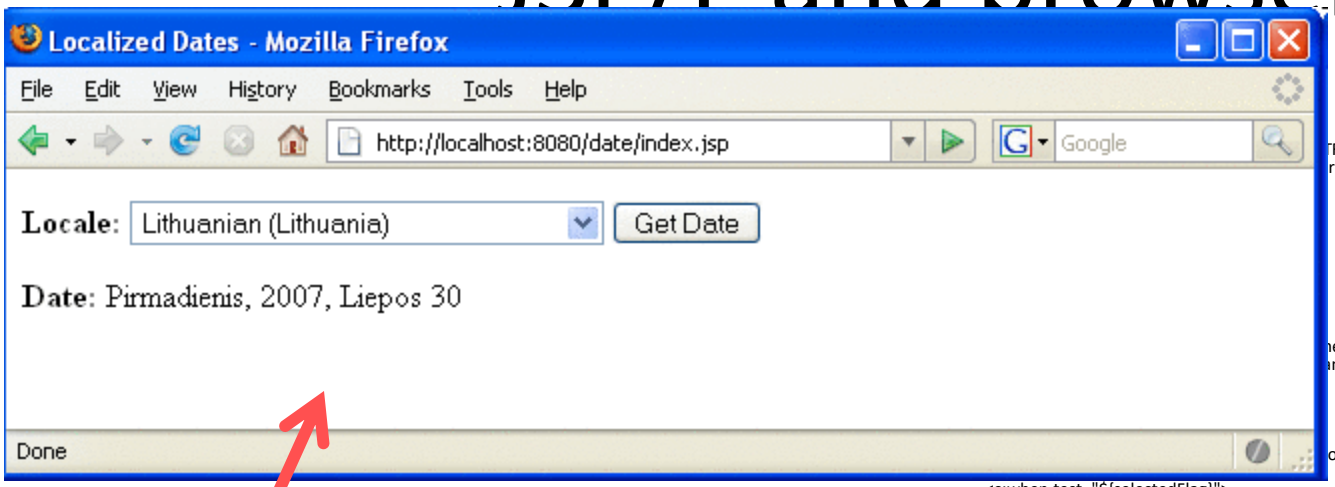


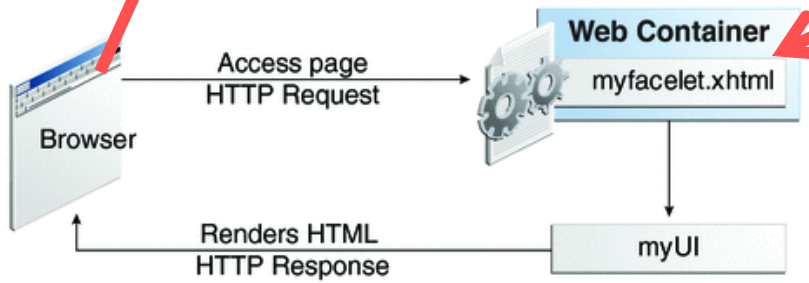
Figure 29-2 *UML Sequence diagram, Model 2 Architecture*

Xhtml/jsp is template
JSF servlet generates html
GUI present html

JSP/F and browser




```
TF-8" %>
re

ethod="post">
am.locale}" />

ocaleNames}" >
    <c:when test="${selectedFlag}">
      <c:choose>
        <c:when
            test="${f:equals(selectedLocaleString, localeString)}" >
          <option selected>${localeString}</option>
        </c:when>
        <c:otherwise>
          <option>${localeString}</option>
        </c:otherwise>
      </c:choose>
    </c:when>
    <c:otherwise>
      <option>${loc  String}</option>
    </c:otherwi  >
</c:cho  >
</c:  Each>
/select>
<input type="submit" name="Submit" value="Get Date">
</form>

<c:if test="${selectedFlag}" >
  <jsp:setProperty name="locales"
     property="selectedLocaleString"
     value="${selectedLocaleString}" />
  <jsp:useBean id="date" class="mypkg.MyDate"/>
  <jsp:setProperty name="date" property="locale"
     value="${locales.selectedLocale}"/>
  <b>Date: </b>${date.date}</c:if>
</body>
</html>
```



From http://netbeans.dzone.com/articles/develop-java-ee-6-app-jsf2

JSF = Servlet + JSF + …

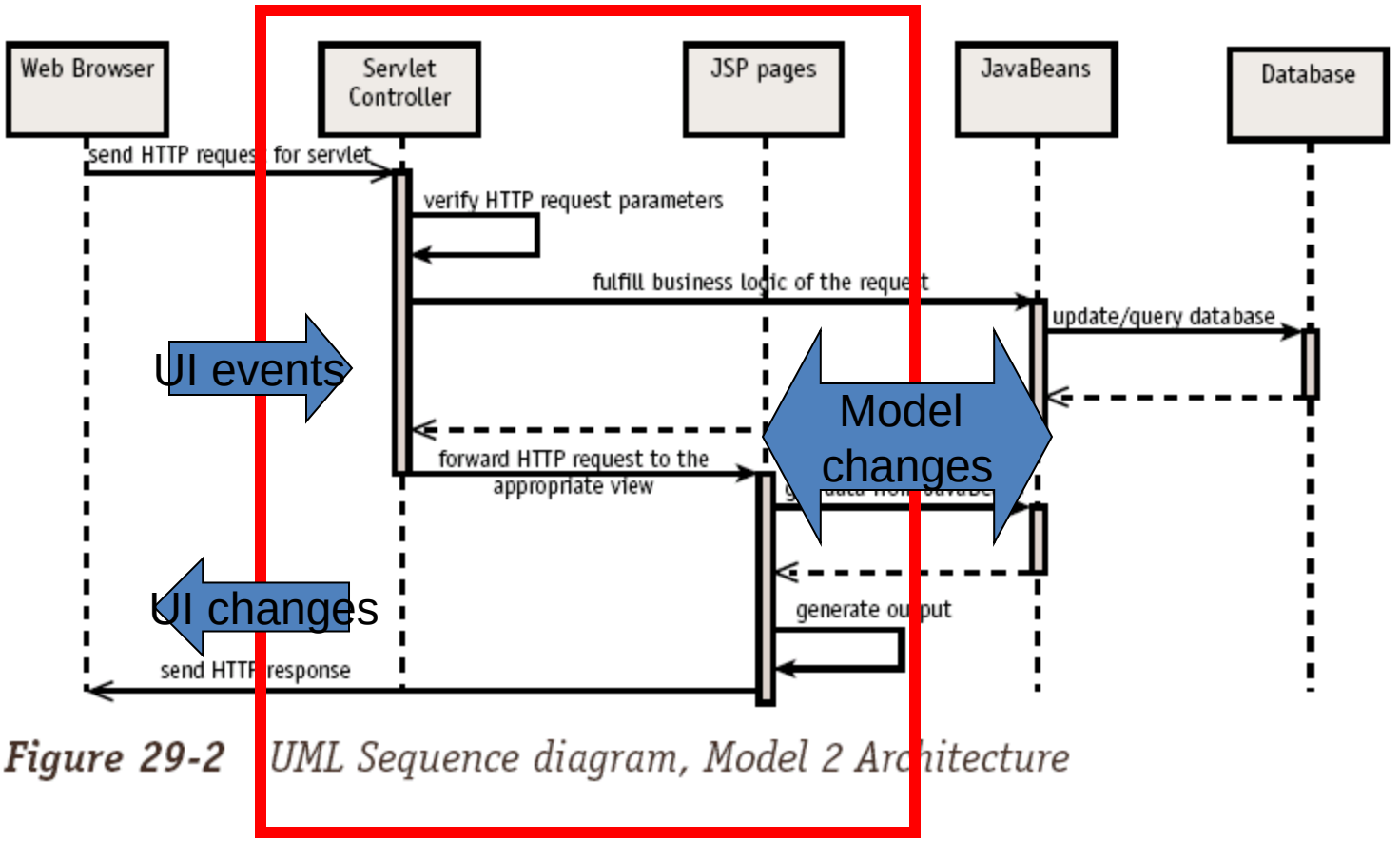
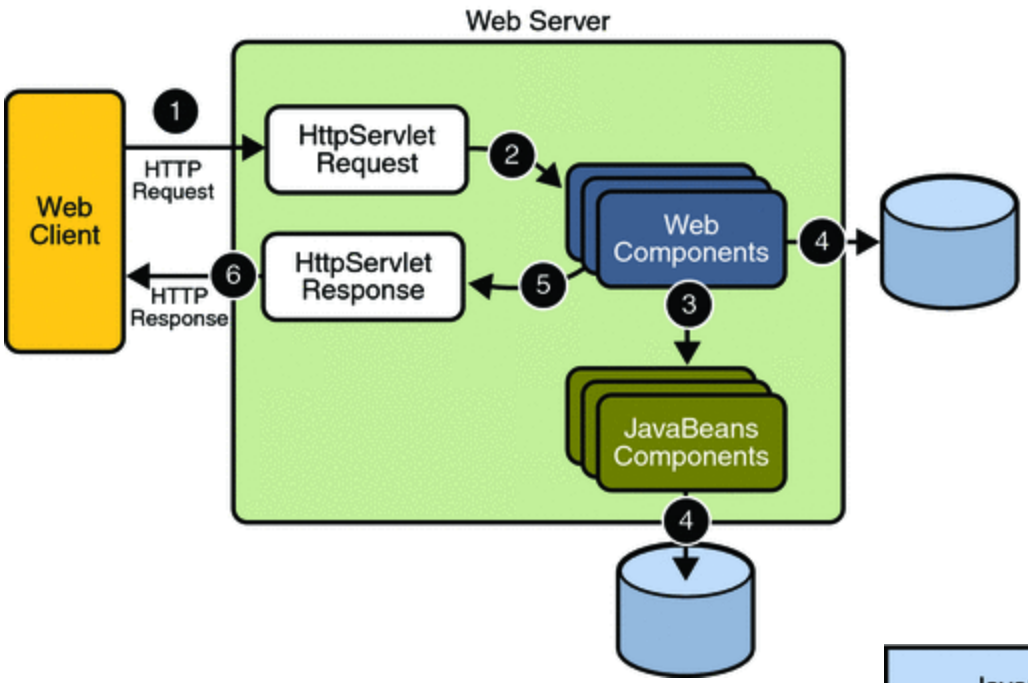
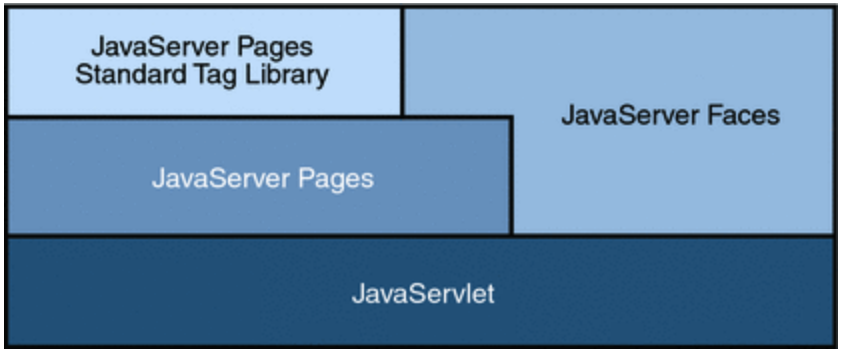


Figure 29-2 UML Sequence diagram, Model 2 Architecture

View Controller Model



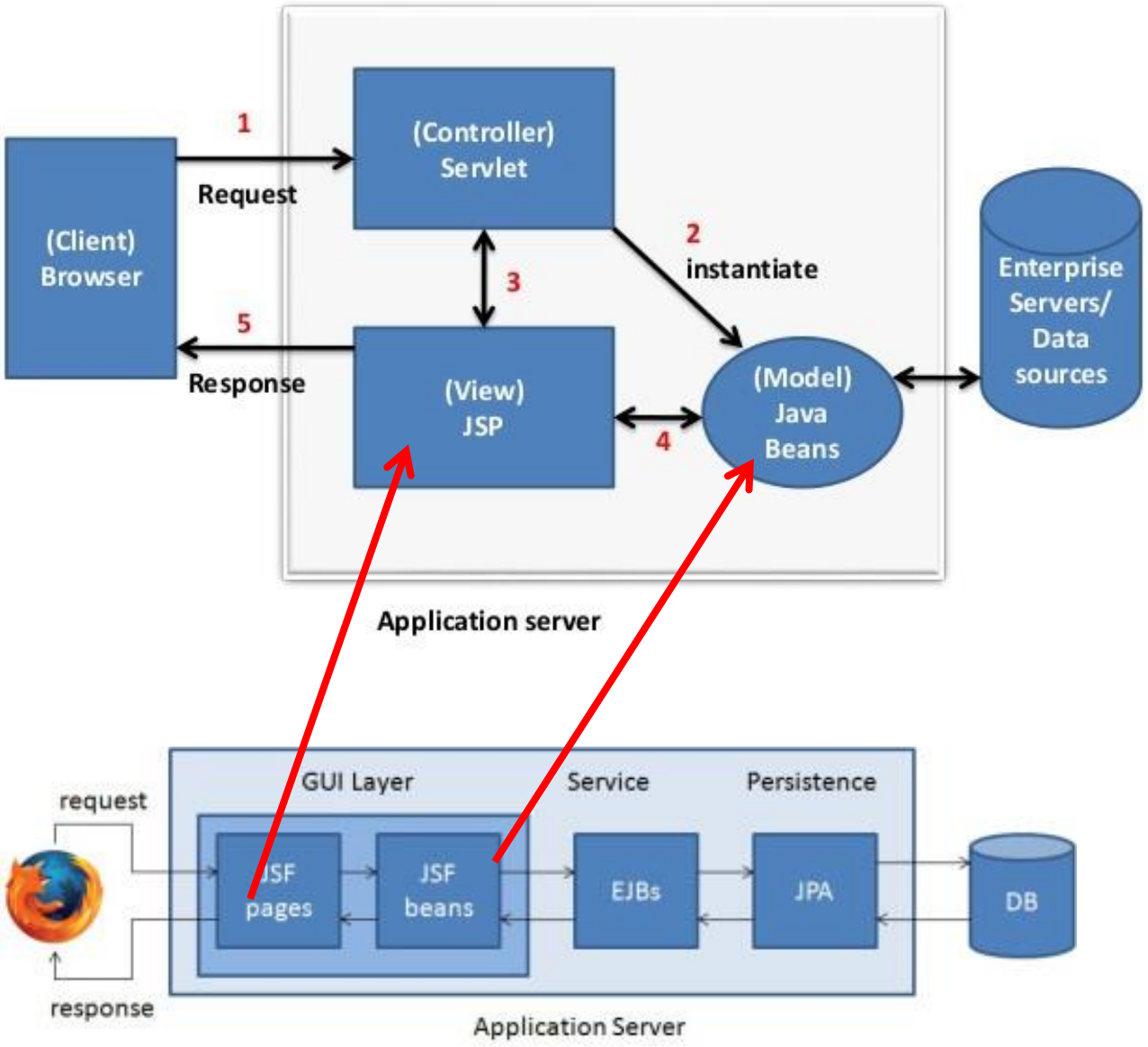


https://docs.oracle.com/cd/E19575-01/819-3669/bnadr/index.html

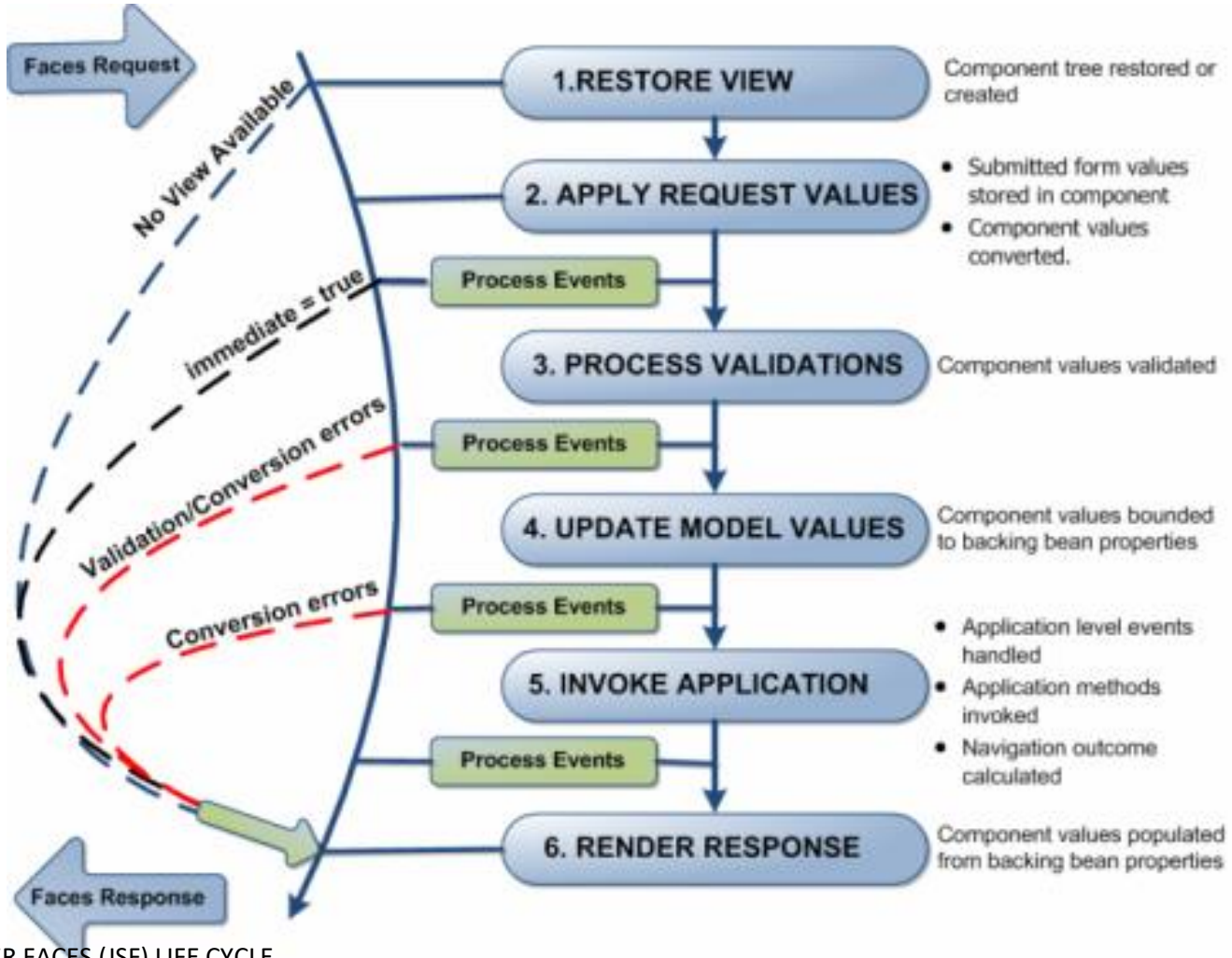
What Is a JSF Web UI?

- Web.xml
  - Support servlets, etc..
- Faces-config.xml (default name =
  - navigation rules, configures beans
  - other custom objects, such as custom components
- JSP pages (xhtml)
  - Custom objects such as custom components, validators, converters, or listeners.
  - custom tags for representing custom objects on the page
- backing beans (depend on CDI)

MVC Architecture

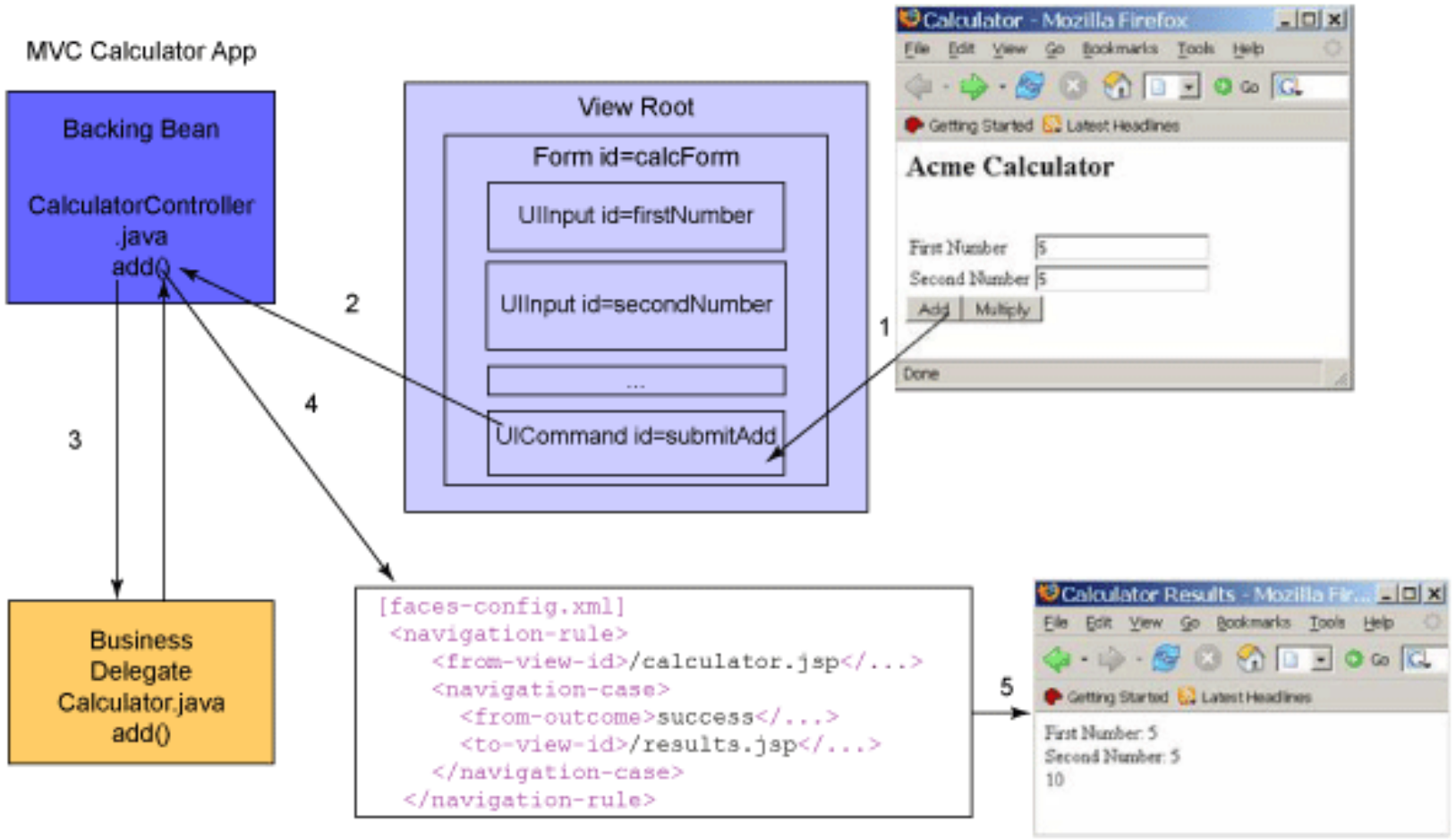


The JSF lifecycle: an overview



JAVA SERVER FACES (JSF) LIFE CYCLE
https://justforchangesake.wordpress.com/2014/05/18/java-server-faces-jsf-life-cycle/

JSF example


MVC Calculator App
Backing Bean
CalculatorController
.java
add()
Business
Delegate
Calculator.java
add()
View Root
Form id=calcForm
UIInput id=firstNumber
UIInput id=secondNumber
...
UICommand id=submitAdd
1
2
3
4
5
Calculator - Mozilla Firefox
Acme Calculator
First Number
Second Number
Add
Multiply
Done
[faces-config.xml]
<navigation-rule>
<from-view-id>/calculator.jsp</...>
<navigation-case>
<from-outcome>success</...>
<to-view-id>/results.jsp</...>
</navigation-case>
</navigation-rule>
Calculator Results - Mozilla Fir...
First Number: 5
Second Number: 5
10

Java based...

http://struts.apache.org/

http://www.zkoss.org/product/zk

http://primefaces.org/

http://myfaces.apache.org/

http://www.icesoft.org/java/projects/ICEfaces/

Java Server Faces based...

htt

duct/zk

http://myfaces.apache.org/

Open Source

http://www.icesoft.org/java/projects/ICEfaces/

Some remarks...

- More complex JSF controls follow a similar approach to Java Swing
  - specific data model requirement – usually an interface
  - MVC approach
    - The model changes, the view changes and vice-versa
    - JSF or Swing engine manage events
- JSF proposes the model and has some basic controls
  - There a plenty of JSF controls from different sources
    - Primefaces - http://www.primefaces.org/
      - (http://www.primefaces.org/showcase/)
    - Icefaces - http://www.icefaces.org/main/home/
      - (http://icefaces-showcase.icesoft.org/showcase.jsf)
    - RichFaces - http://jboss.org/richfaces
    - MyFaces - http://myfaces.apache.org/
  - There are JSF and AJAX compatible solutions
- Details on these can be found elsewhere

JSF and spring boot

- Both rely on servlets
- “native” servlets are not natural to spring boot
  - Servlet runtime as explicit beans ☹
- In javaEE
  - Web.xml
  - Faces-config.xml

```
public class Application extends SpringBootServletInitializer {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

    @Bean
    public ServletRegistrationBean servletRegistrationBean() {
        FacesServlet servlet = new FacesServlet();
        return new ServletRegistrationBean(servlet, "*.jsf");
    }

    @Bean
    public FilterRegistrationBean rewriteFilter() {
        FilterRegistrationBean rwFilter = new FilterRegistrationBean(new RewriteFi
        rwFilter.setDispatcherTypes(EnumSet.of(DispatcherType.FORWARD, DispatcherT
                DispatcherType.ASYNC, DispatcherType.ERROR));
        rwFilter.addUrlPatterns("/*");
        return rwFilter;
    }
}
```

[Developing JSF applications with Spring Boot
https://auth0.com/blog/developing-jsf-applications-with-spring-boot/](https://auth0.com/blog/developing-jsf-applications-with-spring-boot/)

JoinFaces

maven central 3.0.2 | build passing | coverage 98% | Dependency Status | javadoc 3.0.2 | license apache

This project enables JSF usage inside JAR packaged Spring Boot Application.

It autoconfigures PrimeFaces, PrimeFaces Extensions, BootsFaces, ButterFaces, RichFaces, OmniFaces, AngularFaces, Mojarra and MyFaces libraries to run at embedded Tomcat, Jetty or Undertow servlet containers.

It also aims to solve JSF and Spring Boot integration features. Current version includes JSF and CDI annotations support and Spring Security JSF Facelet Tag support.

How to use

JoinFaces Example shows JSF Spring Boot Starter usage. It may help you to choose the JSF Spring Boot Starter that fits your needs.

You can find more examples here.

JSF - PrimeFaces Example using Spring Boot and Maven
https://www.codenotfound.com/jsf-primefaces-example-spring-boot-maven.html
https://github.com/joinfaces/joinfaces#joinfaces

JoinFaces

maven central 3.0.2 | build passing | cove

This project enables JSF usage inside

It autoconfigures PrimeFaces, PrimeF and MyFaces libraries to run at embe

It also aims to solve JSF and Spring E Spring Security JSF Facelet Tag suppo

How to use

JoinFaces Example shows JSF Spring needs.

You can find more examples here.

CodeNotFound.com Blog Spring-Kafka Spring-WS Spring-JMS Apache CXF JAX-WS

JSF - PrimeFaces Example using Spring Boot and Maven

A detailed step-by-step tutorial on how to implement a PrimeFaces Example with Spring Boot and Maven.

Twitter
Facebook
Google+

PRIMEFACES

PrimeFaces is an open source component library for JavaServer Faces (JSF). It provides a collection of mostly visual components (widgets) that can be used by JSF programmers to build the UI for a web application. An overview of these widgets can be found at the PrimeFaces showcase.

In the following tutorial, we will configure, build and run a Hello

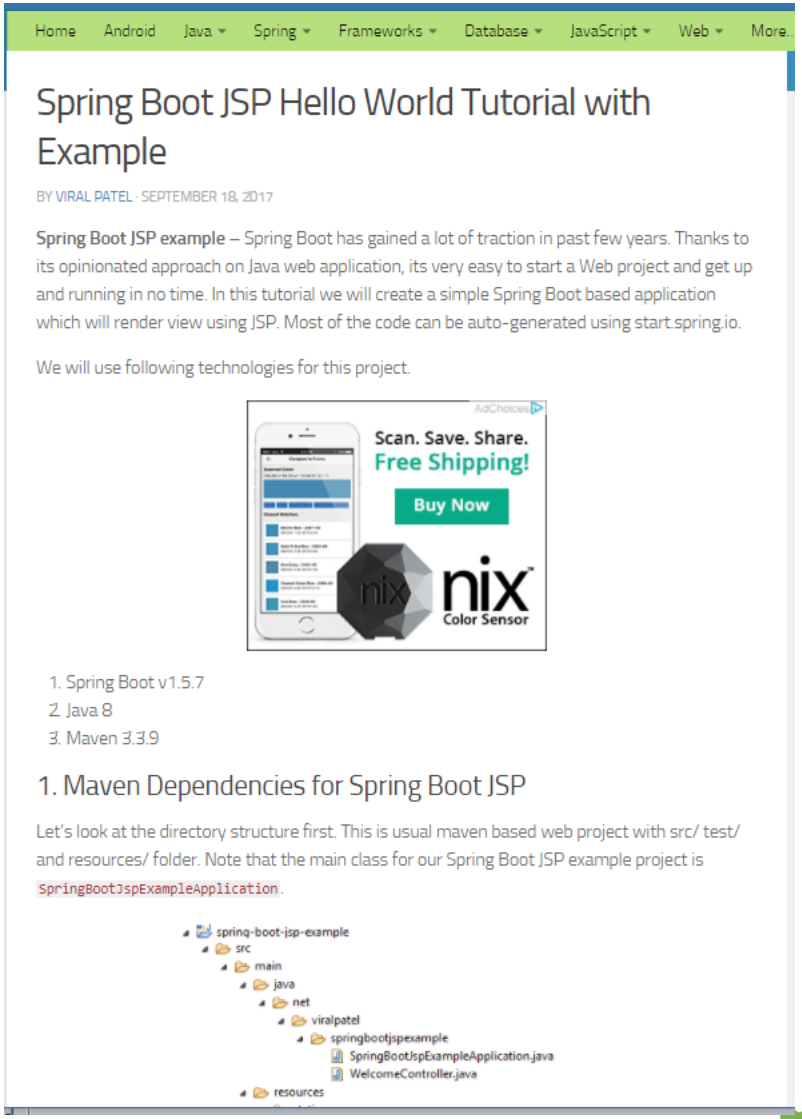
The JoinFaces project enables JSF usage inside a JAR packaged Spring Boot application. It auto-configures PrimeFaces, PrimeFaces Extensions, BootsFaces, ButterFaces, RichFaces, OmniFaces, AngularFaces, Mojarra and MyFaces libraries to run on embedded Tomcat, Jetty or Undertow servlet containers.

- JoinFaces 2.4
- Spring Boot 1.5
- Maven 3.5

JSF - PrimeFaces Example using Spring
https://www.codenotfound.com/jsf-p
https://github.com/joinfaces/joinface

JSP and Spring boot

- Support dynamic content as java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.
- spring boot must
  - access the resource
  - decode and publish
- Needs some configuration



Home Android Java Spring Frameworks Database JavaScript Web More

Spring Boot JSP Hello World Tutorial with Example

BY VIRAL PATEL · SEPTEMBER 18, 2017

Spring Boot JSP example – Spring Boot has gained a lot of traction in past few years. Thanks to its opinionated approach on Java web application, its very easy to start a Web project and get up and running in no time. In this tutorial we will create a simple Spring Boot based application which will render view using JSP. Most of the code can be auto-generated using start.spring.io.

We will use following technologies for this project.

1. Spring Boot v1.5.7
2. Java 8
3. Maven 3.3.9

1. Maven Dependencies for Spring Boot JSP

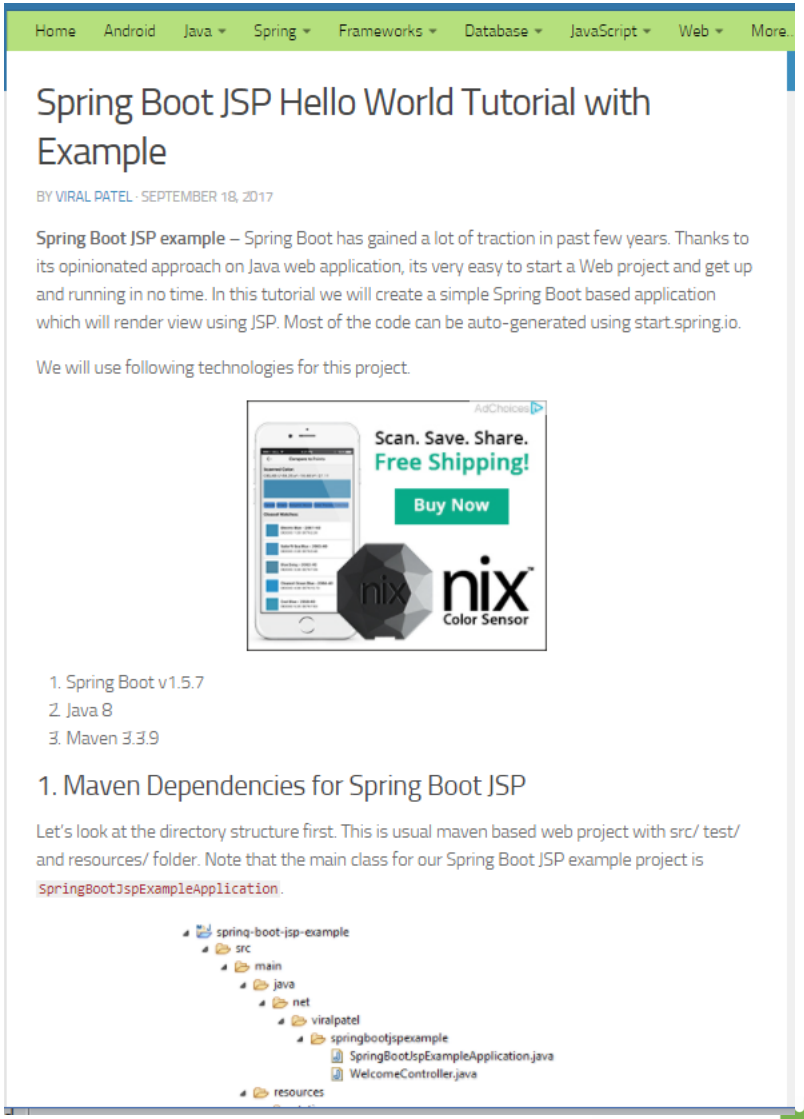
Let's look at the directory structure first. This is usual maven based web project with src/ test/ and resources/ folder. Note that the main class for our Spring Boot JSP example project is `SpringBootJspExampleApplication`.

spring-boot-jsp-example
src
main
java
net
viralpatel
springbootjspexample
SpringBootJspExampleApplication.java
WelcomeController.java
resources

Spring Boot JSP Hello World Tutorial with Example
http://viralpatel.net/blogs/spring-boot-jsp-hello-world-example/

JSP and Spring boot

- API of J2EE
- accept request and generate dynamic Response.
  - Support dynamic content as java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>.
  - implicit objects, predefined tags, expression language …
- **JSP and servlet are one or same**
  - translated into servlet at Run time.
- spring boot must
  - access the resource
  - decode and publish



Home Android Java Spring Frameworks Database JavaScript Web More

Spring Boot JSP Hello World Tutorial with Example

BY VIRAL PATEL · SEPTEMBER 18, 2017

Spring Boot JSP example – Spring Boot has gained a lot of traction in past few years. Thanks to its opinionated approach on Java web application, its very easy to start a Web project and get up and running in no time. In this tutorial we will create a simple Spring Boot based application which will render view using JSP. Most of the code can be auto-generated using start.spring.io.

We will use following technologies for this project.

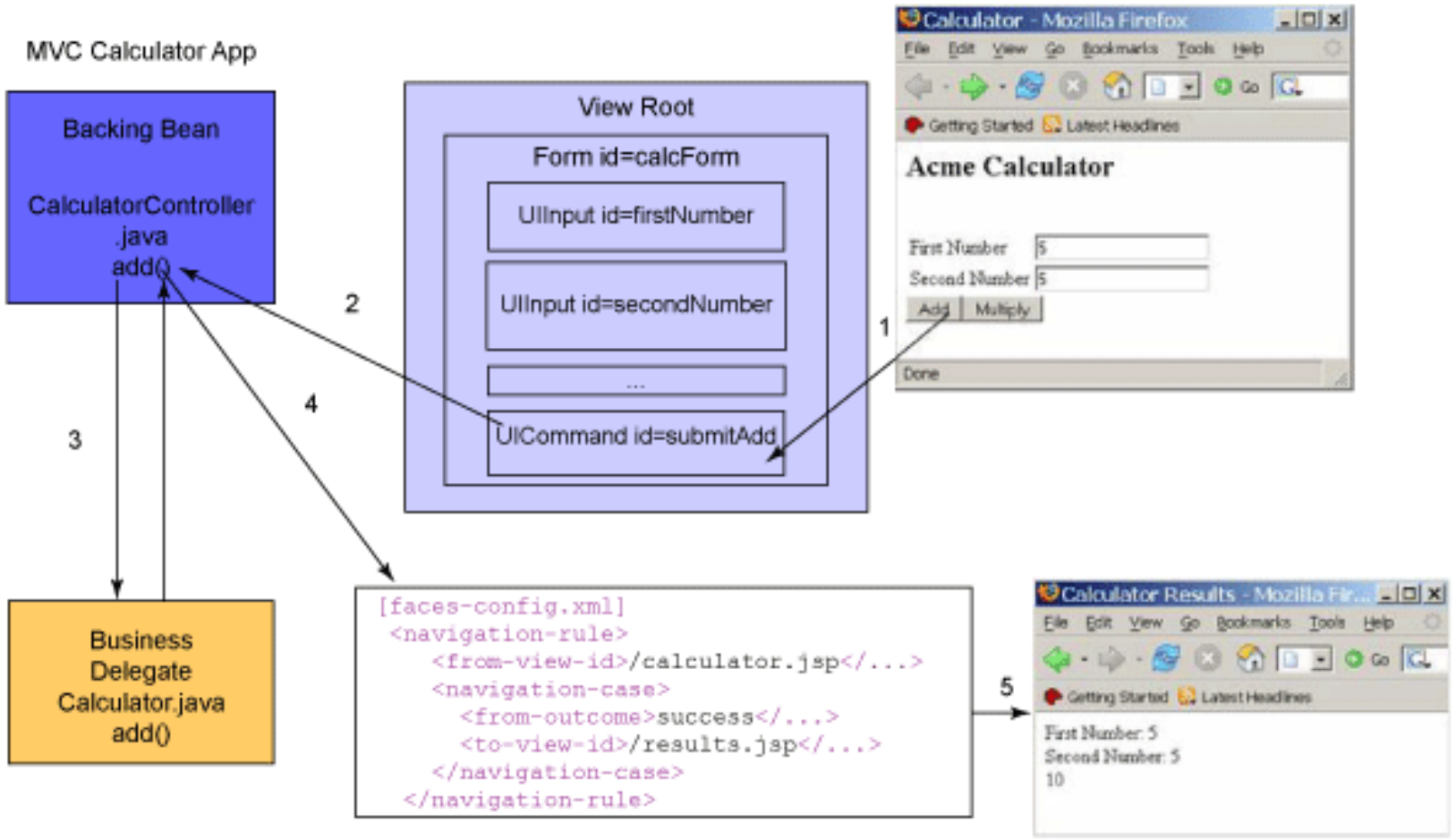
1. Spring Boot v1.5.7
2. Java 8
3. Maven 3.3.9

1. Maven Dependencies for Spring Boot JSP

Let's look at the directory structure first. This is usual maven based web project with src/ test/ and resources/ folder. Note that the main class for our Spring Boot JSP example project is `SpringBootJspExampleApplication`.

Spring Boot JSP Hello World Tutorial with Example
http://viralpatel.net/blogs/spring-boot-jsp-hello-world-example/

JSF example


MVC Calculator App
Backing Bean
CalculatorController
.java
add()
Business
Delegate
Calculator.java
add()
View Root
Form id=calcForm
UIInput id=firstNumber
UIInput id=secondNumber
...
UICommand id=submitAdd
1
2
3
4
5
Calculator - Mozilla Firefox
Acme Calculator
First Number
Second Number
Add
Multiply
Done
[faces-config.xml]
<navigation-rule>
<from-view-id>/calculator.jsp</...>
<navigation-case>
<from-outcome>success</...>
<to-view-id>/results.jsp</...>
</navigation-case>
</navigation-rule>
Calculator Results - Mozilla Fir...
First Number: 5
Second Number: 5
10

Declare bean management

faces-config.xml

```
<faces-config>
   ...
  <managed-bean>
    <description>
      The "backing file" bean that backs up the calculator
   webapp
    </description>
    <managed-bean-name>CalcBean</managed-bean-name>
    <managed-bean-class>
       com.arcmind.jsfquickstart.controller.CalculatorContr
   oller</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>

</faces-config>
```

Declare bean management: the alias

faces-config.xml

```
<faces-config>
   ...
  <managed-bean>
    <description>
      The "backing file" bean that backs up the calculator
 webapp
    </description>
    <managed-bean-name>CalcBean</managed-bean-name>
    <managed-bean-class>
       com.arcmind.jsfquickstart.controller.CalculatorContr
 oller</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>

</faces-config>
```

Declare bean management: the class

faces-config.xml

```
<faces-config>
   ...
  <managed-bean>
    <description>
      The "backing file" bean that backs up the calculator
   webapp
    </description>
    <managed-bean-name>CalcBean</managed-bean-name>
    <managed-bean-class>
       com.arcmind.jsfquickstart.controller.CalculatorContr
   oller</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>

</faces-config>
```

Declare bean management: the scope

faces-config.xml

```
<faces-config>
   ...
  <managed-bean>
    <description>
      The "backing file" bean that backs up the calculator
   webapp
    </description>
    <managed-bean-name>CalcBean</managed-bean-name>
    <managed-bean-class>
       com.arcmind.jsfquickstart.controller.CalculatorContr
   oller</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>

</faces-config>
```

Declare navigation rules

faces-config.xml

```
<navigation-rule>
  <from-view-id>/calculator.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/results.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Declare navigation rules: receiving ...

faces-config.xml

```
<navigation-rule>
  <from-view-id>/calculator.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/results.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

Declare navigation rules: go to...

faces-config.xml

```
<navigation-rule>
  <from-view-id>/calculator.jsp</from-view-id>
  <navigation-case>
    <from-outcome>success</from-outcome>
    <to-view-id>/results.jsp</to-view-id>
  </navigation-case>
</navigation-rule>
```

View the model object

```
package example.model;
public class Calculator {

    public int add(int a, int b) {
        return a + b;
    }


  public int multiply(int a, int b) {
        return a * b;
    }

}
```

Gluing the model and view (1)

```
Package example.controller;

import example.model.Calculator;

public class CalculatorController {

    private Calculator calculator = new
    Calculator();

    private int firstNumber = 0; /**
    First number used in operation. */

    private int result = 0; /** Result
    of operation on first number and
    second number. */

    private int secondNumber = 0; /**
    Second number used in operation. */

    public CalculatorController() {
    super();    }
```

```
public void setCalculator(Calculator
    aCalculator) { (...) }

public void setFirstNumber(int
    aFirstNumber) { (...) }
public int getFirstNumber() {
    (...)    }

public int getResult() {        (...)
    }

public void setSecondNumber(int
    aSecondNumber) {        (...)    }

public int getSecondNumber() {  (...) }

public String add() {            (...)
    }

public String multiply() {        (...)
    }
}
```

Gluing the model and view (2)

```
public void setCalculator(Calculator
    aCalculator) {
      this.calculator = aCalculator;
}
public void setFirstNumber(int
    aFirstNumber) {
  this.firstNumber = aFirstNumber;
}
public int getFirstNumber() {
  return firstNumber;
}
public int getResult() {
 return result;
}
public void setSecondNumber(int
    aSecondNumber) {
    this.secondNumber = aSecondNumber;
}
public int getSecondNumber() {
    return secondNumber;
}
```

```
public String add() {
         result =
    calculator.add(firstNumber,
    secondNumber);
    return "success";
}
public String multiply() {
    result =
    calculator.multiply(firstNumber,
    secondNumber);
    return "success";
}
```

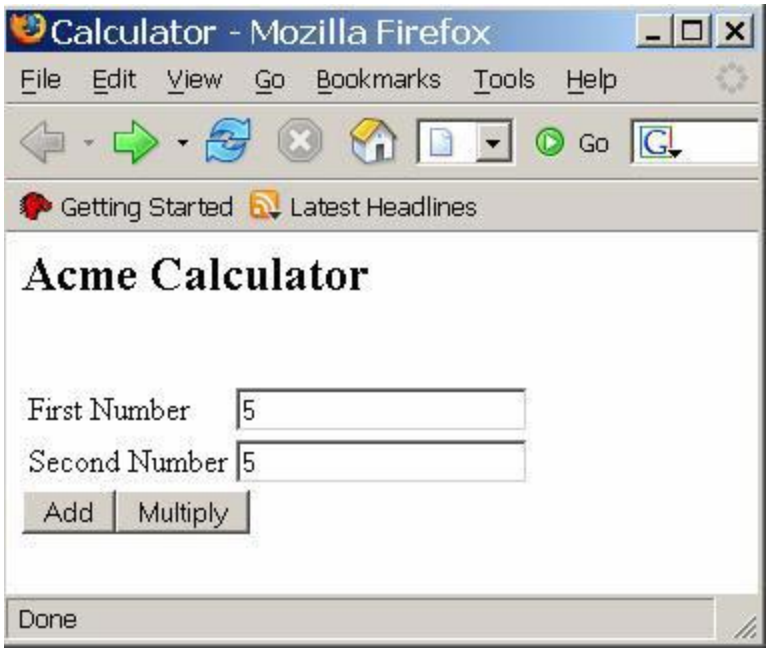
Create the index.jsp page

```
<jsp:forward page="/calc/calculator.jsp" />
```

Create the calculator.jsp page

Declare JSF stuff – do not intend to detail...

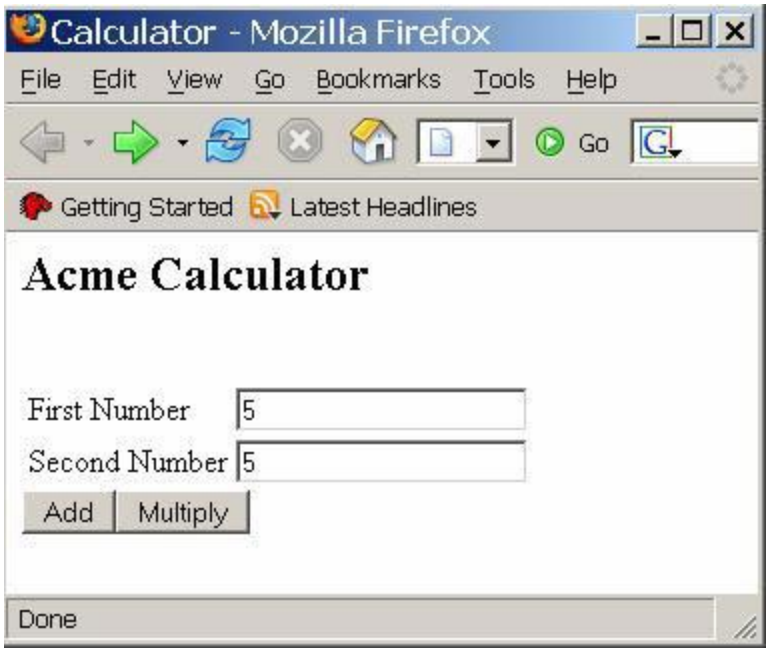
```
<%@ taglib
   uri="http://java.sun.com/jsf/ht
   ml" prefix="h" %>
<%@ taglib
   uri="http://java.sun.com/jsf/co
   re" prefix="f" %>
```



Create the calculator.jsp page

Create a JSF form

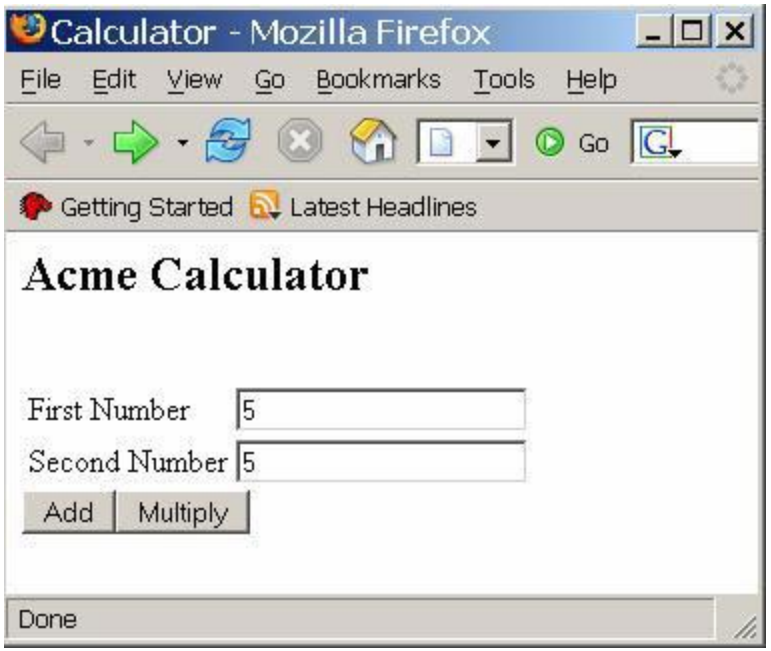
```
<f:view>
  <h:form id="calcForm">
     ...
  </h:form>
</f:view>
```



Create the calculator.jsp page

Add the UI – This an example. Exaustive illustration of all UI components is not the objective of this introduction

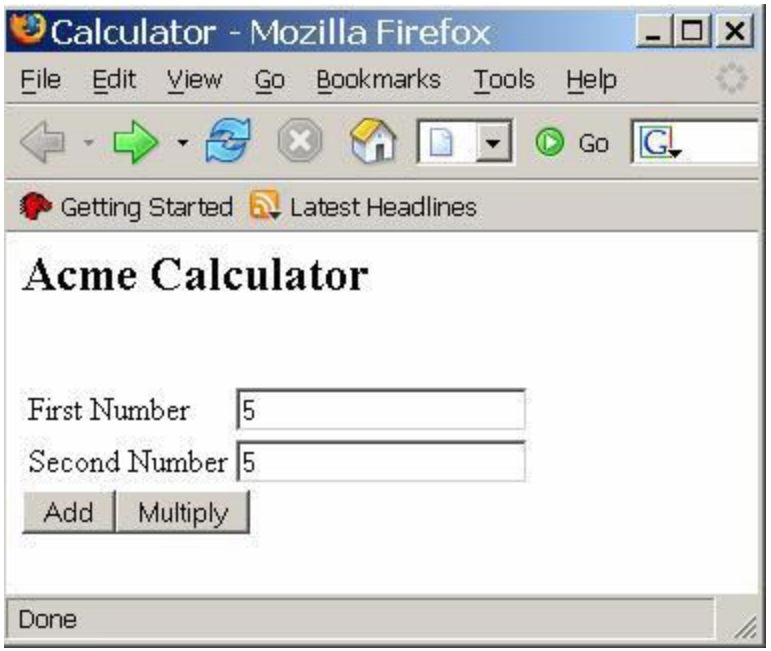
```
<h:panelGrid columns="3">
    <h:outputLabel value="First Number"
    for="firstNumber" />
    <h:inputText id="firstNumber"
    value="#{CalcBean.firstNumber}"
    required="true" />
        <h:message for="firstNumber" />

    <h:outputLabel value="Second
    Number" for="secondNumber" />
    <h:inputText id="secondNumber"
    value="#{CalcBean.secondNumber}"
    required="true" />
        <h:message for="secondNumber"
    />
</h:panelGrid>
```



Create the calculator.jsp page

Add the UI – add the buttons

```
<h:panelGroup>
    <h:commandButton
   id="submitAdd"
   action="#{CalcBean.add}"
   value="Add" />
    <h:commandButton
   id="submitMultiply"
   action="#{CalcBean.multiply}"
   value="Multiply" />
</h:panelGroup>
```



Create the results.jsp page

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
...
<f:view>
  First Number: <h:outputText id="firstNumber"
    value="#{CalcBean.firstNumber}"/>
  <br />
  Second Number: <h:outputText id="secondNumber"
    value="#{CalcBean.secondNumber}"/>
  <br />
  Result: <h:outputText id="result"
    value="#{CalcBean.result}"/>
  <br />
</f:view>
```

JSP vs Controller

```
<h:panelGrid columns="3">
    <h:outputLabel value="First
   Number" for="firstNumber" />
    <h:inputText id="firstNumber"
   value="#{CalcBean.firstNumber}"
   required="true" />
        <h:message
   for="firstNumber" />

    <h:outputLabel value="Second
   Number" for="secondNumber" />
    <h:inputText id="secondNumber"
   value="#{CalcBean.secondNumber}
   " required="true" />
        <h:message
   for="secondNumber" />
</h:panelGrid>
```

```
public class CalculatorController {

    private Calculator calculator = new
    Calculator();

    private int firstNumber = 0; /**
    First number used in operation. */

    private int result = 0; /** Result
    of operation on first number and
    second number. */

    private int secondNumber = 0; /**
    Second number used in operation. */
```

JSP vs Controller

```
<h:panelGroup>
    <h:commandButton
   id="submitAdd"
   action="#{CalcBean.add}"
   value="Add" />
    <h:commandButton
   id="submitMultiply"
   action="#{CalcBean.multiply}"
   value="Multiply" />
</h:panelGroup>
```

```
(...)
public void setCalculator(Calculator
   aCalculator) { (...) }

public void setFirstNumber(int
   aFirstNumber) { (...) }
public int getFirstNumber() {
   (...)    }

public int getResult() {        (...)
   }

public void setSecondNumber(int
   aSecondNumber) {        (...)    }

public int getSecondNumber() {  (...) }

public String add() {            (...)
   }

public String multiply() {        (...)
   }
```

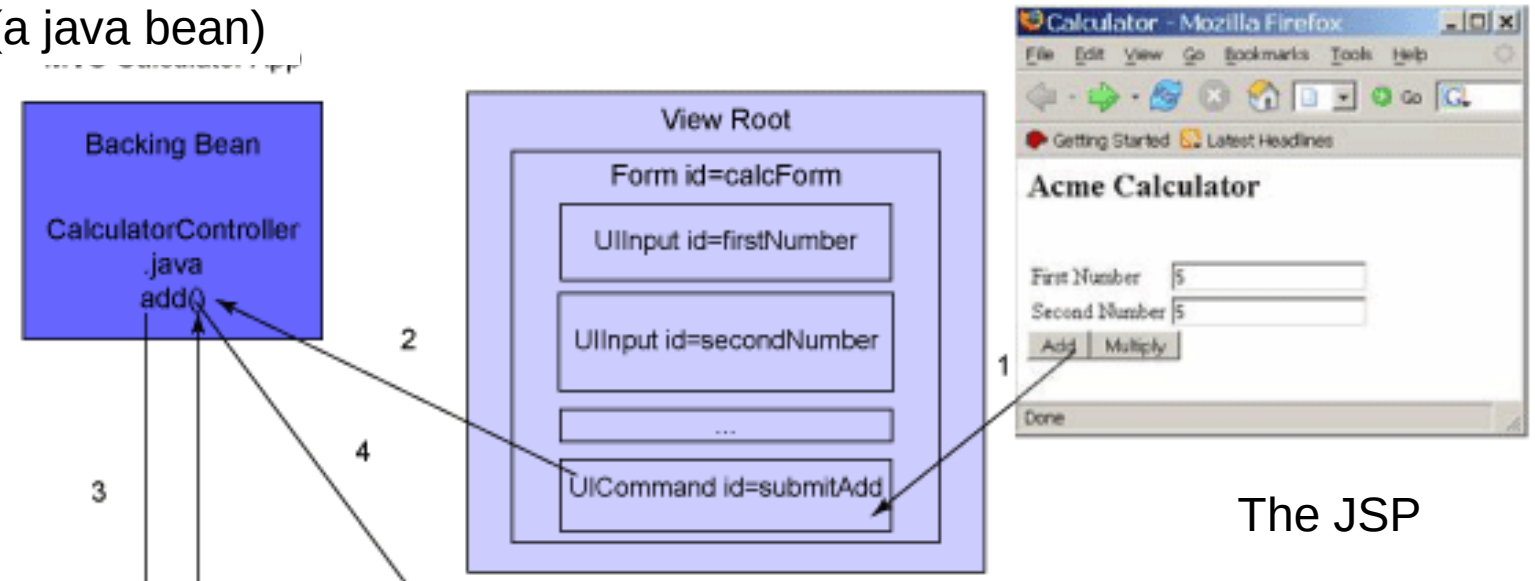
JSP vs Controller

```
<%@ taglib
    uri="http://java.sun.com/jsf/html"
    prefix="h" %>
<%@ taglib
    uri="http://java.sun.com/jsf/core"
    prefix="f" %>
...
<f:view>
  First Number: <h:outputText
    id="firstNumber"
    value="#{CalcBean.firstNumber}"/>
  <br />
  Second Number: <h:outputText
    id="secondNumber"
    value="#{CalcBean.secondNumber}"/>
  <br />
  Result: <h:outputText id="result"
    value="#{CalcBean.result}"/>
  <br />
</f:view>
```

```
public class CalculatorController {

    private Calculator calculator = new
    Calculator();

    private int firstNumber = 0; /**
    First number used in operation. */

    private int result = 0; /** Result
    of operation on first number and
    second number. */

    private int secondNumber = 0; /**
    Second number used in operation. */
```

A final overview

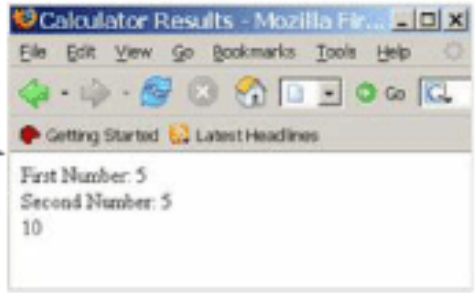
The controller
(a java bean)



The JSP

```
[faces-config.xml]
 <navigation-rule>
    <from-view-id>/calculator.jsp</...>
    <navigation-case>
      <from-outcome>success</...>
      <to-view-id>/results.jsp</...>
    </navigation-case>
 </navigation-rule>
```

5



The resources

The Flow

Some references

- JSF 2.0 Tutorial (See Navigation sections)
  - http://www.mkyong.com/tutorials/jsf-2-0-tutorials/
- JSF tutorial
  - http://www.tutorialspoint.com/jsf/index.htm
- JSF PrimeFaces Tutorial
  - https://www.journaldev.com/2990/jsf-primefaces-tutorial

- Other links – curated
  - https://tinyurl.com/y9m48wvd

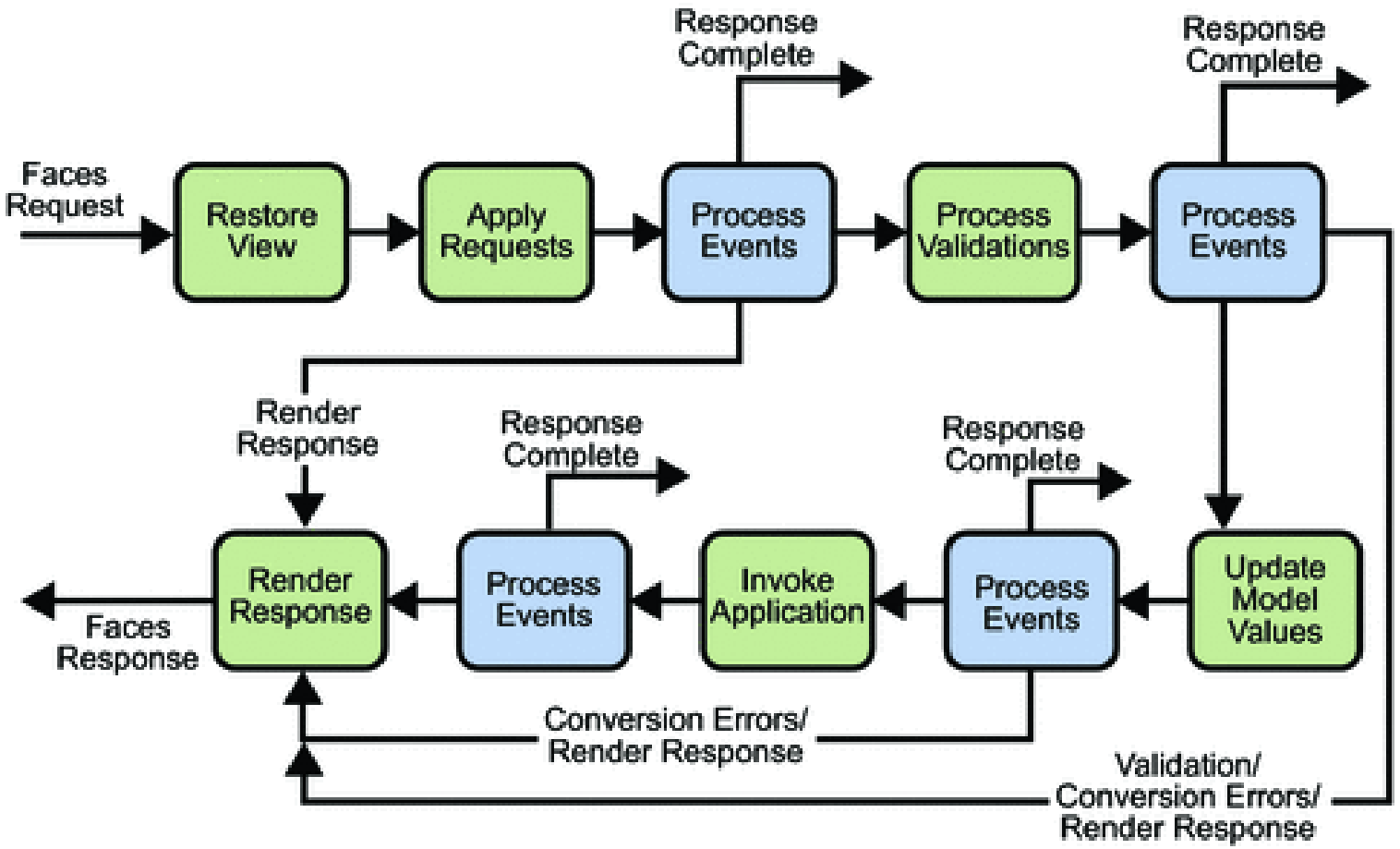
Some examples

- JSF 2.0 hello world example
  - http://www.mkyong.com/jsf2/jsf-2-0-hello-world-example/
- The Obligatory Calculator in JSF 2.0
  - http://buttso.blogspot.pt/2010/03/obligatory-calculator-in-jsf-20.html
- Tutorial web development (with JSF) III: Basic arithmetics
  - http://goo.gl/jFYSyT
- React.js and Spring Data REST
  - https://spring.io/guides/tutorials/react-and-spring-data-rest/
-

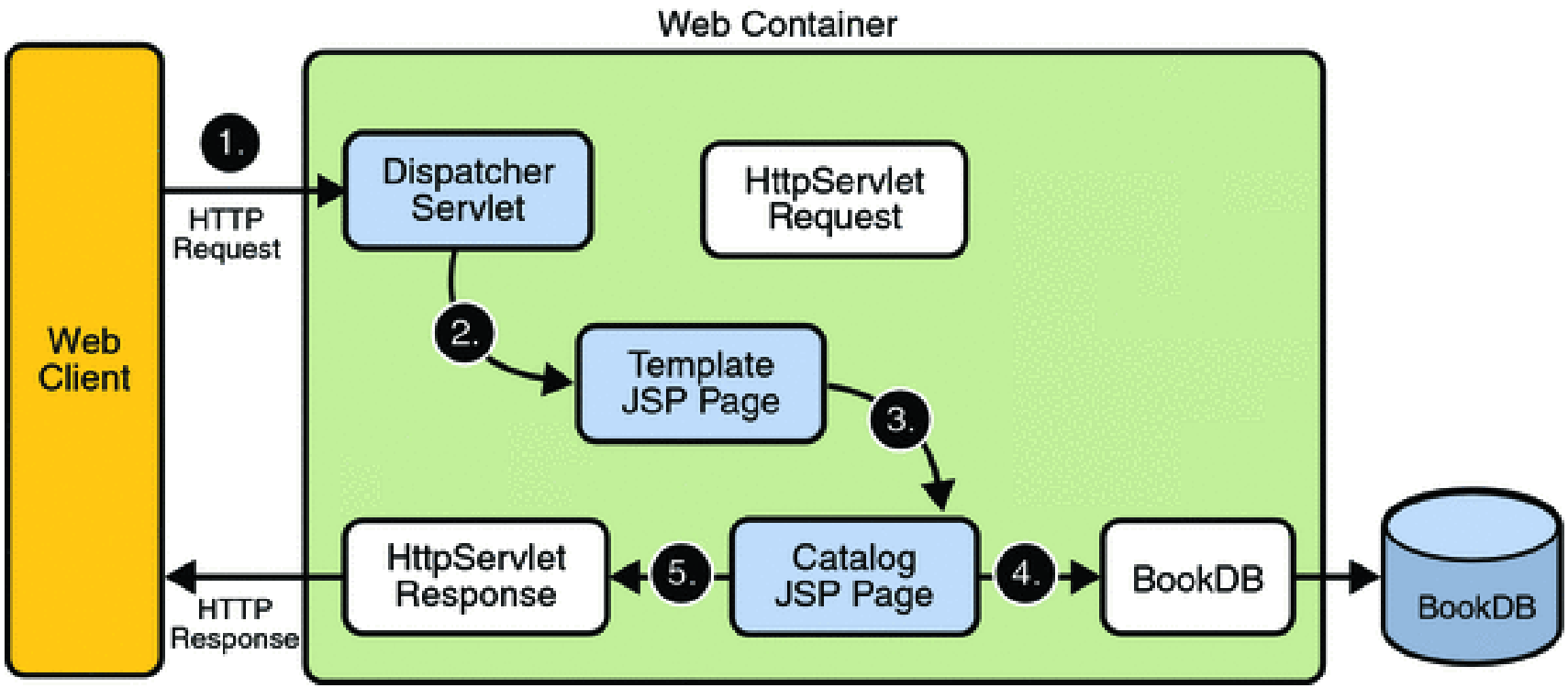
The END

React

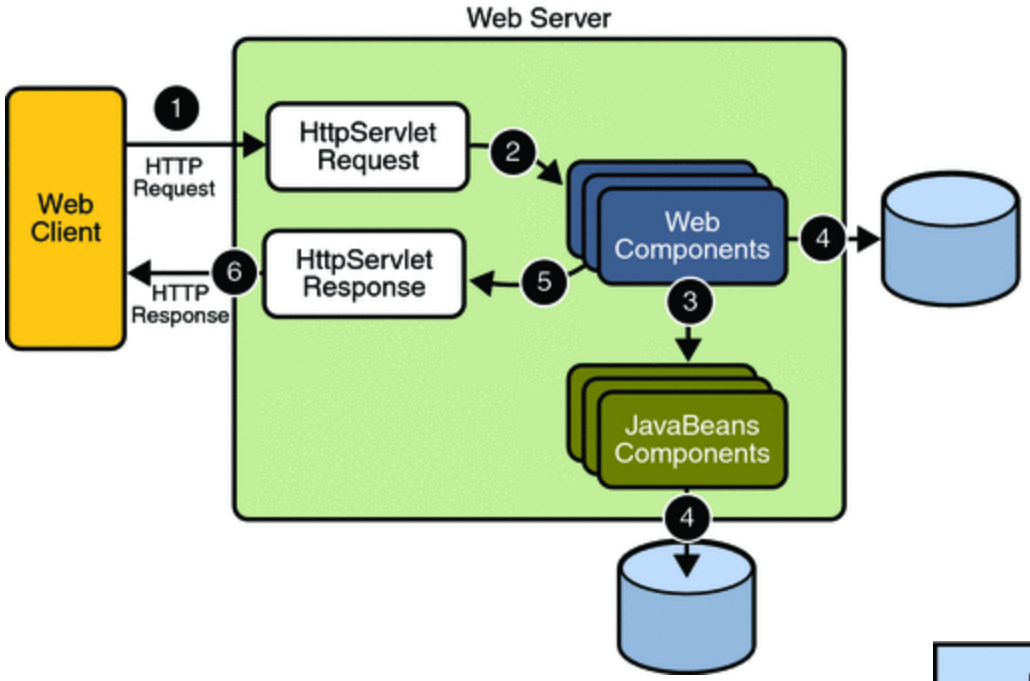
The JSF lifecycle: an overview

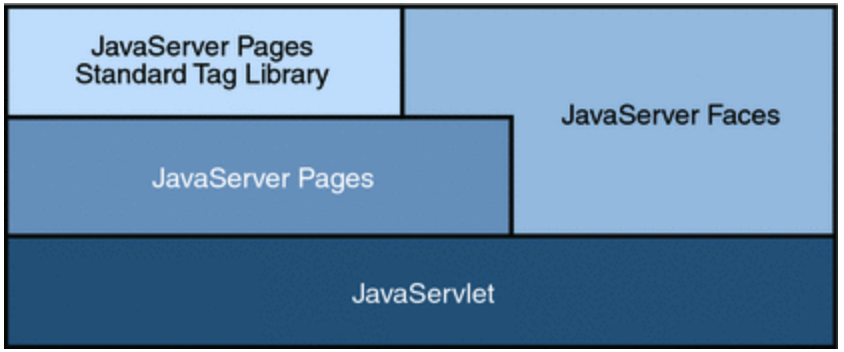


Java Server Pages



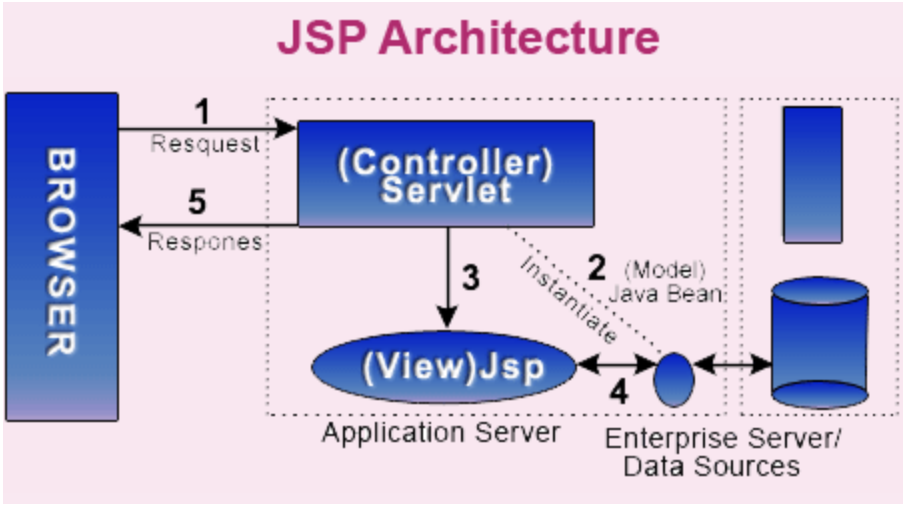
https://docs.oracle.com/javaee/5/tutorial/doc/bnall.html





https://docs.oracle.com/cd/E19575-01/819-3669/bnadr/index.html

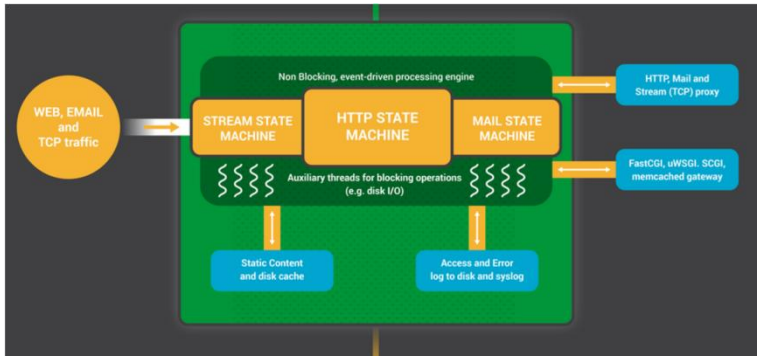
JSP Page Architecture



https://www.roseindia.net/jsp/jsp-page-architecture-and-its-life-cycle.shtml

Event Driven examples

Ngynx: event driven server



- *master* process
  - performs the privileged operations such as reading configuration and binding to ports, and then creates a small number of child processes (the next three types).
- *cache loader*
  - process runs at startup to load the disk-based cache into memory, and then exits. It is scheduled conservatively, so its resource demands are low.
- *cache manager*
  - process runs periodically and prunes entries from the disk caches to keep them within the configured sizes.
- *worker* processes
  - do all of the work! They handle network connections, read and write content to disk, and communicate with upstream servers.

Inside NGINX: How We Designed for Performance & Scale
https://dzone.com/articles/inside-nginx-how-we-designed

Ngynx vs apache

apache

- multi process/multi threaded architecture,
- a request is being served, either a thread or a process is created
  - Block / waits when data is needed from the database,and files from disk, etc

ngynx

- an event driven single threaded architecture
- All requests are handled by a single thread.
  - thread pops up whenever a new connection, or some thing is required(not wasting resources.).
- a very small number of nginx worker process can serve a very very large number of requests.

how is nginx different from apache
https://www.slashroot.in/how-is-nginx-different-from-apache

Criterion	Apache	NGINX
Static speed	Second to NGINX	2.5x faster than Apache
Dynamic speed	Both score same in this area	
OS support	Unix, Windows and MacOSX	Works great with Unix-like OS. Not so with windows.
Security	Both have excellent security track record	
Flexibility	Highly customizable architecture	Difficult to customize modules suitable for the server due to complex base architecture.
Support	Excellent community with widespread user base. Lots of support provided online.	Provides community support through mailing lists, IRC, stack overflow and forum.
Cost	Open source hence free to download and use.	Open source license available along with paid license for advanced features like NGINX plus.

Apache Vs NGINX – A brief comparison
https://www.e2enetworks.com/apache-vs-nginx-brief-comparison/

And example


STOP