22.04.2020
JSF, MVC

Намиот Д.Е.
dnamiot@gmail.com

JSF

- JavaServer Faces (JSF) - MVC web framework

- Упрощение разработки пользовательских интерфейсов в server-side приложениях

- Стандартизация подхода к разработке веб-приложений

- Переиспользование кода при разработке веб-приложений

- Наборы стандартных компонент

JSF

- Веб-страница с интерактивными формами
- Формы – из стандарта HTML (W3C)
- Значения из форм передаются server-side приложению - это HTTP запрос
- Из HTTP запроса извлекаются данные
- Данные обрабатываются в соответствии с какой-то бизнес логикой
- Формируется отклик

- Техническая сторона реализации – это сервлеты (JSP)
- JSF – задание “стандартной’ архитектуры процесса
- Фреймворк – это стандартизация процесса
- Ключевой момент – переиспользование (кода, знаний разработчиков)

JSF

• Веб-страница содержит какие-то формы
• Эти интерактивные элементы, очевидно, переиспользуются приложениями
• Во всех приложениях интерактивные элементы на веб-странице как-то связаны с серверным кодом (данными)
• Например:
 - какие-то поля ввода являются обязательными и пользователю нужно сообщать об ошибке при отсутствии данных
 - где-то нужно отображать в поле ввода значение по умолчанию, которое соответствует полу объекта (значению на стороне сервера)
-введенные значения должны инициализировать некоторый JavaBean и т.д.

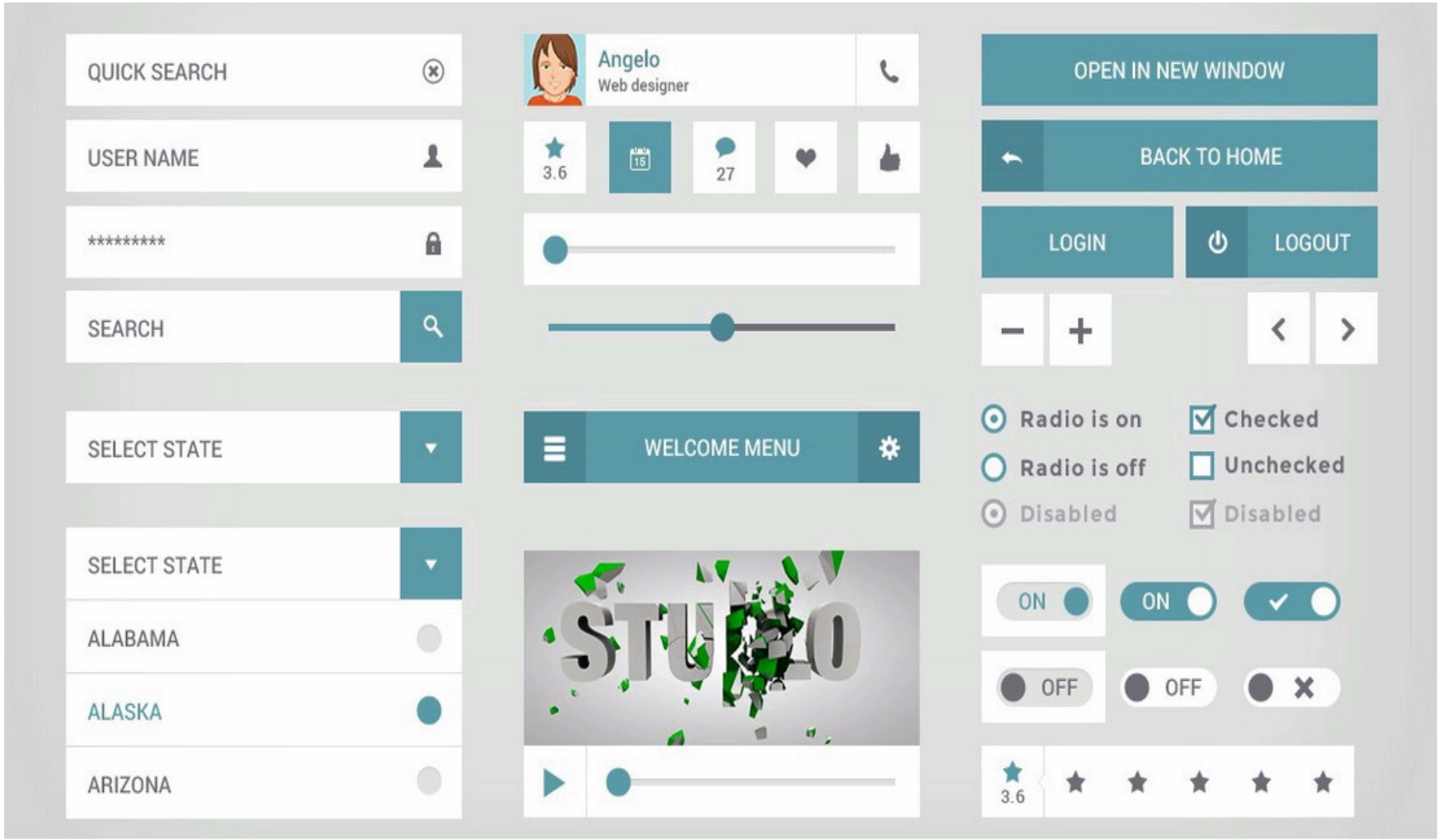
• Это то, что и стандартизует фреймворк JSF

JSF

• JSF – идея сократить время на разработку и сопровождение веб-приложений, предложив механизм (pipe) для взаимодействия клиентских компонент (UI) и серверного приложения. Основные предложения JSF для веб-разработки:

- поддержка переиспользуемых UI компонент
- стандартизация передачи данных между UI компонентами
- поддержка состояния UI между запросами
- создание собственных компонент
- обработка клиентский событий на сервере

UI components



JSF UI model

Создание веб-приложений из коллекций UI компонент.
Упор делался на набор компонент, их корректную отрисовку (rendering) на разных устройствах.

В состав JSF входят:

Core library

A set of base UI components - standard HTML input elements

Extension of the base UI components to create additional UI component libraries or to extend existing components

Multiple rendering capabilities that enable JSF UI components to render themselves differently depending on the client types

JSF MVC

Веб-фреймворк для работы с UI компонентами и испоьзованием их в веб-приложениях. Построен с использованием шаблона Model View Controller (MVC).

Это шаблон проектирования, который разбивает приложения на 3 логических компоненты:

• Model – данные, авторизация пользователей

• View – пользовательский интерфейс

• Controller – бизнес-логика

Цели:

• Явно выделить презентационный уровень (дизайн).

• Обеспечить возможность разрабатывать уровни параллельно

• Переиспользование результатов

• Независимое изменение уровней

JSF архитектура

JSF технически построен на Java-веб технологиях, все работает в контейнере, основан на сервлетах.

Что внутри:

JavaBeans components as models containing application-specific functionality and data

A custom tag library for representing event handlers and validators

A custom tag library for rendering UI components

UI components represented as stateful objects on the server

Server-side helper classes

Validators, event handlers, and navigation handlers

Application configuration resource file for configuring application resources

JSF summary

JSF основан на шаблоне Model-View-Controller

JSF поддерживает модель UI компонент с поддержкой состояний. Каждая компонента имеет привязанные к ней данные

JSF разделяет функционал компонент от их отображения. Renderer – отображение компонент на клиенте. Его можно менять. Стандартная реализация - HTML renderer.

JSF поддерживает обработку событий в UI компонентах

JSF поддерживает: проверку данных, привязку данных к компонентам (data binding), преобразования данных при передаче между UI и моделью

JSF servlet

```
<servlet>
      <servlet-name>Faces Servlet</servlet-name>
      <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>/faces/*</url-pattern>
   </servlet-mapping>

   <servlet-mapping>
      <servlet-name>Faces Servlet</servlet-name>
      <url-pattern>*.jsf</url-pattern>
   </servlet-mapping>
```

JSF example

```
<html>
   <head>
      <title>JSF Tutorial!</title>
   </head>

   <body>
      #{helloWorld.getMessage()}
   </body>
</html>
```

JSF managed bean

```
import javax.faces.bean.ManagedBean;

@ManagedBean(name = "helloWorld", eager = true)
@ApplicationScoped
public class HelloWorld {

   public HelloWorld() {
      System.out.println("HelloWorld started!");
   }

   public String getMessage() {
      return "Hello World!";
   }
}
```

Managed beans annotations

Application (@ApplicationScoped): Application scope persists across all users' interactions with a web application.

Session (@SessionScoped): Session scope persists across multiple HTTP requests in a web application.

View (@ViewScoped): View scope persists during a user's interaction with a single page (view) of a web application.

Request (@RequestScoped): Request scope persists during a single HTTP request in a web application.

None (@NoneScoped): Indicates a scope is not defined for the application.

Controls and beans

<input type=“text” … >

||

V

h:inputText – custom tag в JSF

```
<h:inputText id="quantity"
             size="4"
             value="#{item.quantity}"
             title="#{bundle.ItemQuantity}">
    <f:validateLongRange minimum="1"/>
</h:inputText>
```

JSF controls

h:column
Represents a column of data in a data component
A column of data in an HTML table
A column in a table

h:commandButton
Submits a form to the application
An HTML <input type=type> element, where the type value can be "submit", "reset", or "image"
A button

h:commandLink
Links to another page or location on a page
An HTML <a href> element
A hyperlink

h:dataTable
Represents a data wrapper
An HTML <table> element
A table that can be updated dynamically

JSF controls

h:form
Represents an input form (inner tags of the form receive the data that will be submitted with the form)
An HTML <form> element
No appearance

h:graphicImage
Displays an image
An HTML <img> element
An image

h:inputHidden
Allows a page author to include a hidden variable in a page
An HTML <input type="hidden"> element
No appearance

h:inputSecret
Allows a user to input a string without the actual string appearing in the field
An HTML <input type="password"> element
A text field, which displays a row of characters instead of the actual string entered

JSF controls

h:inputText
Allows a user to input a string
An HTML <input type="text"> element
A text field

h:inputTextarea
Allows a user to enter a multiline string
An HTML <textarea> element
A multi-row text field

h:message
Displays a localized message
An HTML tag if styles are used
A text string

h:messages
Displays localized messages
A set of HTML tags if styles are used
A text string

JSF controls

h:outputFormat
Displays a localized message
Plain text
Plain text

h:outputLabel
Displays a nested component as a label for a specified input field
An HTML <label> element
Plain text

h:outputLink
Links to another page or location on a page without generating an action event
An HTML <a> element
A hyperlink

h:outputText
Displays a line of text
Plain text
Plain text

JSF: общие атрибуты

binding
Identifies a bean property and binds the component instance to it.

id
Uniquely identifies the component.

immediate
If set to true, indicates that any events, validation, and conversion associated with the component should happen when request parameter values are applied,

rendered
Specifies a condition under which the component should be rendered. If the condition is not satisfied, the component is not rendered.

style
Specifies a Cascading Style Sheet (CSS) style for the tag.

styleClass
Specifies a CSS class that contains definitions of the styles.

value
Specifies the value of the component, in the form of a value expression.

JSF примеры

```
<h:inputText id="name"
             label="Customer Name"
             size="30"
             value="#{cashier.name}"
             required="true"

    requiredMessage="#{bundle.ReqCustomerName}">
    <f:valueChangeListener
        type="dukesbookstore.listeners.NameChanged" />
</h:inputText>
```

JSF: core tags

f:actionListener
Adds an action listener to a parent component

f:phaseListener
Adds a PhaseListener to a page

f:setPropertyActionListener
Registers a special action listener whose sole purpose is to push a value into a managed bean when a form is submitted

f:valueChangeListener
Adds a value-change listener to a parent component

JSF core tags

f:converter

Adds an arbitrary converter to the parent component

f:convertDateTime

Adds a DateTimeConverter instance to the parent component

f:convertNumber

Adds a NumberConverter instance to the parent component

JSF core tags

```
<h:outputText value="#{cashier.shipDate}">
    <f:convertDateTime type="date" dateStyle="full" />
</h:outputText>

<h:outputText value="#{cashier.shipDate}">
    <f:convertDateTime
        pattern="EEEEEEEE, MMM dd, yyyy" />
</h:outputText>

<h:outputText value="#{cart.total}" >
    <f:convertNumber currencySymbol="$"
    type="currency"/>
</h:outputText>
```

JSF core tags

f:validateDoubleRange
Adds a DoubleRangeValidator to a component
f:validateLength

Adds a LengthValidator to a component
f:validateLongRange
Adds a LongRangeValidator to a component

f:validator
Adds a custom validator to a component
f:validateRegEx

Adds a RegExValidator to a component
f:validateBean
Delegates the validation of a local value to a BeanValidator

f:validateRequired
Enforces the presence of a value in a component

JSF core tags

Attribute configuration
f:attribute
Adds configurable attributes to a parent component

Localization
f:loadBundle
Specifies a ResourceBundle that is exposed as a Map

Parameter substitution
f:param
Substitutes parameters into a MessageFormat instance and adds query string name-value pairs to a URL

Ajax
f:ajax
Associates an Ajax action with a single component or a group of components based on placement

Event
f:event
Allows installing a ComponentSystemEventListener on a component

JSF: listeners

valueChangeListener
actionListener

```
<h:inputText id="name"
             size="30">
    <f:valueChangeListener
        type=“полное имя класса" />
</h:inputText>

<h:commandLink id="Duke" action="bookstore">
    <f:actionListener
        type=“полное имя класса" />
    <h:outputText value=“Link"/>
</h:commandLink>
```

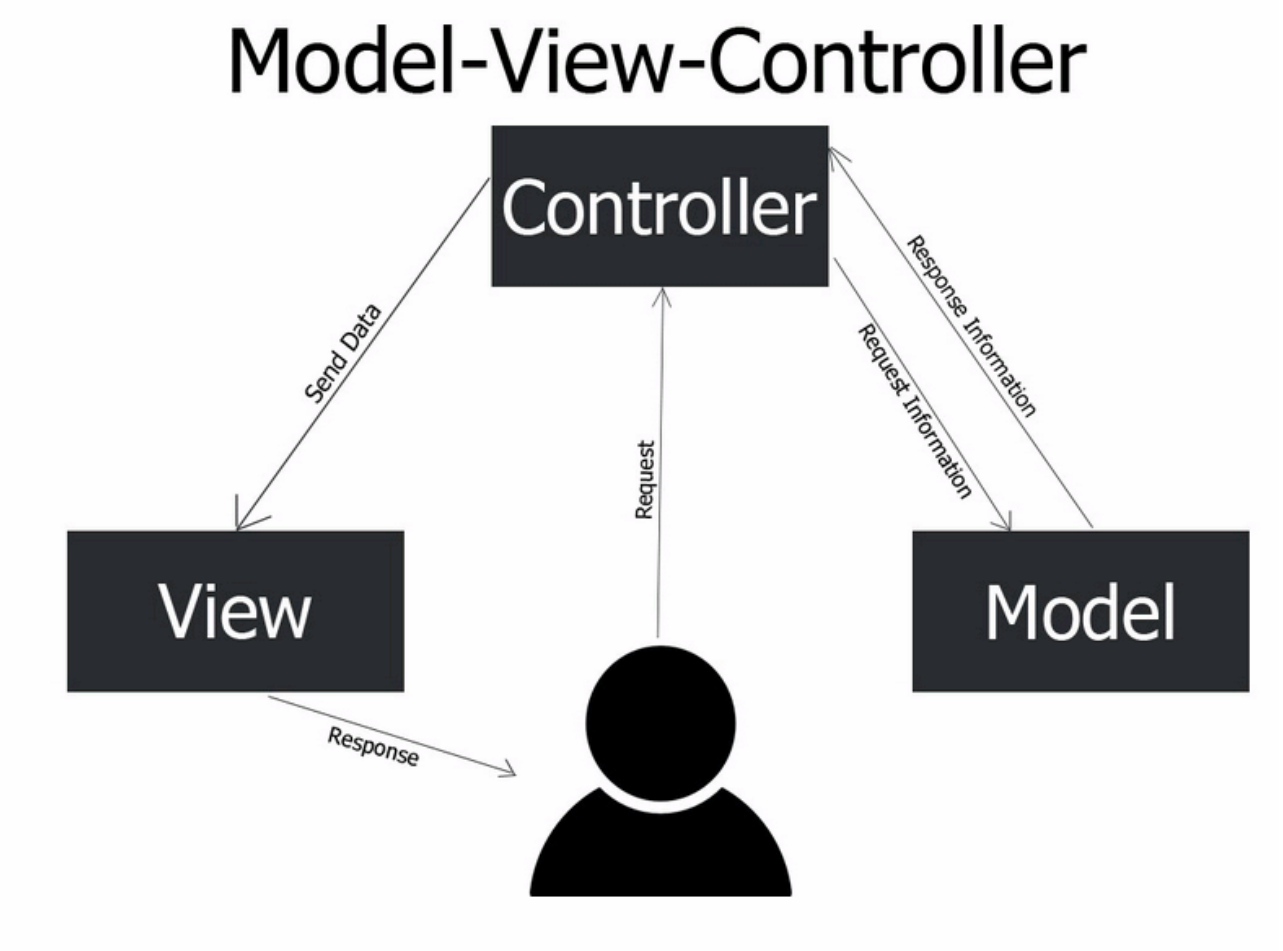
JSF: listener

```
public class NameChanged implements ValueChangeListener {

    @Override
    public void processValueChange(ValueChangeEvent event)
        throws AbortProcessingException {

        if (null != event.getNewValue()) {
            FacesContext.getCurrentInstance().getExternalContext().
                getSessionMap().put("name", event.getNewValue());
        }
    }
}
```

JSF validator

```
<h:inputText id="quantity" size="4"
      value="#{item.quantity}" >
    <f:validateLongRange minimum="1"/>
</h:inputText>
```

MVC


Model-View-Controller
Controller
Send Data
Request
Request Information
Response Information
View
Model
Response

MVC Web приложение

- HTTP запрос приходит от пользователя
- Запрос принимает контроллер
- Для обработки запроса нужны какие-то данные – это модель
- Обработав запрос необходимо сформировать отклик (HTTP отклик)
- Оклик предъявляется пользователю (view)

MVC Web приложение

- <form method=“post” action=“/a1”>
  </form>
- <form method=“post” action=“/a2”>
  </form>
- Next

  Для каждого типа запроса нужно писать свою реакцию.

MVC Web приложение

- <form method=“post” action=“C?a1”>
  </form>
- <form method=“post” action=“C?a2”>
  </form>
- Next
  Одна реакция с разными параметрами.

Это пример реализации контроллера

MVC Web приложение

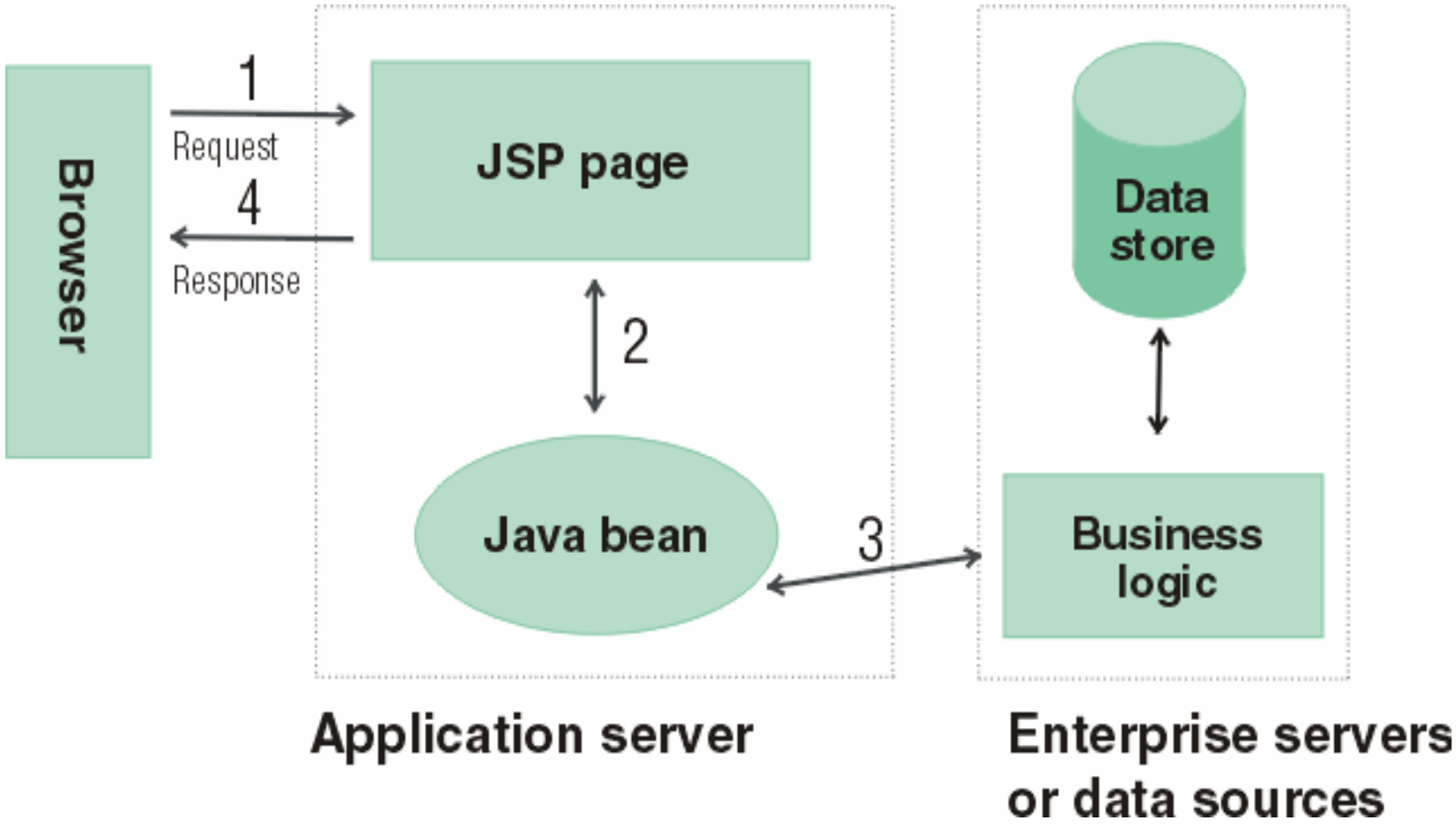
Контроллер в таком исполнении есть компонента, которая содержит бизнес-логику. Это сервлет

Модель – единый интерфейс к данным

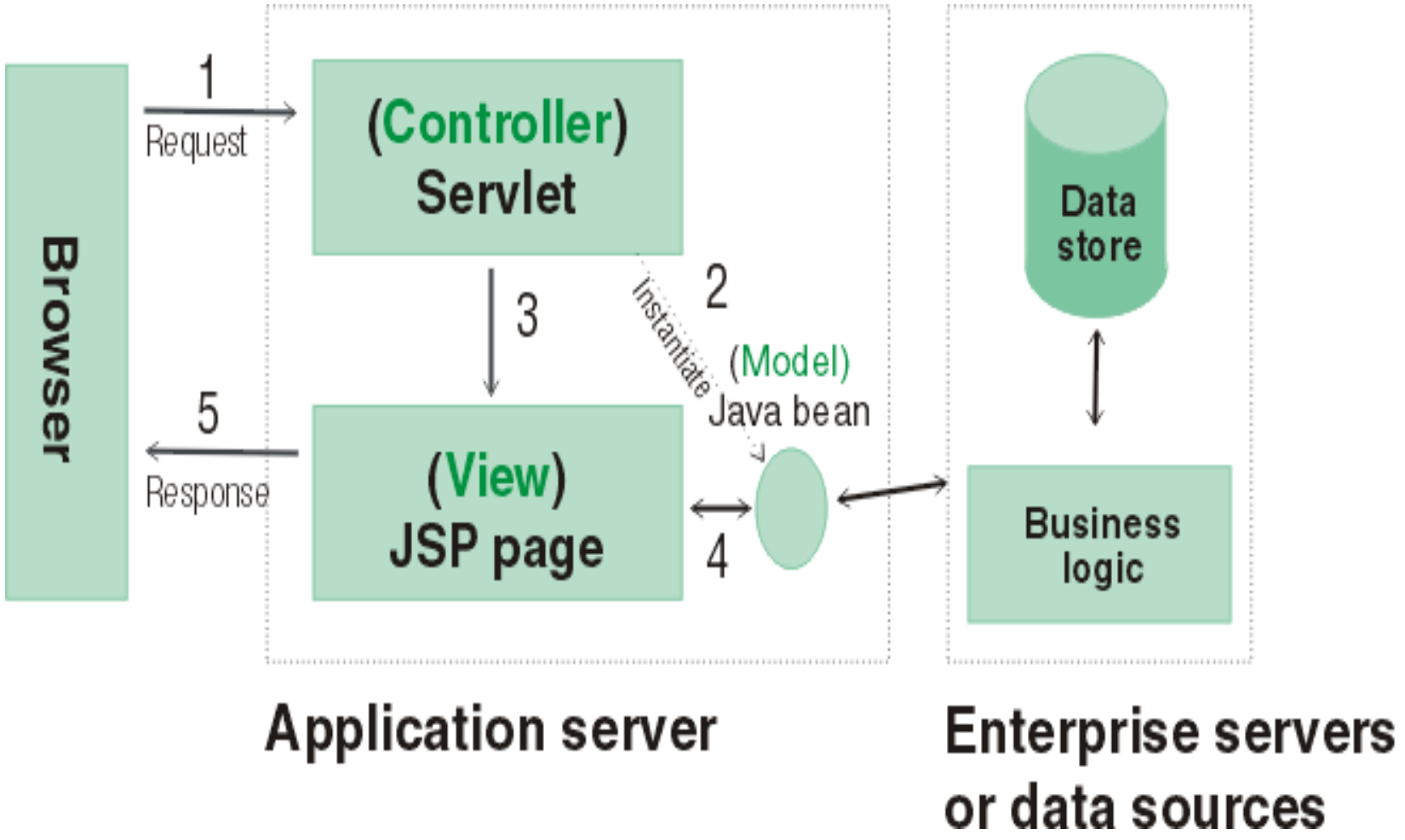
Представление – контроллер, обработав запрос, переправляет его на страницу, которая будет показана пользователю

Вот эта страница и есть представление

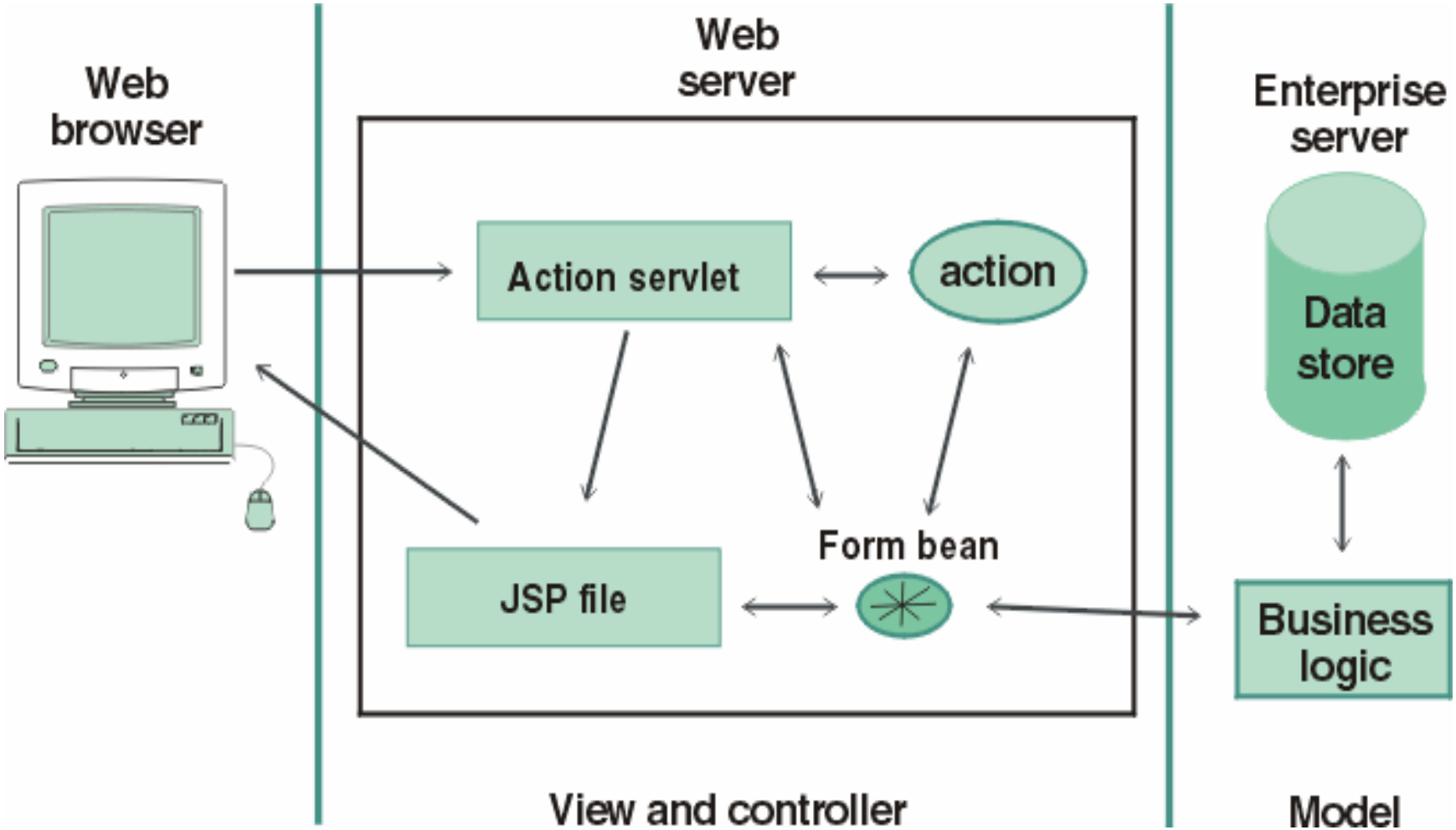
Model 1


Browser
1
Request
4
Response
JSP page
2
Java bean
3
Data store
Business logic
Application server
Enterprise servers or data sources

Model 2


Browser
1
Request
(Controller)
Servlet
2
Instantiate
(Model)
Java bean
3
5
Response
(View)
JSP page
4
Data
store
Business
logic
Application server
Enterprise servers
or data sources

Struts


Web browser
Web server
Enterprise server
Action servlet
action
Data store
Form bean
JSP file
Business logic
View and controller
Model

Struts model

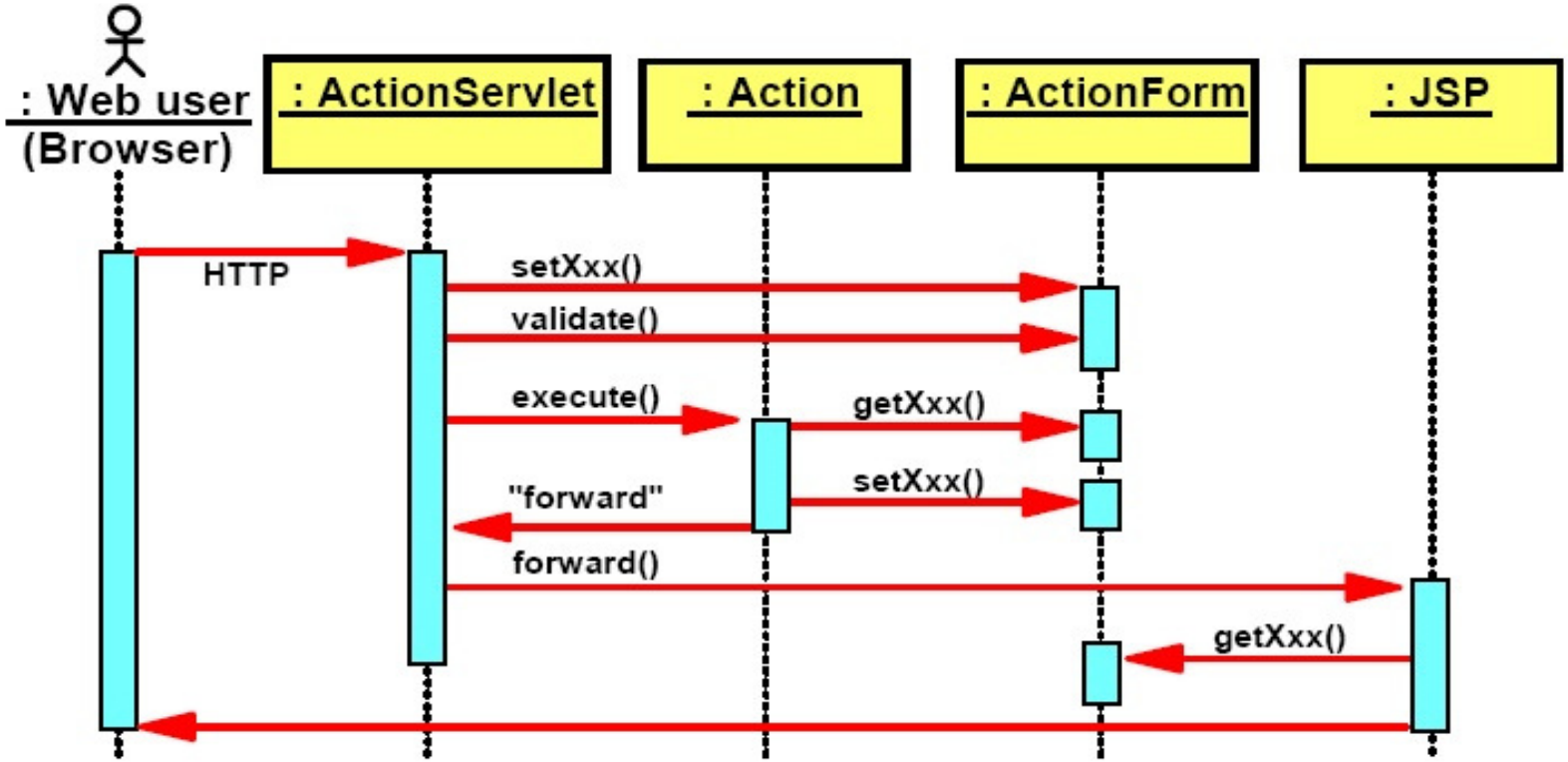


Figure 12-48 Struts request sequence

Struts: example

index.jsp

```
<!DOCTYPE html>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Basic Struts 2 Application </title>
   </head>
   <body>
      <h1>Welcome To Struts 2!</h1>
      <p><a href="<s:url action='hello'/>">Hello World</a></p>
   </body>
</html>
```

Struts: конфигурация

struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
        "http://struts.apache.org/dtds/struts-2.5.dtd">
<struts>
    <constant name="struts.devMode" value="true" />

    <package name="basicstruts2" extends="struts-default">
        <action name="index">
            <result>/index.jsp</result>
        </action>

        <action name="hello" class="org.apache.struts.helloworld.action.HelloWorldAction"
      method="execute">
            <result name="success">/HelloWorld.jsp</result>
        </action>
    </package>
</struts>
```

Struts: example

HelloWorld.jsp

```
<!DOCTYPE html>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>   <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Hello World!</title>
  </head>
  <body>
    <h2><s:property value="messageStore.message" /></h2>
  </body>
</html>
```

Struts: beans

MessageStore.java

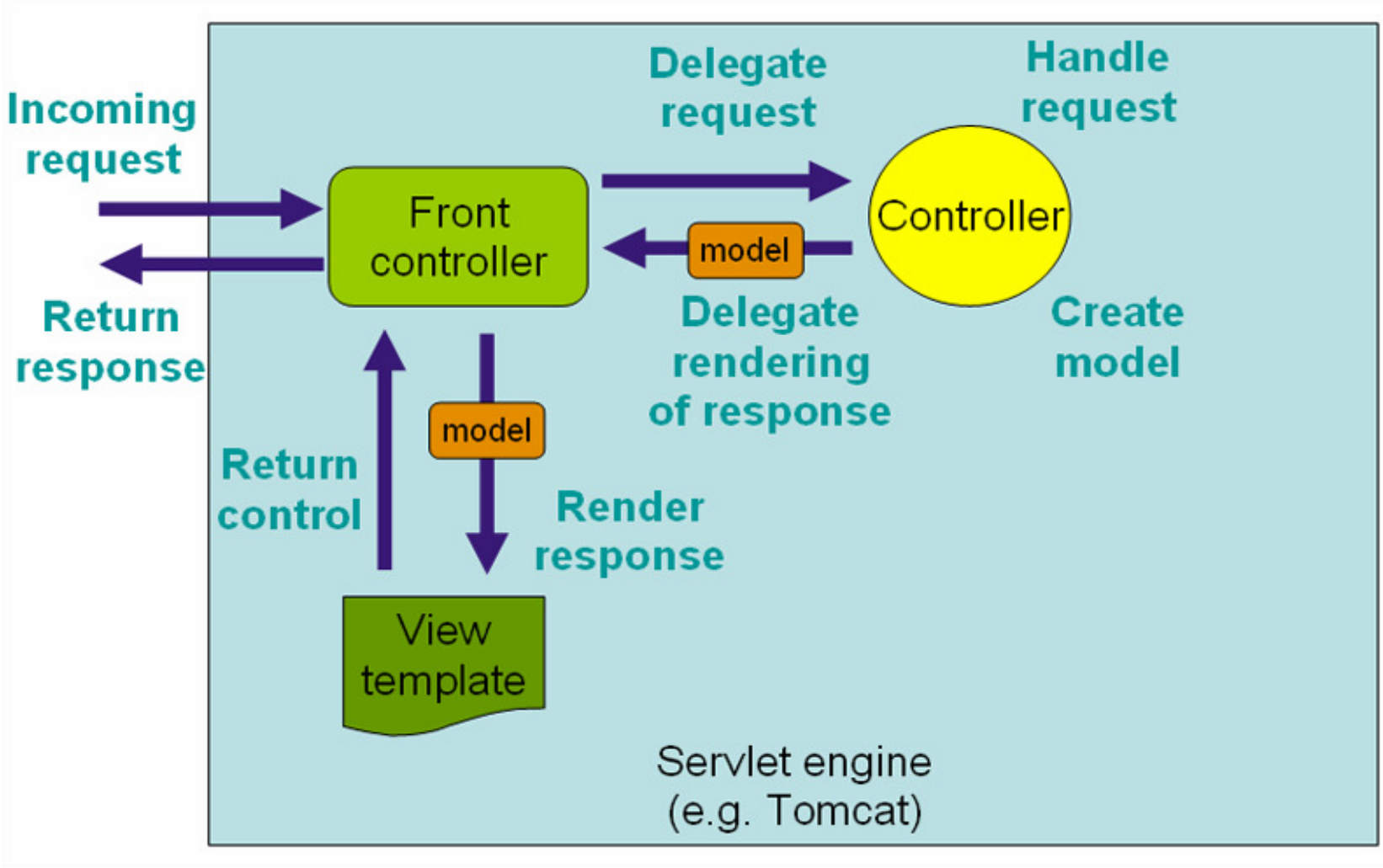
```
package org.apache.struts.helloworld.model;

public class MessageStore {
    private String message;

    public MessageStore() {
        message = "Hello Struts User";
    }

    public String getMessage() {
        return message;
    }
}
```

Struts: action

HelloWorldAction.java

```
package org.apache.struts.helloworld.action;

import org.apache.struts.helloworld.model.MessageStore;

import com.opensymphony.xwork2.ActionSupport;

public class HelloWorldAction extends ActionSupport {
    private MessageStore messageStore;

    public String execute() {
        messageStore = new MessageStore() ;

        return SUCCESS;
    }

    public MessageStore getMessageStore() {
        return messageStore;
    }
}
```

Spring MVC


Incoming request
Return response
Front controller
Delegate request
Handle request
Controller
model
Delegate rendering of response
Create model
model
Return control
Render response
View template
Servlet engine (e.g. Tomcat)