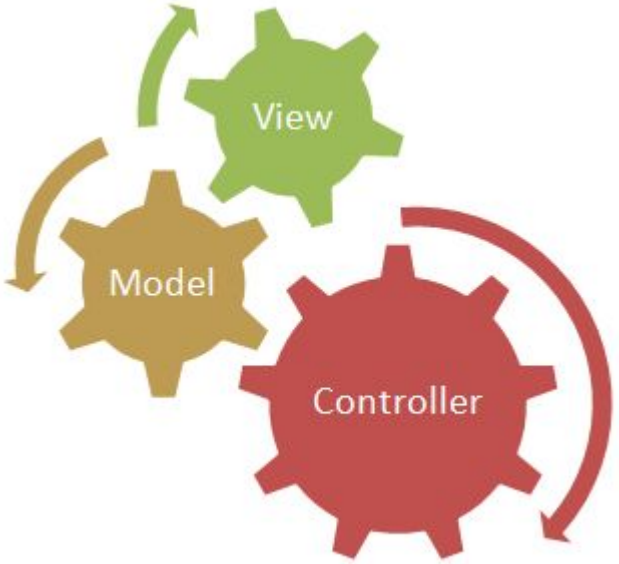
# Backend:

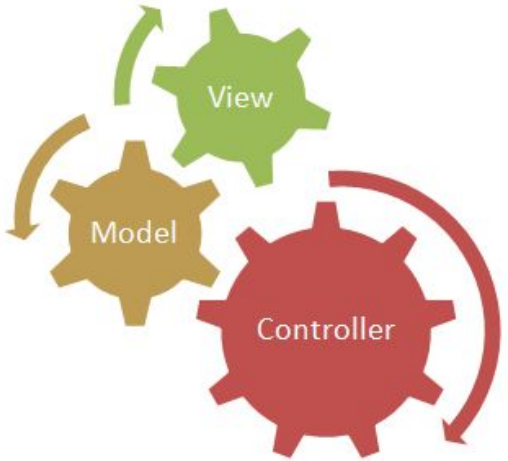

**Java D3 Mentoring Program BY/RU/KZ**

April, 2017

**Dzianis Kashtsialian**
Lead Software Engineer
9+ years experience on <epam>
JMP Curator & Mentor

# AGENDA

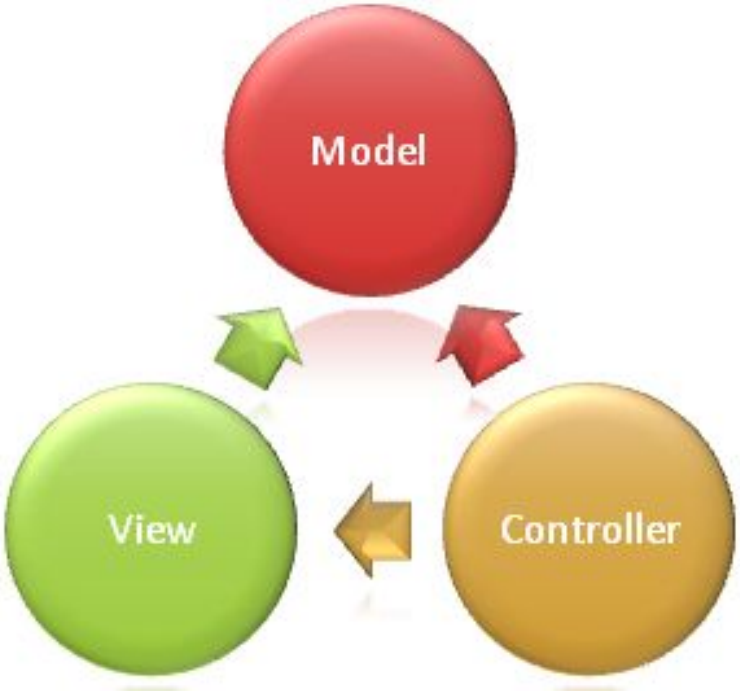
1. MVC and Spring MVC
2. DispatcherServlet / Controller / View / Model
3. Internationalization(i18n) / Themes / Templates (Apache Tiles)
4. Mapping of URLs, Validation support (JSR-349)
5. File Upload Handling
6. Supporting Servlet 3.0 Code-Based (Java-Based) Configuration
7. Handling Exceptions
8. Spring MVC and Spring security
9. Spring MVC Workshop

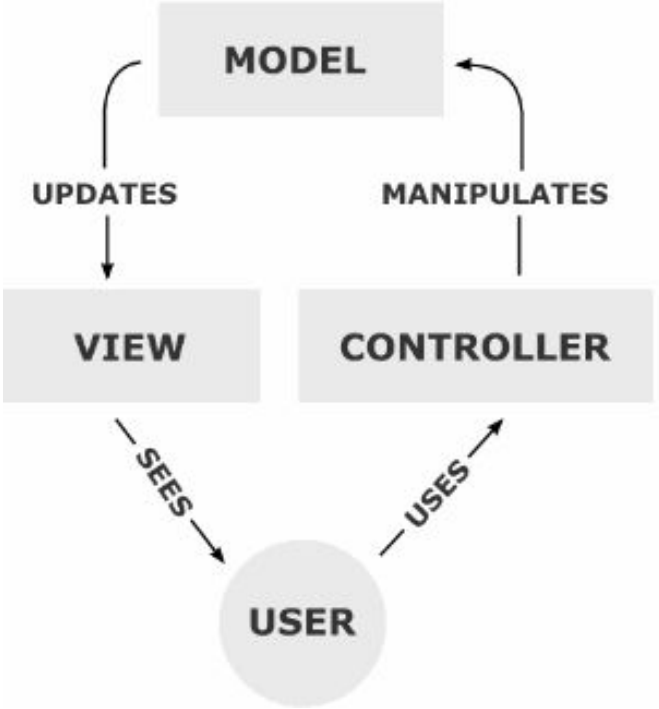


# INTRODUCING MVC AND SPRING MVC

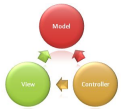
# Introducing MVC and Spring MVC

## What is MVC?



# Introduction to Web Spring MVC




## Main features

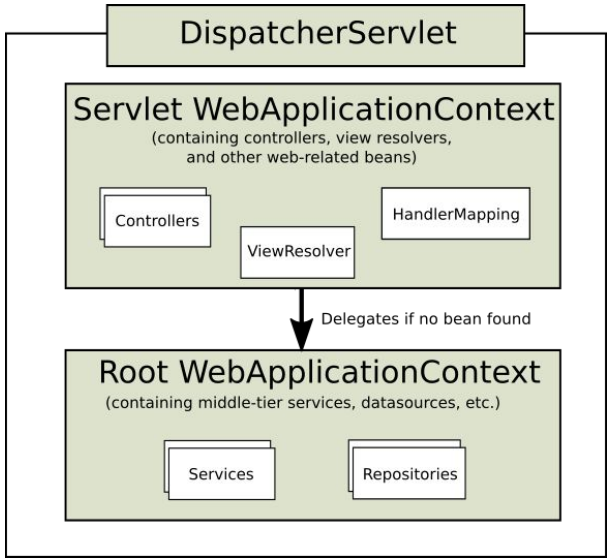
- Spring MVC provides a clear separation between a model, view and controller
- Provides both XML-based and annotation-based approaches.
- Enriched by Spring application context.
- Provides a broad range of pre-configured facilities.
- Takes convention over configuration approach.
- Uses open-close principle.

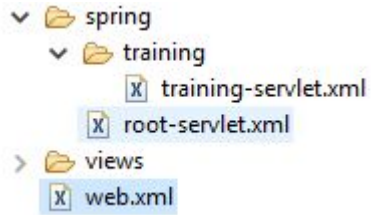
## Benefits

- Decoupling views and models
- Reduces the complexity of your design
- Makes code more flexible
- Makes code more maintainable

# Introducing MVC and Spring MVC

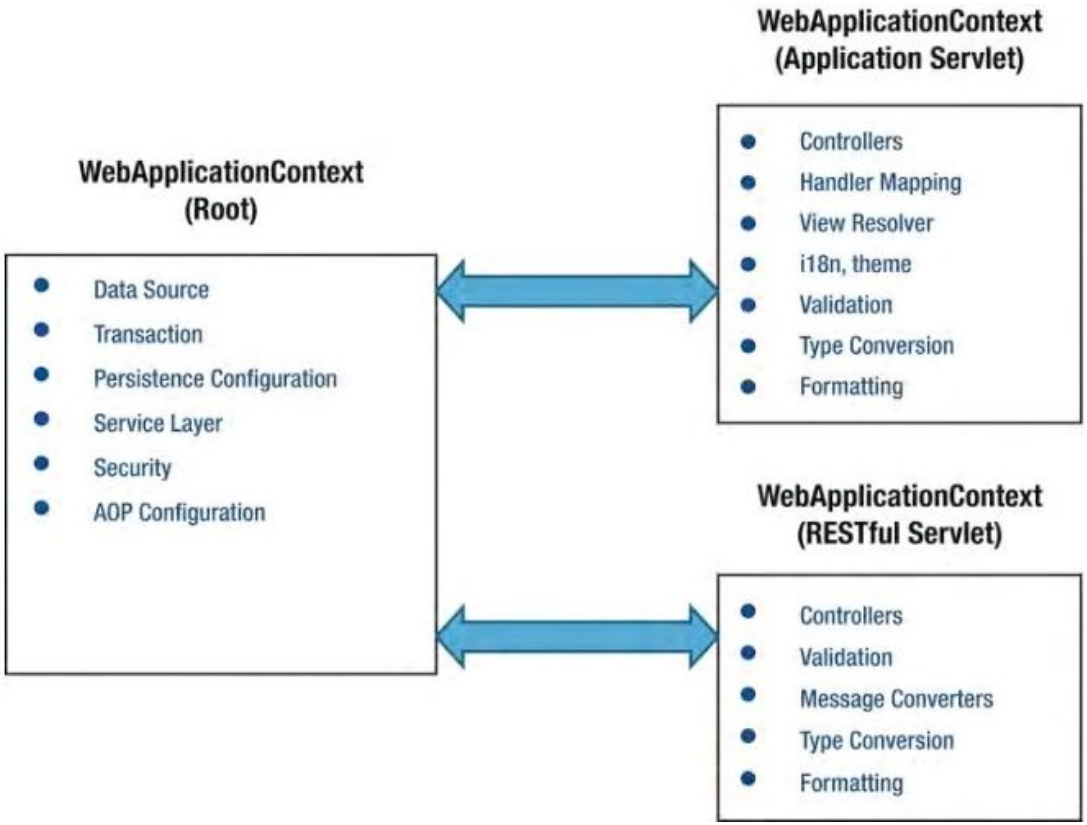
## Spring MVC WebApplicationContext Hierarchy



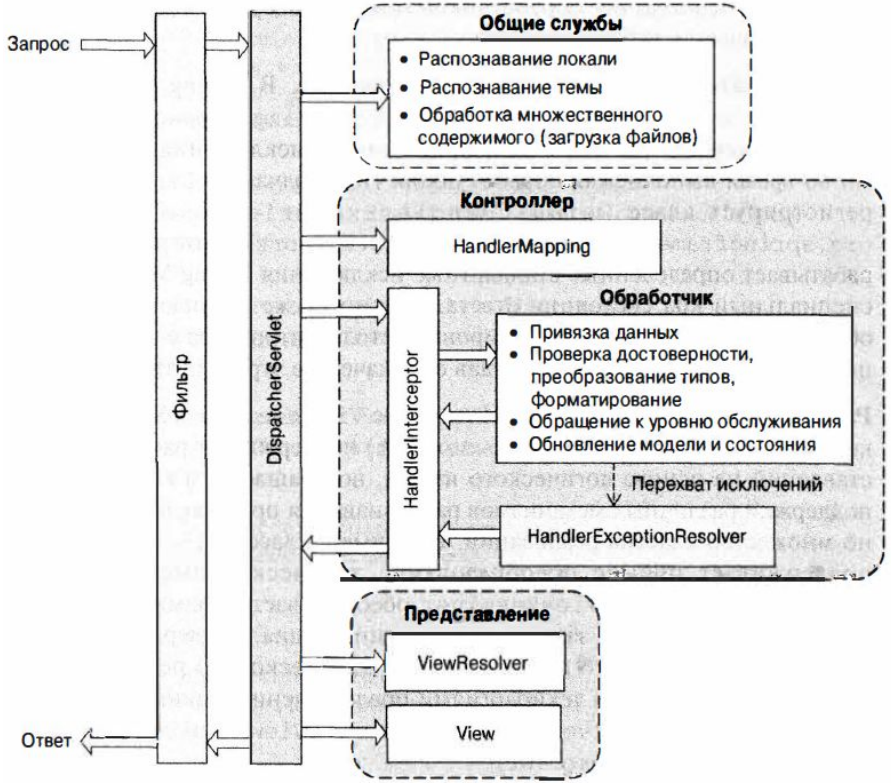




```
<!-- The definition of the Root Spring Container shared by all Servlets
    and Filters -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-servlet.xml</param-value>
</context-param>


<!-- Processes application requests -->
<servlet>
    <servlet-name>training</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/training/training-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
    <multipart-config>
        <max-file-size>5000000</max-file-size>
    </multipart-config>
</servlet>

<servlet-mapping>
    <servlet-name>training</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

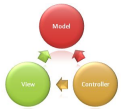
# Spring MVC WebApplicationContext Hierarchy



# Introducing MVC and Spring MVC
## Spring MVC Request Life Cycle



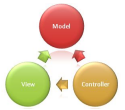
# Introducing MVC and Spring MVC
## Intercepting requests with a HandlerInterceptor




```
<beans>
    <bean id="handlerMapping"
            class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping">
        <property name="interceptors">
            <list>
                <ref bean="officeHoursInterceptor"/>
            </list>
        </property>
    </bean>

    <bean id="officeHoursInterceptor"
            class="samples.TimeBasedAccessInterceptor">
        <property name="openingTime" value="9"/>
        <property name="closingTime" value="18"/>
    </bean>
<beans>
```

```
package samples;

public class TimeBasedAccessInterceptor extends HandlerInterceptorAdapter {

    private int openingTime;
    private int closingTime;

    public void setOpeningTime(int openingTime) {
        this.openingTime = openingTime;
    }

    public void setClosingTime(int closingTime) {
        this.closingTime = closingTime;
    }

    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
            Object handler) throws Exception {
        Calendar cal = Calendar.getInstance();
        int hour = cal.get(HOUR_OF_DAY);
        if (openingTime <= hour && hour < closingTime) {
            return true;
        }
        response.sendRedirect("http://host.com/outsideOfficeHours.html");
        return false;
    }
}
```

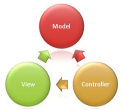
# Introducing MVC and Spring MVC
## Spring MVC Request Life Cycle




```
<!-- Resolves views selected for rendering by @Controllers to .jsp resources
    in the /WEB-INF/views directory -->
<beans:bean class="org.springframework.web.servlet.view.UrlBasedViewResolver"
    id="tilesViewResolver">
    <beans:property name="viewClass"
        value="org.springframework.web.servlet.view.tiles3.TilesView" />
</beans:bean>
```

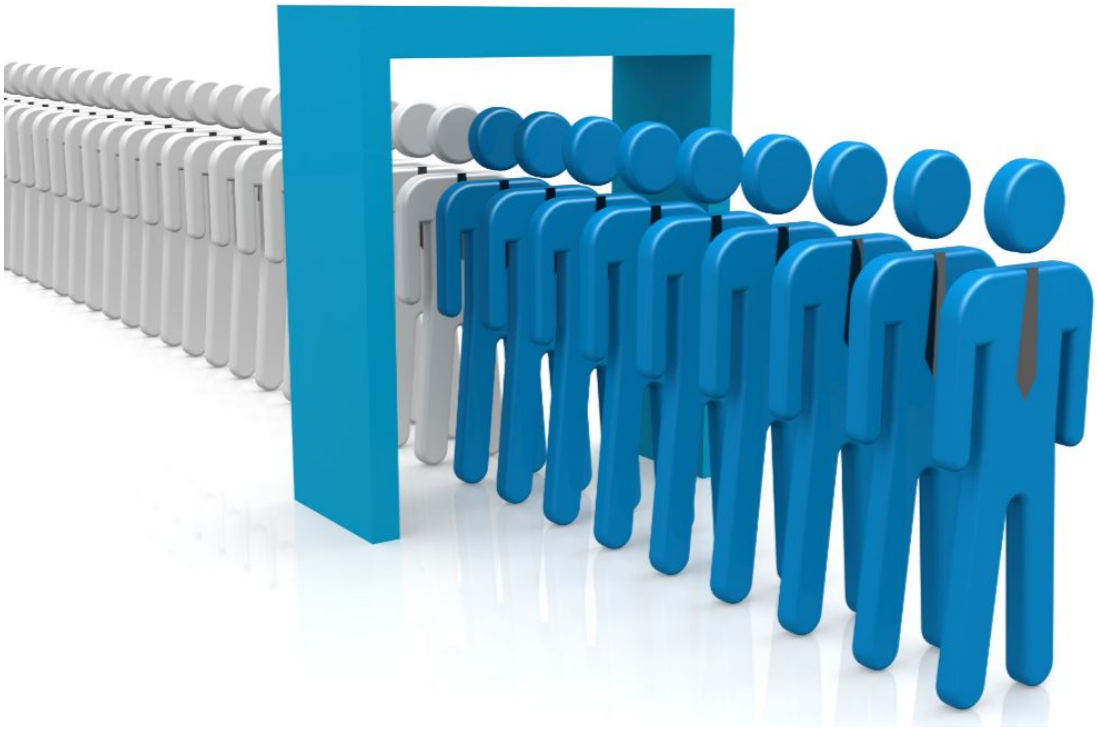
- BeanNameViewResolver
- FreeMarkerViewResolver
- InternalResourceViewResolver
- JasperReportsViewResolver
- ResourceBundleViewResolver
- UrlBasedViewResolver
- VelocityLayoutViewResolver
- VelocityViewResolver
- XmlViewResolver
- XsltViewResolver

# Spring MVC Configuration




To configure Spring MVC support for web applications:

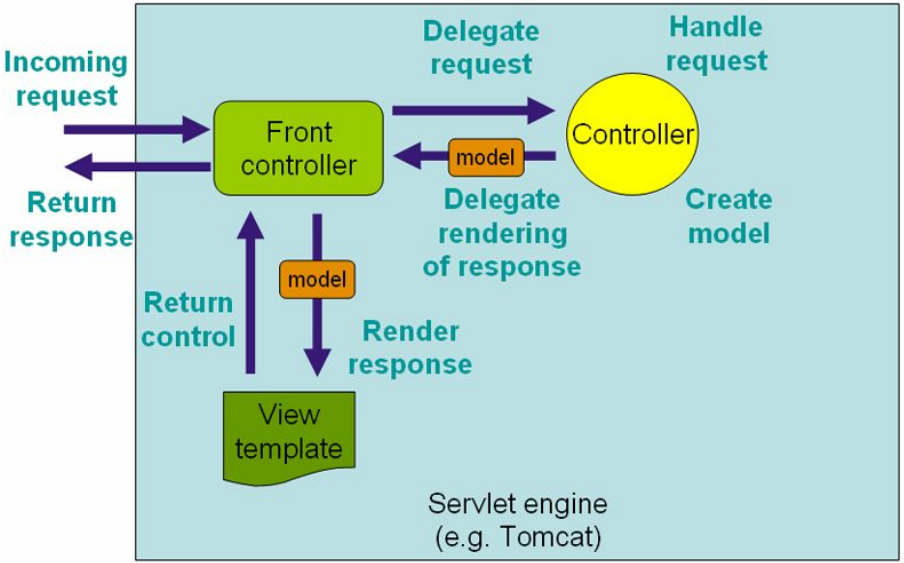
- Configuring the root **WebApplicationContext**
- Configuring the servlet **filters** required by Spring MVC
- Configuring the **dispatcher servlets** within the application

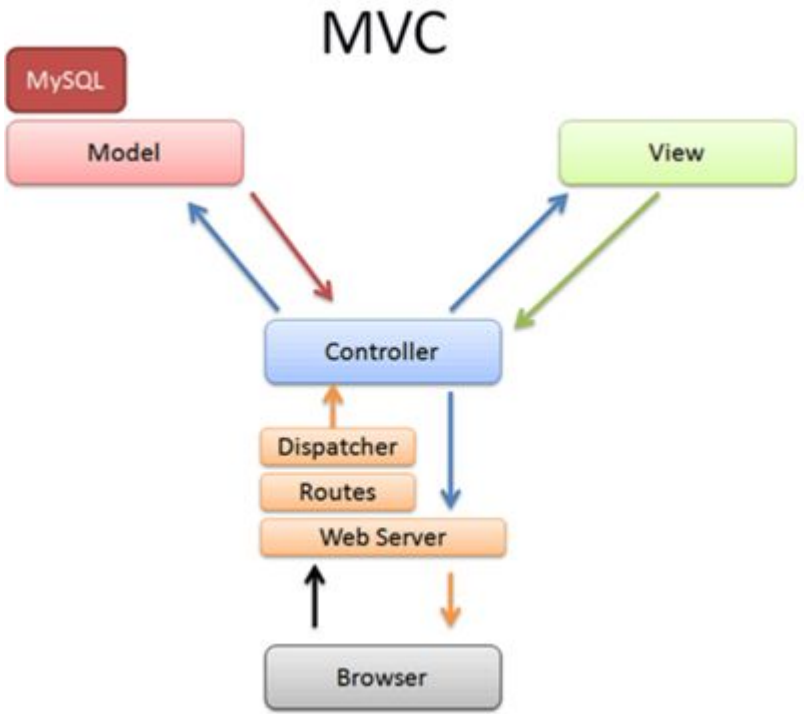


# DISPATCHERSERVLET, CONTROLLER, VIEW, MODEL

# DispatcherServlet, Controller, View, Model

## DispatcherServlet



# DispatcherServlet Code Overview

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0"
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd">
...
</web-app>
```

# DispatcherServlet Code Overview



```
<!-- The definition of the Root Spring Container shared by all Servlets
    and Filters -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-servlet.xml</param-value>
</context-param>
```

# DispatcherServlet Code Overview


```
<filter>
    <filter-name>HttpMethodFilter</filter-name>
    <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>HttpMethodFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

# DispatcherServlet Code Overview

```
<!-- Listener -->

<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

# DispatcherServlet Code Overview

```
<!-- Processes application requests -->
<servlet>
    <servlet-name>training</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/training/training-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
    <multipart-config>
        <max-file-size>5000000</max-file-size>
    </multipart-config>
</servlet>

<servlet-mapping>
    <servlet-name>training</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

# Controllers



- **Controllers** provide access to the application behavior that you typically define through a service interface.
- **Controllers** interpret user input and transform it into a model that is represented to the user by the view.
- Spring implements a **controller** in a very abstract way, which enables you to create a wide variety of controllers.

```
@RequestMapping("/contacts")
@Controller
public class ContactController {
    private ContactService contactService;
    @RequestMapping(method = RequestMethod.GET)
    public String list(Model uiModel) {
        List<Contact> contacts = contactService.findAll();
        uiModel.addAttribute("contacts", contacts);
        return "contacts/list";
    }
    @Autowired
    public void setContactService(ContactService contactService) {
    this.contactService = contactService;
    }
}
```

# View Examples



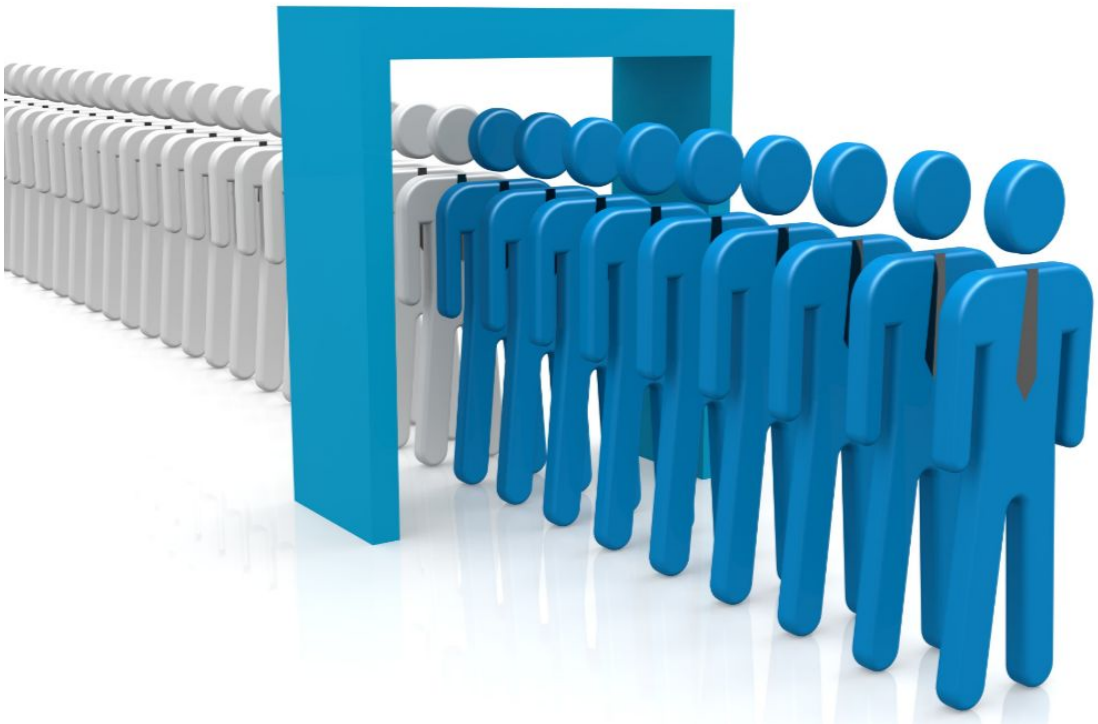
```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page"
     xmlns:c="http://java.sun.com/jsp/jstl/core"
     xmlns:joda="http://www.joda.org/joda/time/tags"
     version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>

    <h1>Contact Listing</h1>

    <c:if test="${not empty contacts}">
        <table>
            <thead>
                <tr>
                    <th>First Name</th>
                    <th>Last Name</th>
                    <th>Birth Date</th>
                </tr>
            </thead>
            <tbody>
                <c:forEach items="${contacts}" var="contact">
                    <tr>
                        <td>${contact.firstName}</td>
                        <td>${contact.lastName}</td>
                        <td><joda:format value="${contact.birthDate}" pattern="yyyy-MM-dd"/></td>
                    </tr>
                </c:forEach>
            </tbody>
        </table>
    </c:if>
</div>
```

# Model examples


- Map<String, Object>
- Populated by controllers
- Contains all data needed by the „view“
- Not containing business logic

# Supported method return types


- ModelAndView
- Model
- Map
- View
- String
- void
- ….



# INTERNATIONALIZATION(I18N) / THEMES / TEMPLATES (APACHE TILES)

# Internationalization


- Enable i18n in the early stage.
- Properties files within the e.g /WEB-INF/i18n folder:
  - ✔ The **application*.properties**
  - ✔ The **message*.properties**

# Configuring i18n in DispatcherServlet

```
    <beans:bean
        class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor"
        p:paramName="lang" />
</interceptors>
```

# Configuring i18n in DispatcherServlet

```
<beans:bean
    class="org.springframework.context.support.ReloadableResourceBundleMessageSource"
    id="messageSource" p:basenames="WEB-INF/i18n/messages,WEB-INF/i18n/application"
    p:fallbackToSystemLocale="true" />
```

# Configuring i18n in DispatcherServlet

```
<beans:bean class="org.springframework.web.servlet.i18n.CookieLocaleResolver"
    id="localeResolver" p:cookieName="locale" />
```

# Internationalization(i18n) / Themes / Templates (Apache Tiles)

## Example View


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page"
    xmlns:c="http://java.sun.com/jsp/jstl/core"
    xmlns:joda="http://www.joda.org/joda/time/tags"
    xmlns:spring="http://www.springframework.org/tags"
    version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>

    <spring:message code="label_contact_list" var="labelContactList"/>
    <spring:message code="label_contact_first_name" var="labelContactFirstName"/>
    <spring:message code="label_contact_last_name" var="labelContactLastName"/>
    <spring:message code="label_contact_birth_date" var="labelContactBirthDate"/>

    <h1>${labelContactList}</h1>
    <c:if test="${not empty contacts}">
        <table>
            <thead>
                <tr>
                    <th>${labelContactFirstName}</th>
                    <th>${labelContactLastName}</th>
                    <th>${labelContactBirthDate}</th>
                </tr>
            </thead>
            <tbody>
                <c:forEach items="${contacts}" var="contact">
                    <tr>
                        <td>${contact.firstName}</td>
                        <td>${contact.lastName}</td>
                        <td><joda:format value="${contact.birthDate}" pattern="yyyy-MM-dd"/></td>
                    </tr>
                </c:forEach>
            </tbody>
        </table>
    </c:if>
</div>
```

# Internationalization(i18n) / Themes / Templates (Apache Tiles)

## Using Themes

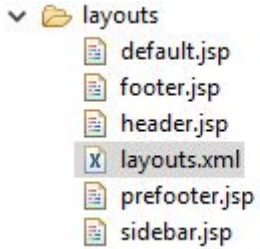

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xh
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://www.springframework.org/tags" prefix="spring"%>
<%@ taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>

<html>
<head>
<meta http-equiv="Content-type" content="text/html; charset=UTF-8" />

<title><spring:message code="welcome_h3" arguments="${app_name}" /></title>

<spring:theme code="styleSheet" var="app_css" />
<spring:url value="/${app_css}" var="app_css_url" />
<link rel="stylesheet" type="text/css" media="screen"
    href="${app_css_url}" />
```

```
<beans:bean
    class="org.springframework.web.servlet.theme.ThemeChangeInterceptor" />

<beans:bean
    class="org.springframework.ui.context.support.ResourceBundleThemeSource"
    id="themeSource" />
<beans:bean class="org.springframework.web.servlet.theme.CookieThemeResolver"
    id="themeResolver" p:cookieName="theme" p:defaultThemeName="standard" />
```

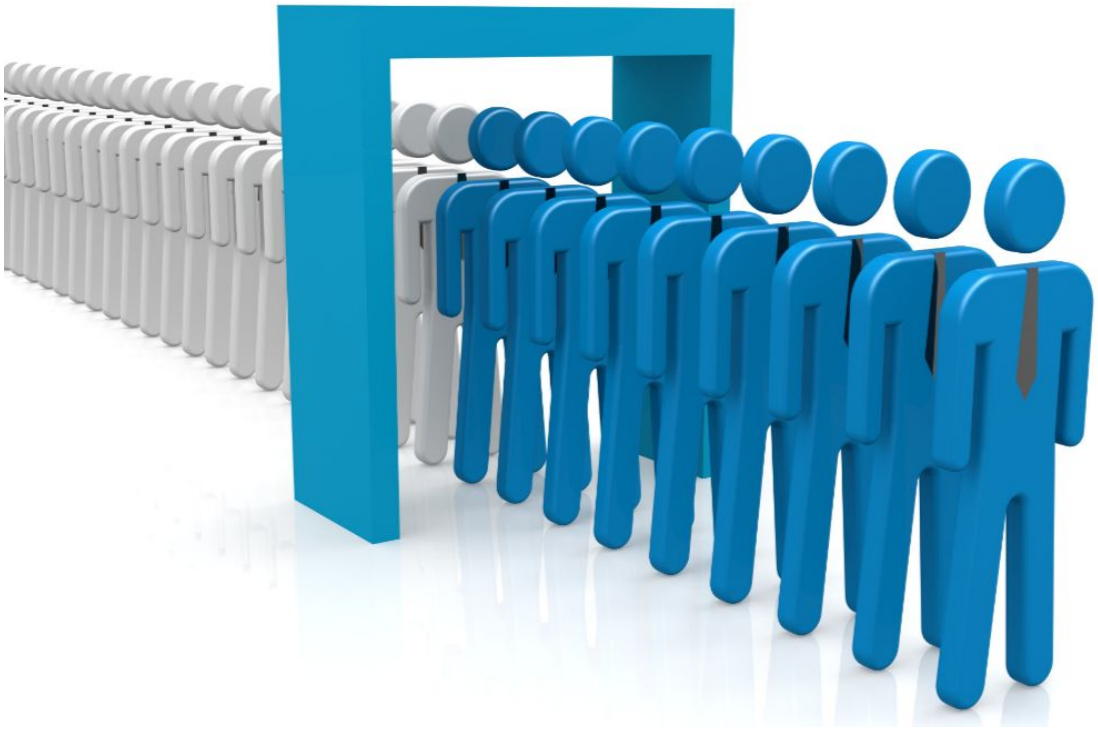
# Using Themes


- FixedThemeResolver
- SessionThemeResolver
- CookieThemeResolver

# Using templates

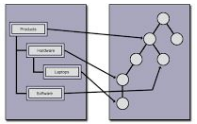

```xml
<?xml version="1.0" encoding="UTF-16"?>
<!DOCTYPE tiles-definitions PUBLIC
        "-//Apache Software Foundation//DTD Tiles Configuration 2.1//EN"
        "http://tiles.apache.org/dtds/tiles-config_3_0.dtd">

<tiles-definitions>
    <definition name="default" template="/WEB-INF/layouts/default.jsp">
        <put-attribute name="header" value="/WEB-INF/layouts/header.jsp" />
        <put-attribute name="sidebar" value="/WEB-INF/layouts/sidebar.jsp" />
        <put-attribute name="prefooter" value="/WEB-INF/layouts/prefooter.jsp" />
        <put-attribute name="footer" value="/WEB-INF/layouts/footer.jsp" />
    </definition>
</tiles-definitions>
```



# MAPPING OF URLS, VALIDATION SUPPORT

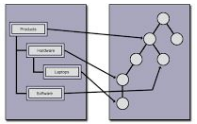
# Mapping of URLs, Validation support (JSR-349)

## Mapping of URLs to the views



@RequestMapping

# Mapping of URLs to the views

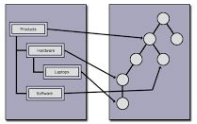


## @RequestMapping

- ✔ consumes

  consumes="application/json“;
  consumes=“!application/json“;
  "text/plain", "application/*;

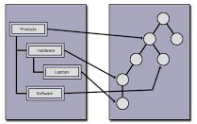
# Mapping of URLs to the views



## @RequestMapping

- ✔ consumes
  - consumes="application/json“;
  - consumes=“!application/json“;
  - "text/plain", "application/*;
- ✔ headers
  - headers="myHeader=myValue“;
  - headers="!myHeader=myValue“;
  - headers="myHeader", headers="!myHeader“;

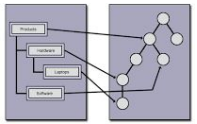
# Mapping of URLs to the views



## @RequestMapping

- ✔ consumes
  consumes="application/json“;
  consumes=“!application/json“;
  "text/plain", "application/*;
- ✔ headers
  headers="myHeader=myValue“;
  headers="!myHeader=myValue“;
  headers="myHeader", headers="!myHeader“;
- ✔ method
  RequestMethod.GET, RequestMethod.POST etc.

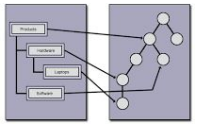
# Mapping of URLs to the views



## @RequestMapping

- ✔ consumes
  consumes="application/json“;
  consumes=“!application/json“;
  "text/plain", "application/*;
- ✔ headers
  headers="myHeader=myValue“;
  headers="!myHeader=myValue“;
  headers="myHeader", headers="!myHeader“;
- ✔ method
  RequestMethod.GET, RequestMethod.POST etc.
- ✔ name
- ✔ params
  params="myParam=myValue“;
  params="!myParam=myValue“;

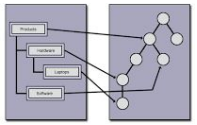
# Mapping of URLs to the views



## @RequestMapping

- ✔ consumes
  consumes="application/json“;
  consumes=“!application/json“;
  "text/plain", "application/*;
- ✔ headers
  headers="myHeader=myValue“;
  headers="!myHeader=myValue“;
  headers="myHeader", headers="!myHeader“;
- ✔ method
  RequestMethod.GET, RequestMethod.POST etc.
- ✔ name
- ✔ params
  params="myParam=myValue“;
  params="!myParam=myValue“;
- ✔ path

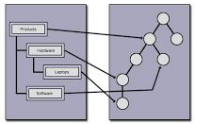
# Mapping of URLs to the views



## @RequestMapping

- ✔ consumes
  consumes="application/json“;
  consumes=“!application/json“;
  "text/plain", "application/*;
- ✔ headers
  headers="myHeader=myValue“;
  headers="!myHeader=myValue“;
  headers="myHeader", headers="!myHeader“;
- ✔ method
  RequestMethod.GET, RequestMethod.POST etc.
- ✔ name
- ✔ params
  params="myParam=myValue“;
  params="!myParam=myValue“;
- ✔ path
- ✔ produces

# Mapping of URLs to the views

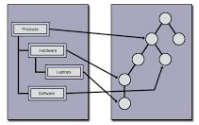


## @RequestMapping

- ✔ consumes
  consumes="application/json“;
  consumes=“!application/json“;
  "text/plain", "application/*;
- ✔ headers
  headers="myHeader=myValue“;
  headers="!myHeader=myValue“;
  headers="myHeader", headers="!myHeader“;
- ✔ method
  RequestMethod.GET, RequestMethod.POST etc.
- ✔ name
- ✔ params
  params="myParam=myValue“;
  params="!myParam=myValue“;
- ✔ path
- ✔ produces
- ✔ value

# Mapping of URLs, Validation support (JSR-349)

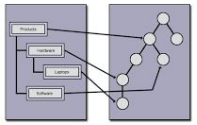
## Mapping example

```
@Controller
@RequestMapping(value = "/registration")
public class RegistrationPageController {
```

# Mapping of URLs, Validation support (JSR-349)

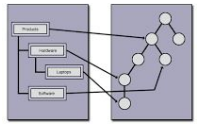
## Mapping example

```
@Controller
@RequestMapping(value = "/registration")
public class RegistrationPageController {

@RequestMapping(method = RequestMethod.GET)
public String registration(Model model) {
```

# Mapping of URLs, Validation support (JSR-349)

## Mapping example

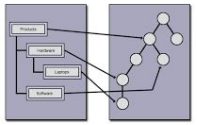



```
@Controller
@RequestMapping(value = "/registration")
public class RegistrationPageController {

@RequestMapping(method = RequestMethod.GET)
public String registration(Model model) {
```

```
@RequestMapping(value = "/{id}", method = RequestMethod.GET)
public String show(@PathVariable("id") Long id, Model uiModel) {
```

# Mapping of URLs, Validation support (JSR-349)

## Mapping example

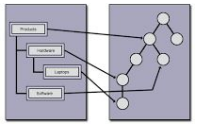
```
@Controller
@RequestMapping(value = "/registration")
public class RegistrationPageController {

@RequestMapping(method = RequestMethod.GET)
public String registration(Model model) {
```

```
@RequestMapping(value = "/{id}", method = RequestMethod.GET)
public String show(@PathVariable("id") Long id, Model uiModel) {
```

```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.GET)
public String updateForm(@PathVariable("id") Long id, Model uiModel) {
    uiModel.addAttribute("contact", contactService.findById(id));
    return "contacts/update";
}
```

# Mapping of URLs, Validation support (JSR-349)

## Mapping example

```
@Controller
@RequestMapping(value = "/registration")
public class RegistrationPageController {

@RequestMapping(method = RequestMethod.GET)
public String registration(Model model) {
```

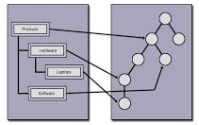
```
@RequestMapping(value = "/{id}", method = RequestMethod.GET)
public String show(@PathVariable("id") Long id, Model uiModel) {
```

```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.GET)
public String updateForm(@PathVariable("id") Long id, Model uiModel) {
    uiModel.addAttribute("contact", contactService.findById(id));
    return "contacts/update";
}
```

```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.POST)
public String update(Contact contact, BindingResult bindingResult, Model uiModel,
                     HttpServletRequest httpServletRequest, RedirectAttributes redirectAttributes, Locale locale)
{
    logger.info("Updating contact");
    if (bindingResult.hasErrors()) {
        uiModel.addAttribute("message", new Message("error", messageSource.getMessage("contact_save_fail",
new Object[]{}, locale)));
        uiModel.addAttribute("contact", contact);
        return "contacts/update";
    }
```

# Mapping of URLs, Validation support (JSR-349)

## Mapping example

```
@Controller
@RequestMapping(value = "/registration")
public class RegistrationPageController {

@RequestMapping(method = RequestMethod.GET)
public String registration(Model model) {
```

```
@RequestMapping(value = "/{id}", method = RequestMethod.GET)
public String show(@PathVariable("id") Long id, Model uiModel) {
```

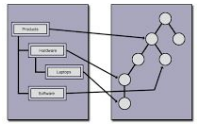
```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.GET)
public String updateForm(@PathVariable("id") Long id, Model uiModel) {
    uiModel.addAttribute("contact", contactService.findById(id));
    return "contacts/update";
}
```

```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.POST)
public String update(Contact contact, BindingResult bindingResult, Model uiModel,
                     HttpServletRequest httpServletRequest, RedirectAttributes redirectAttributes, Locale locale)
{
    logger.info("Updating contact");
    if (bindingResult.hasErrors()) {
        uiModel.addAttribute("message", new Message("error", messageSource.getMessage("contact_save_fail",
new Object[]{}, locale)));
        uiModel.addAttribute("contact", contact);
        return "contacts/update";
    }
```

```
@RequestMapping(value = "/view", params="details=all")
```

# Mapping of URLs, Validation support (JSR-349)

## Mapping example

```
@Controller
@RequestMapping(value = "/registration")
public class RegistrationPageController {

@RequestMapping(method = RequestMethod.GET)
public String registration(Model model) {
```

```
@RequestMapping(value = "/{id}", method = RequestMethod.GET)
public String show(@PathVariable("id") Long id, Model uiModel) {
```

```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.GET)
public String updateForm(@PathVariable("id") Long id, Model uiModel) {
    uiModel.addAttribute("contact", contactService.findById(id));
    return "contacts/update";
}
```

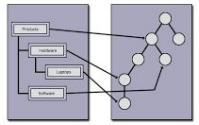
```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.POST)
public String update(Contact contact, BindingResult bindingResult, Model uiModel,
                     HttpServletRequest httpServletRequest, RedirectAttributes redirectAttributes, Locale locale)
{
    logger.info("Updating contact");
    if (bindingResult.hasErrors()) {
        uiModel.addAttribute("message", new Message("error", messageSource.getMessage("contact_save_fail",
new Object[]{}, locale)));
        uiModel.addAttribute("contact", contact);
        return "contacts/update";
    }
```

```
@RequestMapping(value = "/view", params="details=all")
```

/registration/view?details=all

# Mapping of URLs, Validation support (JSR-349)

## Mapping example

```
@Controller
@RequestMapping(value = "/registration")
public class RegistrationPageController {

@RequestMapping(method = RequestMethod.GET)
public String registration(Model model) {
```

```
@RequestMapping(value = "/{id}", method = RequestMethod.GET)
public String show(@PathVariable("id") Long id, Model uiModel) {
```

```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.GET)
public String updateForm(@PathVariable("id") Long id, Model uiModel) {
    uiModel.addAttribute("contact", contactService.findById(id));
    return "contacts/update";
}
```

```
@RequestMapping(value = "/{id}", params = "form", method = RequestMethod.POST)
public String update(Contact contact, BindingResult bindingResult, Model uiModel,
                     HttpServletRequest httpServletRequest, RedirectAttributes redirectAttributes, Locale locale)
{
    logger.info("Updating contact");
    if (bindingResult.hasErrors()) {
        uiModel.addAttribute("message", new Message("error", messageSource.getMessage("contact_save_fail",
new Object[]{}, locale)));
        uiModel.addAttribute("contact", contact);
        return "contacts/update";
    }
```

```
@RequestMapping(value = "/view", params="details=all")
```

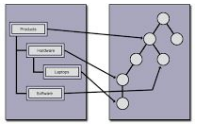
/registration/view?details=all

```
@RequestMapping("/viewProject/{projectId}")
```

/registration/viewProject/10

# Mapping of URLs, Validation support (JSR-349)

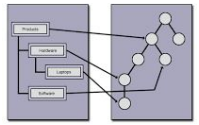
## Mapping example

```
@RequestMapping(path="/owners/{ownerId}", method=RequestMethod.GET)
public String findOwner(@PathVariable String ownerId, Model model) {
    Owner owner = ownerService.findOwner(ownerId);
    model.addAttribute("owner", owner);
    return "displayOwner";
}
```

# Mapping of URLs, Validation support (JSR-349)

## Mapping example

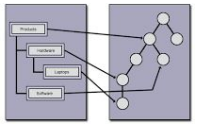



```java
@RequestMapping(path="/owners/{ownerId}", method=RequestMethod.GET)
public String findOwner(@PathVariable String ownerId, Model model) {
    Owner owner = ownerService.findOwner(ownerId);
    model.addAttribute("owner", owner);
    return "displayOwner";
}
```

```java
@RequestMapping("/spring-web/{symbolicName:[a-z-]+}-{version:\\d\\.\\d\\.\\d}{extension:\\.[a-z]+}")
    public void handle(@PathVariable String version, @PathVariable String extension) {
        // ...
    }
}
```

/spring-web/spring-web-3.0.5.jar

# Spring MVC Annotations
## Composed @RequestMapping Variants



- @GetMapping
- @PostMapping
- @PutMapping
- @DeleteMapping
- @PatchMapping

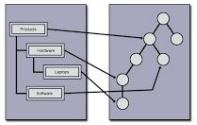
```
@RequestMapping(method = RequestMethod.GET)
public Map<String, Appointment> get() {
return …;
}
```

```
@GetMapping
public Map<String, Appointment> get() {
return …;
}
```

```
@RequestMapping(method = RequestMethod.POST)
public String add(@Valid AppointmentForm
appointment, BindingResult result) {
return …;
}
```

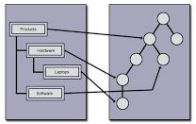
```
@PostMapping
public String add(@Valid AppointmentForm
appointment, BindingResult result) {
return …;
}
```

# Path Patterns



/myPath/*.do

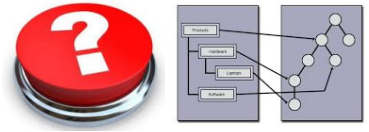
# Path Patterns



/myPath/*.do

/owners/*/pets/{petId}

# Path Patterns

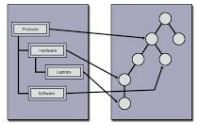


/myPath/*.do

/owners/*/pets/{petId}

/hotels/{hotel}/* VS /hotels/{hotel}/**

# Path Patterns

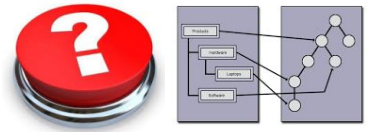


/myPath/*.do

/owners/*/pets/{petId}

/hotels/{hotel}/* VS /hotels/{hotel}/**

# Path Patterns



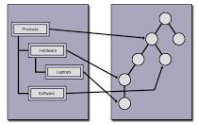
/myPath/*.do

/owners/*/pets/{petId}

/hotels/{hotel}/* VS /hotels/{hotel}/**

/foo/bar* VS /foo/*

# Path Patterns



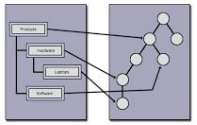
/myPath/*.do

/owners/*/pets/{petId}

/hotels/{hotel}/* VS /hotels/{hotel}/**

/foo/bar* VS /foo/*

# Path Patterns



/myPath/*.do

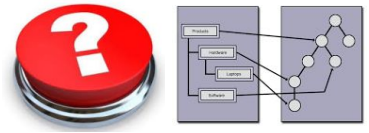
/owners/*/pets/{petId}

/hotels/{hotel}/* VS /hotels/{hotel}/**

/foo/bar* VS /foo/*

/hotels/* VS /hotels/{hotel}

# Path Patterns



/myPath/*.do

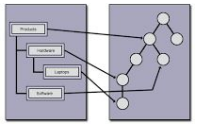
/owners/*/pets/{petId}

/hotels/{hotel}/* VS /hotels/{hotel}/**

/foo/bar* VS /foo/*

/hotels/* VS /hotels/{hotel}

# Path Patterns



/myPath/*.do

/owners/*/pets/{petId}

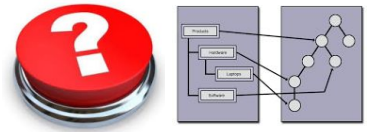
/hotels/{hotel}/* VS /hotels/{hotel}/**

/foo/bar* VS /foo/*

/hotels/* VS /hotels/{hotel}

/** VS /api/{a}/{b}/{c}

# Path Patterns



/myPath/*.do

/owners/*/pets/{petId}

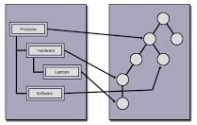
/hotels/{hotel}/* VS /hotels/{hotel}/**

/foo/bar* VS /foo/*

/hotels/* VS /hotels/{hotel}

/** VS /api/{a}/{b}/{c}

# Path Patterns



/myPath/*.do

/owners/*/pets/{petId}

/hotels/{hotel}/* VS /hotels/{hotel}/**

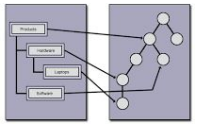
/foo/bar* VS /foo/*

/hotels/* VS /hotels/{hotel}

/** VS /api/{a}/{b}/{c}

/public/** VS /public/path3/{a}/{b}/{c}

# Path Patterns



/myPath/*.do

/owners/*/pets/{petId}

/hotels/{hotel}/* VS /hotels/{hotel}/**

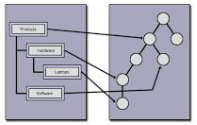
/foo/bar* VS /foo/*

/hotels/* VS /hotels/{hotel}

/** VS /api/{a}/{b}/{c}

/public/** VS /public/path3/{a}/{b}/{c}

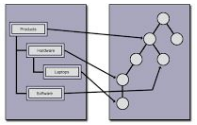
# Spring MVC Annotations
## @RequestBody



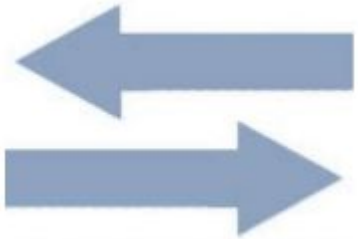
- @PathVariable
- @RequestParam
- @RequestHeader
- @RequestBody

```
@PutMapping("/something")
public void handle(@RequestBody String body, Writer writer) throws IOException {
writer.write(body);
}
```

# Spring MVC Annotations
## @ResponseBody

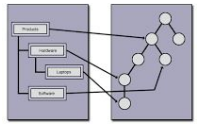


```
@GetMapping("/something")
@ResponseBody
public String helloWorld() {
return "Hello World";
}
```



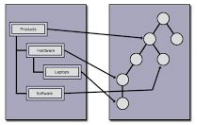
HttpMessageConverter

# @RestController

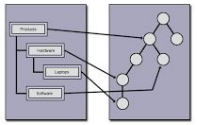
@RestController

# Spring MVC Annotations
## @ModelAttribute on a method



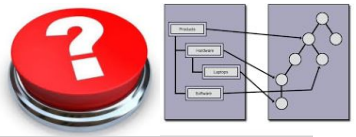
```
// Add one attribute
// The return value of the method is added to the model under the name "account"
// You can customize the name via @ModelAttribute("myAccount")
@ModelAttribute
public Account addAccount(@RequestParam String number) {
  return accountManager.findAccount(number);
}
  // Add multiple attributes
@ModelAttribute
public void populateModel(@RequestParam String number, Model model) {
  model.addAttribute(accountManager.findAccount(number));
  // add more ...
}
```

# Spring MVC Annotations
## @ModelAttribute on method arguments



```
@PostMapping("/owners/{ownerId}/pets/{petId}/edit")
public String processSubmit(@ModelAttribute Pet pet) { }
```

# Content Negotiation

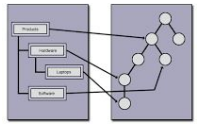


```java
@Configuration
@EnableWebMvc
public class WebConfig extends WebMvcConfigurerAdapter {

    @Override
    public void configureContentNegotiation(ContentNegotiationConfigurer configurer) {
        configurer.mediaType("json", MediaType.APPLICATION_JSON);
    }
}
```

```xml
<mvc:annotation-driven content-negotiation-manager="contentNegotiationManager"/>

<bean id="contentNegotiationManager" class="org.springframework.web.accept.ContentNegotiationManagerFactoryBean">
    <property name="mediaTypes">
        <value>
            json=application/json
            xml=application/xml
        </value>
    </property>
</bean>
```

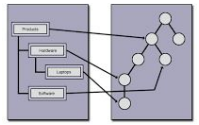
# Mapping of URLs, Validation support (JSR-349)

## Configure JSR-349, “Bean Validation”



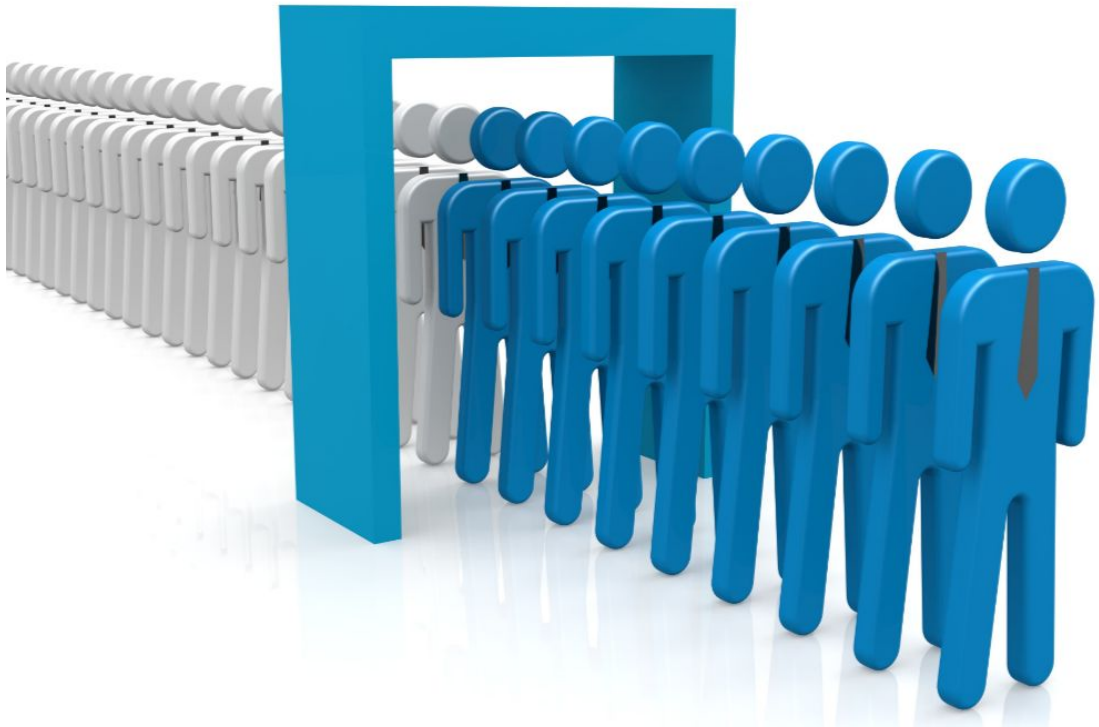
```
@Entity
@Table(name = "contact")
public class Contact implements Serializable {
  private Long id;
  private String firstName;
  private String lastName;

  @Id
  @GeneratedValue(strategy = IDENTITY)
  @Column(name = "ID")
  public Long getId() {
      return id;
  }
  public void setId(Long id) {
      this.id = id;
  }
  @NotEmpty(message="{validation.firstname.NotEmpty.message}")
  @Size(min=3, max=60, message="{validation.firstname.Size.message}")
  @Column(name = "FIRST_NAME")
  public String getFirstName() {
      return firstName;
  }
  public void setFirstName(String firstName) {
      this.firstName = firstName;
  }
  @NotEmpty(message="{validation.lastname.NotEmpty.message}")
  @Size(min=1, max=40, message="{validation.lastname.Size.message}")
  @Column(name = "LAST_NAME")
  public String getLastName() {
      return lastName;
  }
  public void setLastName(String lastName) {
      this.lastName = lastName;
  }
}
```

# Mapping of URLs, Validation support (JSR-349)
## Configure JSR-349, “Bean Validation”



*public String update(**@Valid** Contact contact, ...*
*public String create(**@Valid** Contact contact, ...*

```
<annotation-driven validator="validator"/>
<beans:bean id="validator" class="org.springframework.validation.beanvalidation.LocalValidatorFactoryBean">
        <beans:property name="validationMessageSource" ref="messageSource"/>
 </beans:bean>
```



# FILE UPLOAD HANDLING

# File Upload Handling
## Overview

Apache Commons FileUpload

Tomcat 7 -> Servlet 3.0 -> Spring 3.1

# File Upload Handling
## Enable File Upload support


```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns=http://java.sun.com/xml/ns/javaee
    <!-- Processes application requests -->
    <servlet>
        <servlet-name>training</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/training/training-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
        <multipart-config>
            <max-file-size>5000000</max-file-size>
        </multipart-config>
    </servlet>
</web-app>
```

# File Upload Handling
## Enable File Upload support


```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns=http://java.sun.com/xml/ns/javaee
    <!-- Processes application requests -->
    <servlet>
        <servlet-name>training</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/spring/training/training-servlet.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
        <multipart-config>
            <max-file-size>5000000</max-file-size>
        </multipart-config>
    </servlet>
</web-app>
```

```
<beans:bean
        class="org.springframework.web.multipart.support.StandardServletMultipartResolver"
        id="multipartResolver" >
</beans:bean>
```

# File Upload Handling

## Modifying Views for File Upload Support


```
<div id="contactInfo">
    <form:form modelAttribute="customer" id="customer" method="POST" enctype="multipart/form-data">

  <td><img src="${contactPhotoUrl}/${customer.id}" height="150" width="150"></img></td>
```

# File Upload Handling

## Modifying Views for File Upload Support


```
<div id="contactInfo">
    <form:form modelAttribute="customer" id="customer" method="POST" enctype="multipart/form-data">

  <td><img src="${contactPhotoUrl}/${customer.id}" height="150" width="150"></img></td>
```

```
private void processUploadFile(@ModelAttribute("customer") Customer customer,
        @RequestParam(value = "file", required = false) Part file) {
    if (file != null) {
        logger.info("File name: " + file.getName());
        logger.info("File size: " + file.getSize());
        logger.info("File content type: " + file.getContentType());
        byte[] fileContent = null;
        try {
            InputStream inputStream = file.getInputStream();
            if (inputStream == null)
                logger.info("File inputstream is null");
            fileContent = IOUtils.toByteArray(inputStream);
            customer.setPhoto(fileContent);
        } catch (IOException ex) {
            logger.error("Error saving uploaded file");
        }
        customer.setPhoto(fileContent);
    }
}
```

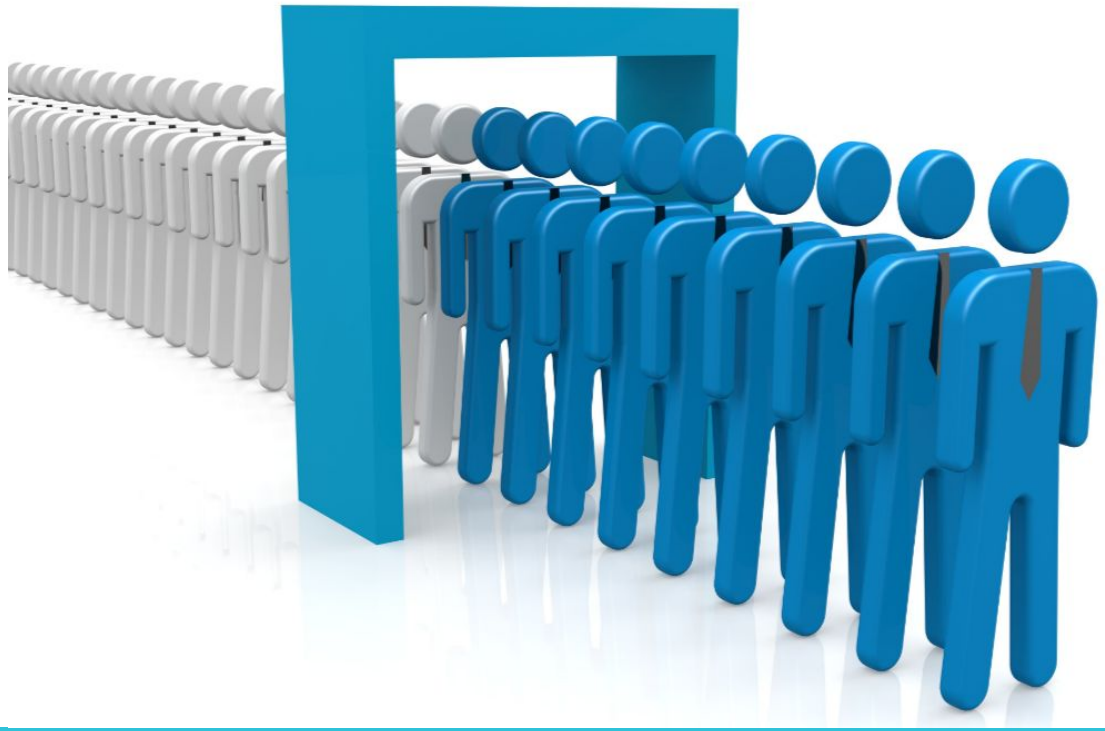
# File Upload Handling
## Modifying Views for File Upload Support


```
<div id="contactInfo">
    <form:form modelAttribute="customer" id="customer" method="POST" enctype="multipart/form-data">

  <td><img src="${contactPhotoUrl}/${customer.id}" height="150" width="150"></img></td>
```

```
private void processUploadFile(@ModelAttribute("customer") Customer customer,
        @RequestParam(value = "file", required = false) Part file) {
    if (file != null) {
        logger.info("File name: " + file.getName());
        logger.info("File size: " + file.getSize());
        logger.info("File content type: " + file.getContentType());
        byte[] fileContent = null;
        try {
            InputStream inputStream = file.getInputStream();
            if (inputStream == null)
                logger.info("File inputstream is null");
            fileContent = IOUtils.toByteArray(inputStream);
            customer.setPhoto(fileContent);
        } catch (IOException ex) {
            logger.error("Error saving uploaded file");
        }
        customer.setPhoto(fileContent);
    }
}
```

```
25 <spring:message code="label_contact_country" var="labelContactCountry" />
26 <spring:url value="/customerpanel/photo" var="contactPhotoUrl" />
27 <spring:url value="/customerpanel" var="editContactUrl" />
28 <spring:url value="/customerpanel/delete" var="deleteContactUrl" />
29 <spring:message code="label_contact_photo" var="labelContactPhoto" />
30 </head>
```

```
@RequestMapping(value = "/photo/{id}", method = RequestMethod.GET)
@ResponseBody
public byte[] downloadPhoto(@PathVariable("id") Long id) {
    Customer customer = customerService.findOne(id);

    if (customer.getPhoto() != null) {
        logger.info("Downloading photo for id: {} with size: {}", cust
    }

    return customer.getPhoto();
}
```



# SUPPORTING SERVLET 3.0 CODE-BASED (JAVA-BASED) CONFIGURATION

# Configuration

Advantages?

# Supporting Servlet 3.0 Code-Based (Java-Based) Configuration
## Configuration

Advantages?

- ❑ Java синтаксис
- ❑ Code IDE advantages
- ❑ Flexibility
- ❑ No need in full build and redeploy

# Configuration

Advantages?

- ❑ Java синтаксис
- ❑ Code IDE advantages
- ❑ Flexibility
- ❑ No need in full build and redeploy

org.springframework.web.WebApplicationInitializer

org.springframework.web.SpringServletContainerInitializer

# Supporting Servlet 3.0 Code-Based (Java-Based) Configuration
## Configuration

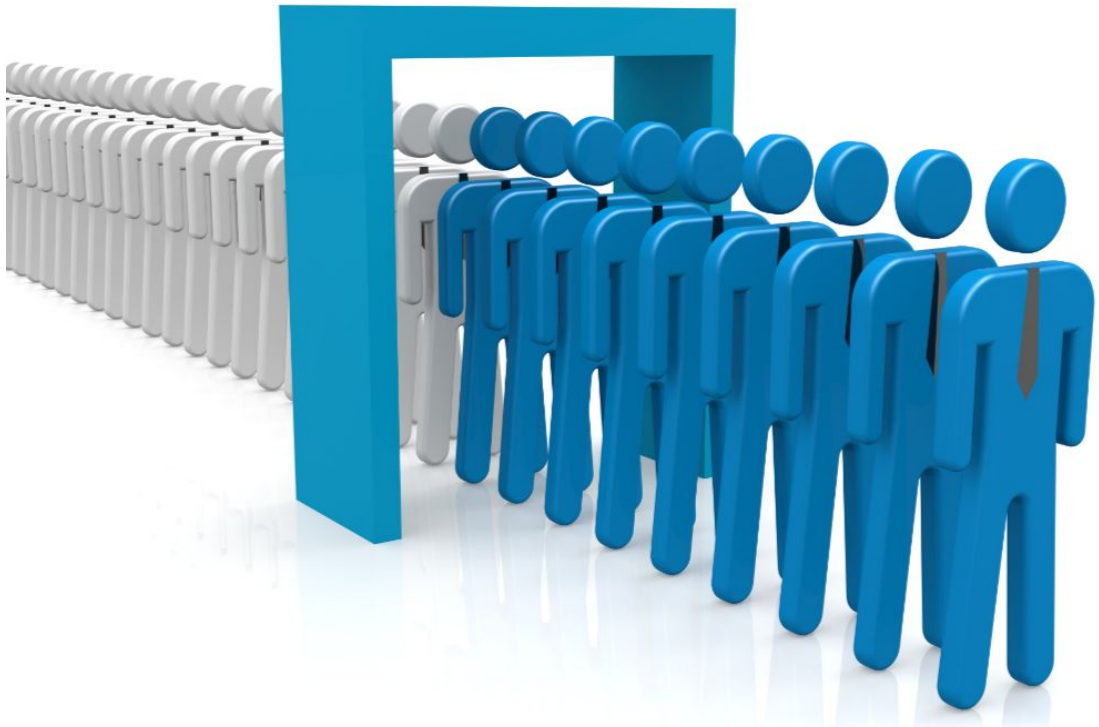
```
public class MyWebAppInitializer implements
WebApplicationInitializer {
   @Override
   public void onStartup(ServletContext container) throws
ServletException {
       XmlWebApplicationContext appContext = new
XmlWebApplicationContext();

appContext.setConfigLocation("/WEB-INF/spring/appServlet/s
ervlet-context.xml");
       ServletRegistration.Dynamic dispatcher =
       container.addServlet("appServlet", new
DispatcherServlet(appContext));
      MultipartConfigElement multipartConfigElement =
      new MultipartConfigElement(null, 5000000, 5000000, 0);
      dispatcher.setMultipartConfig(multipartConfigElement);
      dispatcher.setLoadOnStartup(1);
      dispatcher.addMapping("/");
   }
}
```

# Supporting Servlet 3.0 Code-Based (Java-Based) Configuration

## Configuration

```
public class MyWebAppInitializer implements
WebApplicationInitializer {
   @Override
   public void onStartup(ServletContext container) throws
ServletException {
       XmlWebApplicationContext appContext = new
XmlWebApplicationContext();

appContext.setConfigLocation("/WEB-INF/spring/appServlet/s
ervlet-context.xml");
       ServletRegistration.Dynamic dispatcher =
       container.addServlet("appServlet", new
DispatcherServlet(appContext));
      MultipartConfigElement multipartConfigElement =
      new MultipartConfigElement(null, 5000000, 5000000, 0);
      dispatcher.setMultipartConfig(multipartConfigElement);
      dispatcher.setLoadOnStartup(1);
      dispatcher.addMapping("/");
   }
}
```

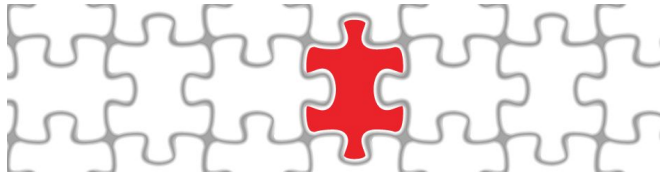
```
<servlet>
      <servlet-name>appServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherSer
vlet</servlet-class>
      <init-param>
          <param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring/appServlet/servlet-context.xml
</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
      <multipart-config>
          <max-file-size>5000000</max-file-size>
      </multipart-config>
</servlet>
<servlet-mapping>
      <servlet-name>appServlet</servlet-name>
      <url-pattern>/</url-pattern>
</servlet-mapping>
```



# HANDLING EXCEPTIONS

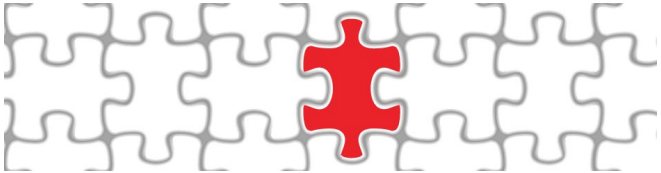
# Handling exceptions
## Dealing with exceptions

- Exceptions -> Status Codes
- org.springframework.web.servlet.HandlerExceptionResolver: Exception -> ModelAndView
- SimpleMappingExceptionResolver: Exceptions -> Views
- @ExceptionHandler, @ControllerAdvice
- @ResponseStatus

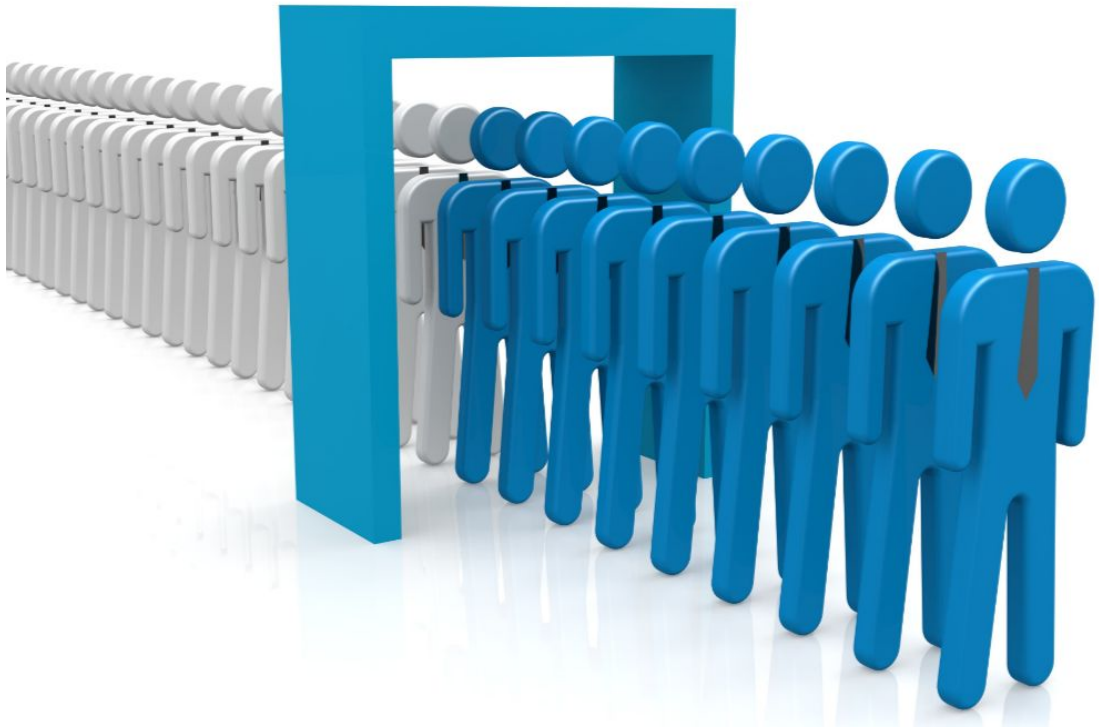


# Handling exceptions
## @ExceptionHandler & @ResponseStatus


```java
@Controller
public class SimpleController {

    // @RequestMapping methods omitted ...

    @ExceptionHandler(IOException.class)
    public ResponseEntity<String> handleIOException(IOException ex) {
        // prepare responseEntity
        return responseEntity;
    }

}
```



```java
@ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
@ExceptionHandler(DataAccessException.class)
public void handleDataAccessError(DataAccessException ex) {}

@ResponseStatus(value = HttpStatus.PAYMENT_REQUIRED, message = "I need money.")
public class PaymentRequiredException {}
```



# SPRING MVC AND SPRING SECURITY

# Introduction

- Declaration Security
- Support for many authentication and authorization schemes, such as basic, form-based, based on digests, JDBC and LDAP.
- Support for security at the level of methods and security annotations JSR-250
- Support for a one-time password
- Container integration support
- Supports anonymous sessions, simultaneous sessions, "remember me" mode, channel-level security and much more

# Libraries


*Maven Dependencies for Spring Security*

| Group ID | Artifact ID | Version | Description |
|---|---|---|---|
| org.springframework.security | spring-security-core | 3.2.1.RELEASE | Spring Security core module |
| org.springframework.security | spring-security-web | 3.2.1.RELEASE | Spring Security web module |
| org.springframework.security | spring-security-config | 3.2.1.RELEASE | Spring Security configuration module |
| org.springframework.security | spring-security-taglibs | 3.2.1.RELEASE | Spring Security JSP tag library |

# Configuring Spring Security


Configure a filter in the web deployment descriptor:

```
<!-- Spring Security Configuration -->
<filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
</filter>
<filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<!-- Filters ... -->
```

# Configuring Spring Security


```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/security"
             xmlns:beans="http://www.springframework.org/schema/beans"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/security
        http://www.springframework.org/schema/security/spring-security.xsd">

    <http use-expressions="true">
        <intercept-url pattern='/WEB-INF' access='permitAll' />
        <form-login login-page="/contacts" authentication-failure-url="/security/loginfail"
                    default-target-url="/contacts" />
        <logout logout-success-url="/contacts"/>
    </http>

    <authentication-manager>
        <authentication-provider>
            <user-service>
                <user name="user" password="user" authorities="ROLE_USER" />
            </user-service>
        </authentication-provider>
    </authentication-manager>
</beans:beans>
```

# Spring MVC and Spring Security

## Adding Login Functions to the Application


```
1  <%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
2  <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
3  <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
4  <%@ taglib uri="http://www.springframework.org/security/tags" prefix="sec" %>
5
6  <c:set var="contextPath" value="${pageContext.request.contextPath}" />
7  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
8  <html>
9      <head>
10         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
11         <title>Login page</title>
12         <spring:message code="label.username" var="username" />
13         <spring:message code="label.password" var="password" />
14         <spring:message code="welcome_titlepane" var="welcome" />
15         <spring:message code="label.notlogin" var="notlogin" />
16         <spring:url var="tocustomerpanel" value="/customerpanel" />
17         <spring:message code="label_customerpanel" var="labelPanel" />
18     </head>
19     <body>
20         ${welcome}
21         <sec:authorize access="!isAuthenticated()">
22             <a href="<c:url value="/login" />" >${notlogin}</a>
23             <p/>
24         </sec:authorize>
25         <sec:authorize access="isAuthenticated()">${labelWelcome}
26             <sec:authentication property="principal.username" />
27             <br/>
28             <a href="${tocustomerpanel}">${labelPanel}</a>
29         </sec:authorize>
30         <sec:authorize access="hasAuthority('ROLE_ADMIN')">
31             You're acting as Admin!'
32         </sec:authorize>
33     </body>
34 </html>
```

# Enable Method-Level Security


```
<!-- Enable controller method level security -->
<security:global-method-security pre-post-annotations="enabled"/>
```

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true, prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

# Spring MVC and Spring Security
## Enable Method-Level Security


```
<!-- Enable controller method level security -->
<security:global-method-security pre-post-annotations="enabled"/>
```

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(securedEnabled = true, prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
```

```
@PreAuthorize("isAuthenticated()")
@RequestMapping("/customerpanel")
@Controller
public class CustomerController {
    private final Logger logger = LoggerFactory.getLogger(CustomerController.class);

    @Autowired
    private CustomerService customerService;
    @Autowired
    private MessageSource messageSource;
```

- http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle
- http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
- http://en.wikipedia.org/wiki/Spring_Framework#Model-view-controller_framework

# HELPFUL LINKS

# THANK YOU FOR YOUR ATTENTION