#Video1 (Why MVC...)

**Servlet** - In the java code the HTML code is embeded. is one ot the first technology providing dynamically generated content to the user. The problem was that the servlet class extending HttpServlet had to contain all the logic and provide HTML code needed to generete HTML file. That was a bit messy.

**JSP** - In the hmtl code the java code is embeded. That was an alternative to the Servlets. One of the problew was to write the whole sql sequences connecting to the DB (<sql:setDataSource var="" driver="" url="" user="" password="".>, <sql:query dataSource="" var="">SELECT \* FROM Employees</sql:query>).

None of above provided clear separation between business logic and HTML(HTML, Java, CSS, JavaScript, JQuery in one file!). **Problem to write, edit and maintain.**

**MVC is the solution**. It separates:

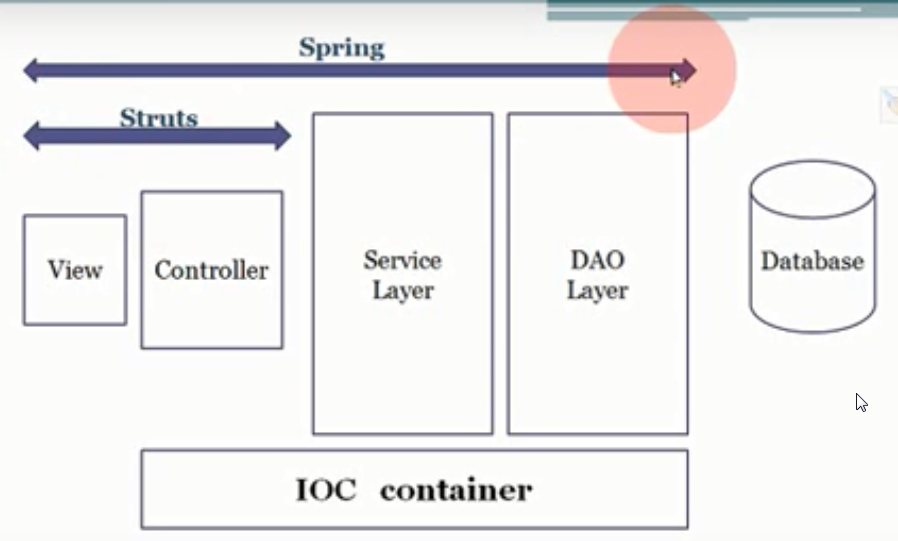
* **Model** - data to be displayed in web page: retrieving data from persistent storage
* **View** - Web page in HTML (simple JSP that contains only HTML and tag library) : HTML + CSS - presents model in user friendy interface (UI,gui)
* **Controller** - contains logic that solves domain problem

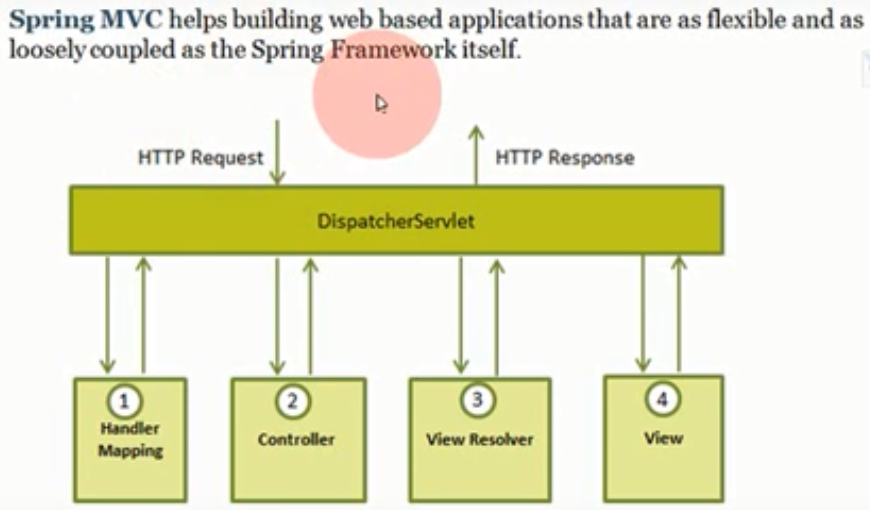
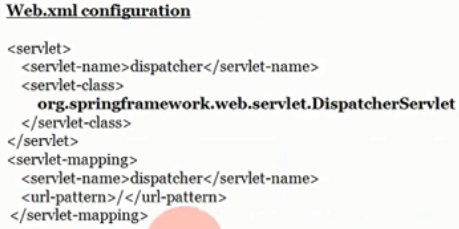
The Flow

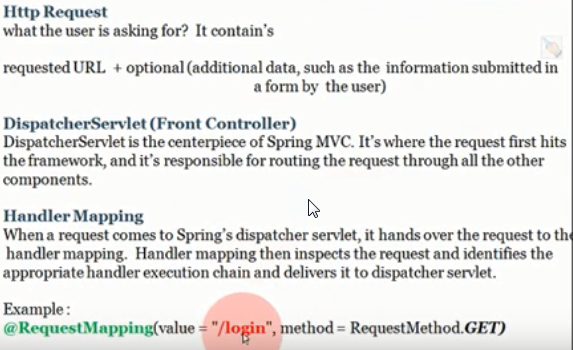
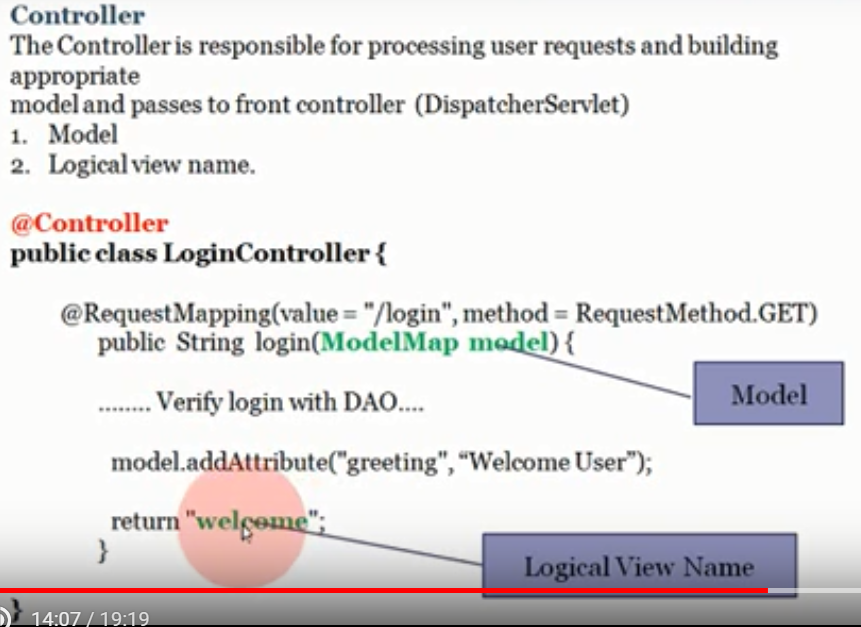
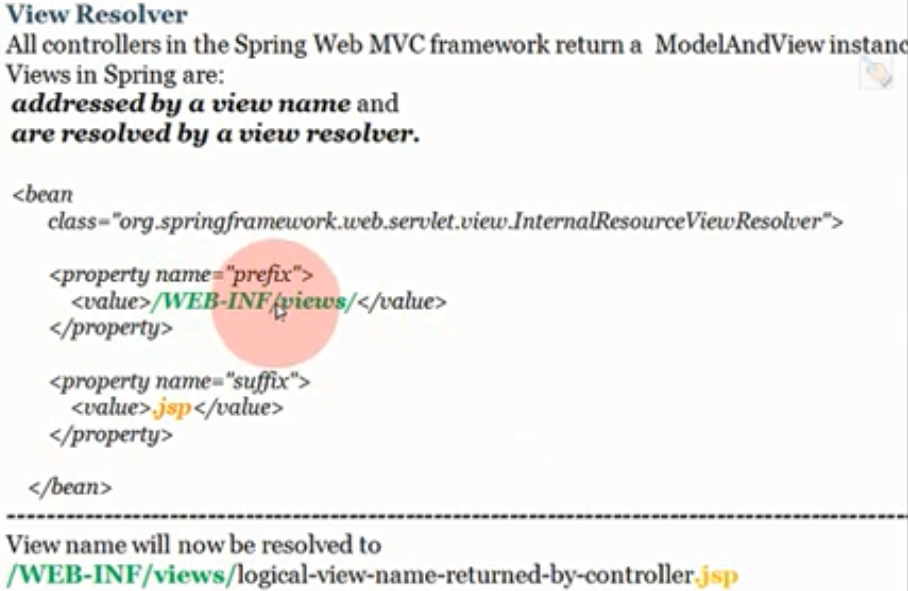
1. User request
2. Controller: prepares the model (call to model)
3. Controller: perform the logic
4. Controller: selects the view
5. Controller: sends the view to the model
6. View: renders the HTML page
7. Page is send to the client

#Video2 (Spring MVC Introduction & architecture)

**IOC Container** – Inversion of control container



**Dispatcher Servlet=Front Controller** – the center piece of the SpringMVC – it routs the information to different component. It is configured in the **Web.xml**. Below is the architecture of the spring framework.  
 

1. **Handler mapping** – selecting a controller that will handle the client request and passing it to the controller. Each request must match to unique **mapping** of a controller. 
2. **Controller** – contains add/delete/update/find. It processes the request, prepares a **model** and passes it to the front controller as well and the logical **view** name.  
   
3. **View resolver** – is a bean in the spring context configuration file. It adds prefix and suffix to build the path to the **view** (that is an instance of ModelAndView.class) and returns it to the front controller.
4. The view and the motel together can generate a HTML that is returned to the client

#video3 (Creating Environment for Spring MVC Using MAVEN)

Creating HelloWorldXml (xml in the name to indicate one of xml or java configuration)

1. Create Dynamic Web Project
   1. Configure the server (choose the server, point to the server installation directory,)
   2. Change source folder on build path to match maven structure. Remove “src” and add:
      1. “src/main/java”
      2. “src/main/resources” for static resources
      3. “src/main/webapp” for views
      4. “src/test/java”
   3. Change the content directory to “src/main/webapp” check generate web.xml.
2. Convert to maven project
   1. **GroupId** - is a base package name, where all the classes will be saved
   2. **ArtifactId** – is the project name
   3. **Version** – in the version of the project

Add dependencies to pom.xml  
 <properties>

<springframework.version>4.2.5.RELEASE</springframework.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${springframework.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>javax.servlet.jsp-api</artifactId>

<version>2.3.1</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

#video 4 (Spring MVC Hello World Application)

Configure web.xml be defining a servlet with indication to spring configuration file