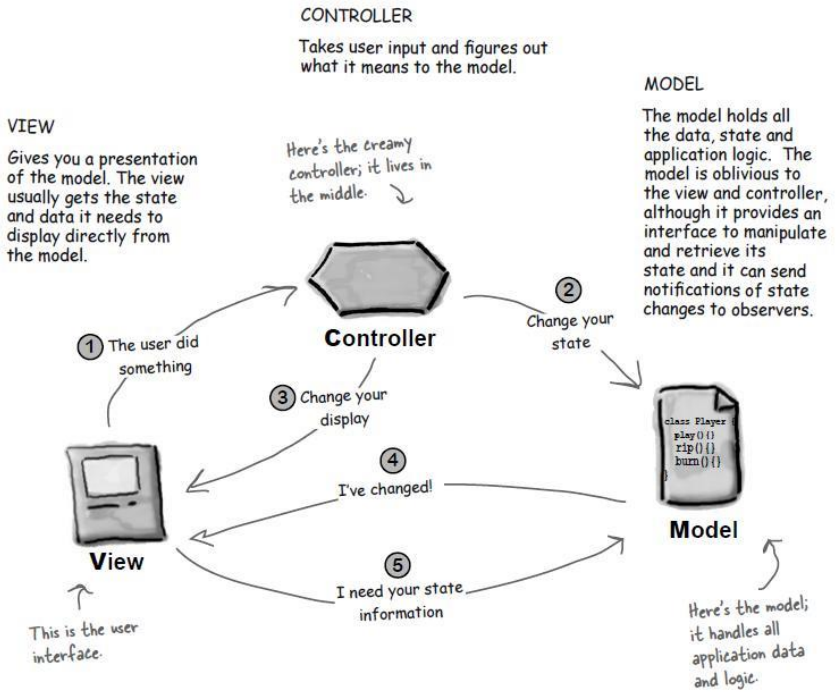
The MVC Pattern

MVC pattern breaks an application into three parts:

- Model: The domain object model / service layer
- View: Template code / markup
- Controller: Presentation logic / action classes

● MVC defines interaction between components to promote separation of concerns and loose coupling

- Enables division of labour between programmers and designers
- Facilitates unit testing
- Easier to understand, change and debug



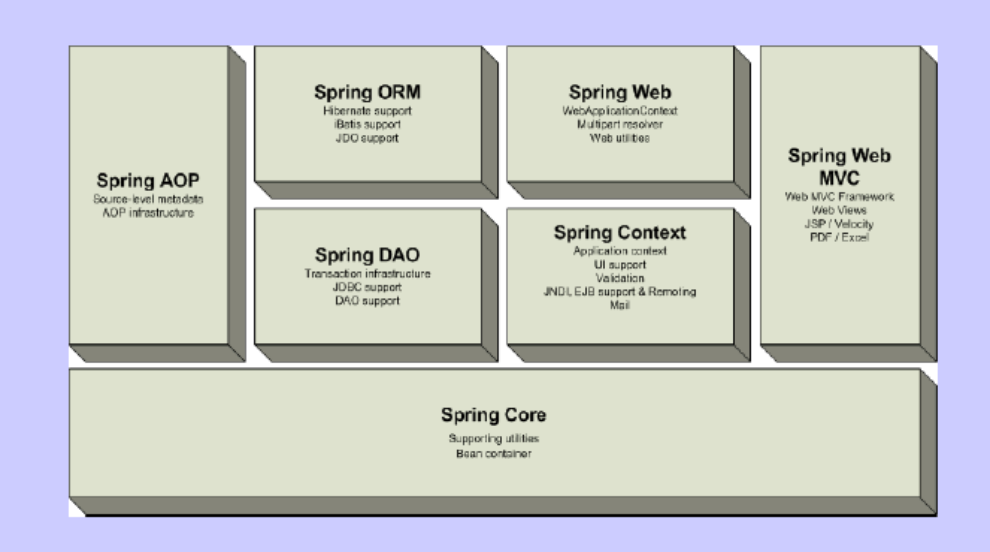
What's Spring MVC?

- A model-view-controller framework for Java web application
- Made to simplify the writing and testing of Java web applications
- Fully integrates with the Spring dependency injection (Inversion of Control) framework
- Open Source Developed and maintained by Interface21,recently purchased by VMWare

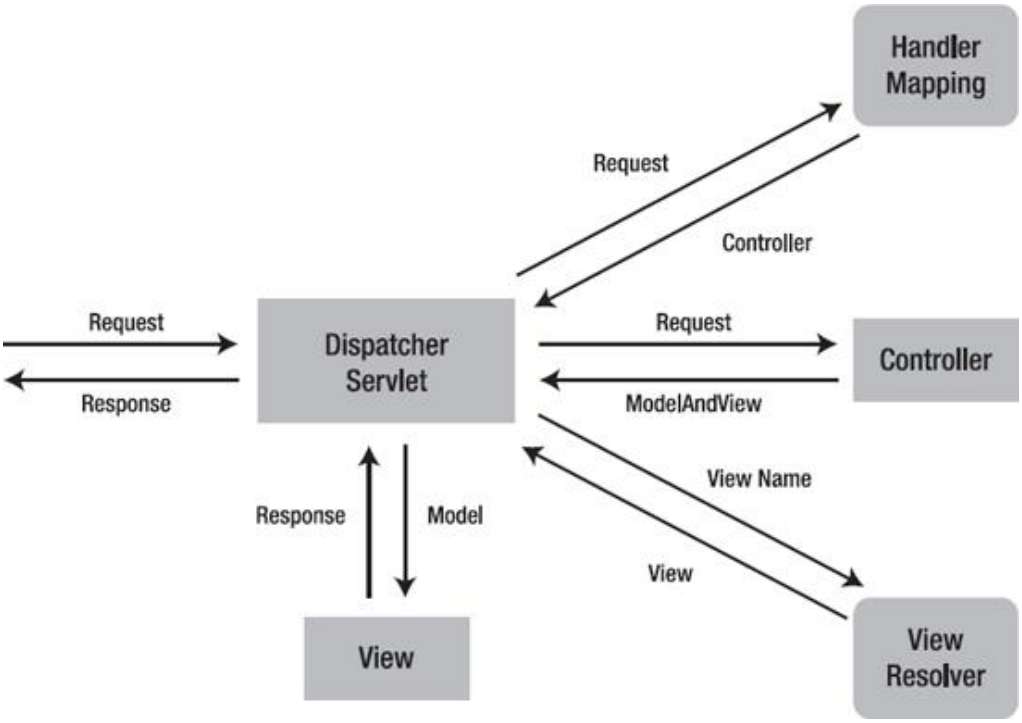
Why Use Spring MVC?

- Easy Testing
- Simple but powerful tag library
- Capable of Convention over Configuration
- Spring supports other view technologies and frameworks
- Normal business objects can be used to back forms
- Very flexible view resolvers

Spring Overview

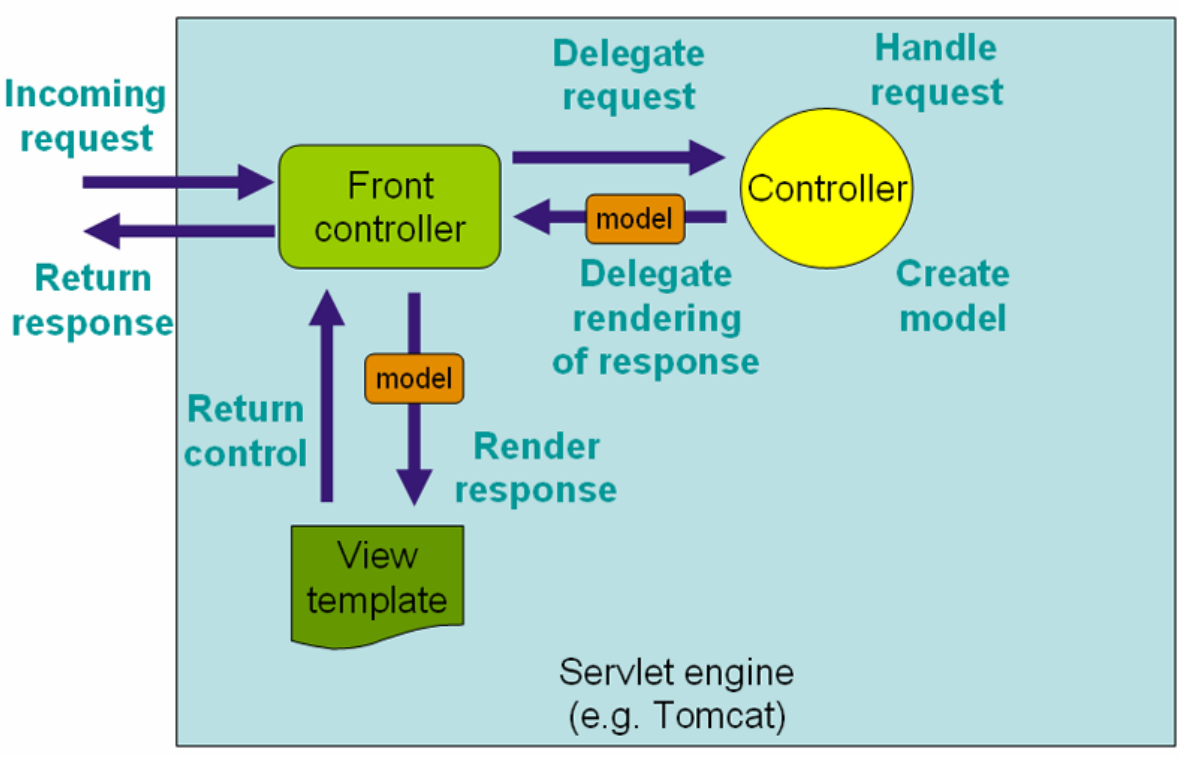


Spring mvc



Dispatcher Servlet:

- Used to handle all incoming requests and route them through Spring
- Uses customizable logic to determine which controllers should handle which requests
- Forwards all responses to through view handlers to determine the correct views to route responses to.
- Exposes all beans defined in Spring to controllers for dependency injection



The requesting processing workflow in Spring Web MVC (high level)

Uses Front Controller Design Pattern

Defining a Dispatcher Servlet

```
<servlet>
    <servlet-name>spring</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>2</load-on-startup>
</servlet>

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

Defining a Dispatcher Servlet named **"spring"** that will intercept all urls to this web application

Handler Mappings

- You can map handlers for incoming HTTP requests in the Spring application context file.
- These handlers are typically controllers that are mapped to partial or complete URLs of incoming requests.

```
<bean id="urlMap"
    class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="urlMap">
        <props>
            <prop key="/signin.htm">signInController</prop>
            <prop key="/signout.htm">signOutController</prop>
        </props>
    </property>
</bean>
```

Controller :

All incoming HTTP requests from a web browser are handled by Controllers.

A *controller*, as the name indicates, controls the view and model by facilitating data exchange between them.

The central components of MVC

- Simply add @Controller annotation to a class
- Use @RequestMapping to map methods to url

```
<beans…>
<context:component-scan base-package="no.uio.inf5750.assignment1"/>
…
</beans>
```

Get all the **@Controller** annotated classes accessible as beans

```
@Controller
public class BaseController {

        @RequestMapping(value="/")
        public String welcome(ModelMap model) {

                model.addAttribute("message", "Whaddap!!");

                //Spring uses InternalResourceViewResolver and return back index.jsp
                return "index";
        }
…
}
```

Model

- Controllers and view share a Java object referred as model, (‘M’ in MVC)

- A model can be of the type Model or can be a Map that can represent the model.

● The view uses this to display dynamic data that has been given by the controller

```
// Controller
@RequestMapping(value = "/{name}", method = RequestMethod.GET)
        public String welcome(@PathVariable String name, ModelMap model) {

        model.addAttribute("message", "Hello " + name);

        return "index";
}
```

```
<%-- VIEW--%>

<html>
   <body>
      <h1>{message}</h1>
   </body>
</html>
```

View Resolvers

Spring uses the notion of view resolvers, which resolve view names to the actual views (enterhours.jsp, for example).We will use Spring's InternalResourceViewResolver class to resolve our view names.

```
<bean id="viewResolver" class="org.springframework.web.
  servlet.view.InternalResourceViewResolver">
  <property name="prefix">
    <value>/WEB-INF/jsp/</value>
  </property>
  <property name="suffix">
    <value>.jsp</value>
  </property>
</bean>
```

View

The view, has no knowledge of the model and business logic and simply renders the data passed to it (as a web page, in our case).