# Проект с офф.сайта

# Теория (нужно еще раз пройтись и зафиксировать)

хорошо объясняется здесь:

<http://www.mkyong.com/spring-mvc/spring-mvc-hello-world-example/>

~~и здесь (типа перевод)~~

<http://javastudy.ru/spring-mvc/spring-mvc-basic/>

public class MyWebApplicationInitializer implements WebApplicationInitializer {

    @Override

    public void onStartup(ServletContext container) {

        ServletRegistration.Dynamic registration = container.addServlet("dispatcher", new DispatcherServlet());

        registration.setLoadOnStartup(1);

        registration.addMapping("/example/\*");

    }

}

<http://javastudy.ru/spring-mvc/spring-mvc-webapplicationcontext/>

не совсем понятно. Проще взглянуть в проект NetBeans, созданный по соотв.разделу (смотри раздел)

ну и tutorialspoint(остальные примеры)

статья на хабре

Офф.сайт - тестирование

# 5. TutorialsPoint

<https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm>

The sequence of events corresponding to an incoming HTTP request to DispatcherServlet:

* After receiving an HTTP request, *DispatcherServlet consults the* ***HandlerMapping*** *to call the appropriate Controller.*
* **The Controller** takes the request and calls the appropriate service methods based on used GET or POST method. The service method will ***set model data*** based on defined business logic and ***returns view name*** to the DispatcherServlet.
* The DispatcherServlet will take help from ViewResolver to pickup the defined view for the request.
* Once view is finalized, The DispatcherServlet passes the model data to the view which is finally rendered on the browser.

All the above mentioned components ie. HandlerMapping, Controller and ViewResolver are parts of **WebApplicationContext** which is an extension of the plain ***ApplicationContext*** with some extra features necessary for web applications.

(см. также предыдущий раздел)

## web.xml

В папке WEB-INF

|  |  |
| --- | --- |
| <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>/WEB-INF/HelloWeb-servlet.xml</param-value>  </context-param>  <listener>  <listener-class>  org.springframework.web.context.ContextLoaderListener  </listener-class>  </listener> | В отличие от проекта по шаблону NetBeans здесь указываем на конфигурационный файл сервлета  Листенер такой же |
| <servlet>  <servlet-name>HelloWeb</servlet-name>  <servlet-class>  org.springframework.web.servlet.DispatcherServlet  </servlet-class>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>HelloWeb</servlet-name>  <url-pattern>**~~\*.jsp~~ /**</url-pattern>  </servlet-mapping> | Имя сервлета здесь задали HelloWeb  Класс сервлета  Маппирование на jsp – косяк! |

## HelloWeb-servlet.xml

|  |  |
| --- | --- |
| <context:component-scan base-package="com.tutorialspoint" />  <bean class=  "org.springframework.web.servlet.view  .InternalResourceViewResolver">  <property name="prefix"  value="/WEB-INF/jsp/" />  <property name="suffix" value=".jsp" />  </bean> | Переименовываем пакет (указываем где искать spring аннотацию)  ViewResolver задан так же (создаём папку jsp)  Смотри описание! |

## HelloController.java

Добавил (add library) Spring framework 4.0.1, чтобы работать с аннотациями

|  |  |
| --- | --- |
| @Controller  public class HelloController{    @RequestMapping(value = **"/hello**", method = RequestMethod.GET)  public String printHello(ModelMap model) {  model.addAttribute("**message**", "Hello Spring MVC Framework!");  return "hello";  }  } |  |

## Hello.jsp

В папке WEB-INF/**hello – это ошибка. Был указан каталог /jsp**

<h2>${**message**}</h2> - переменная указанная в контроллере

# 4. WebApplication в NetBeans

|  |  |
| --- | --- |
|  | JSTL - ? javascript tags library |

|  |  |
| --- | --- |
|  | Имя сервлета-диспетчера и указание как надо представлять jsp страницы для отображения в браузере: страницы с любым именем представляются как файлы .htm.  Проделал эксперимент: заменил в redirect.jsp  <% response.sendRedirect("index.**html**"); %>  И при попытке отобразить в браузере (из вкладки «Servers») - не найдена страница |

## Состав приложения



|  |  |
| --- | --- |
|  | **Web.xml** в Configuration Files и в WEB-INF один и тот же. (ниже описан)  *Конфигурационные файлы Spring*  applicationContext.xml (содержит закомментированный код добавления бинов работы с базой данных и т.д.)  **dispatcher-servlet.xml** (см. подраздел ниже)  *Файлы страниц*  Index.jsp – лежит в WEB-INF/jsp/. Это местоположение определено в резольвере (ViewResolver) в dispatcher-servlet.xml (см.ниже)  Redirect.jsp – перенаправляет на index. Лежит на уровне WEB-INF. Есть [практика](https://www.mkyong.com/spring-mvc/spring-mvc-how-to-include-js-or-css-files-in-a-jsp-page/), что статический контекст так и должен лежать. Хотя это jsp страница с кодом. |

## WEB-INF /web.xml

|  |  |
| --- | --- |
| <context-param>  <param-name>  **contextConfigLocation**  </param-name>  <param-value>  ***/WEB-INF/applicationContext.xml***  </param-value>  </context-param> |  |
| <listener>  <listener-class>  org.springframework.web.context  .**ContextLoaderListener**  </listener-class>  </listener> |  |
| <servlet>  <servlet-name>*dispatcher*</servlet-name>  <servlet-class>  org.springframework.web.servlet.**DispatcherServlet**  </servlet-class>  <load-on-startup>2</load-on-startup>  </servlet> | Указано имя и тип сервлета |
| <servlet-mapping>  <servlet-name>*dispatcher*</servlet-name>  <url-pattern>**\*.htm**</url-pattern>  </servlet-mapping> | Сервлет-диспетчер работает с файлами, заменяя им расширения на htm (файлы с любым названием - \*) |
| …  <welcome-file-list>  <welcome-file>**redirect.jsp**</welcome-file>  </welcome-file-list> | Стартовый html перенаправляющий на указанную в нем страницу |

## dispatcher-servlet.xml

<**bean class="**org.springframework.web.servlet.mvc.support.**ControllerClassNameHandlerMapping"**/>

|  |  |
| --- | --- |
| <!--  *Most controllers will use the ControllerClassNameHandlerMapping above, but*  *for the index controller we are using* ***ParameterizableViewController****, so we must*  *define an explicit mapping for it.*  --> | Большинство контроллеров использует указанный выше мэппинг, но для индекс контроллера мы используем ***ParameterizableViewController*** и поэтому должны определить мэппинг конкретно для него |

<**bean id="urlMapping"** class="org.springframework.web.servlet.handler.**SimpleUrlHandlerMapping**">

<property name="mappings">

<props>

***<prop key="index.htm">indexController</prop>***

</props>

</property>

</bean>

*<!-- The index controller. -->*

<**bean name="indexController"**

**class**="org.springframework.web.servlet.mvc.**ParameterizableViewController**"

**p:viewName="index"** />

|  |  |
| --- | --- |
| <bean id="**viewResolver**"  class="org.springframework.web.servlet  .view.***InternalResourceViewResolver***"  **p:prefix="/WEB-INF/jsp/"**  **p:suffix=".jsp" />** | Для ViewResolver определено место где лежат страницы, и расширение этих страниц |

# 3. Step-by-step – идём дальше!

|  |  |
| --- | --- |
| После перемещения index.jsp в папку «Web pages» при старте проекта браузер неожиданно показал начальную страницу! |  |

|  |  |
| --- | --- |
|  | Файл hello.jsp поместил в эту же папку  ! При запуске указываем этот файл в адресной строке:  [http://localhost:8080/StepByStep/**hello.htm**](http://localhost:8080/StepByStep/hello.htm) |

Не смотря на неказистость результата, произошло очень много:

- при получении запроса на hello.htm сервлет-диспатчер (springapp-servlet.xml) находит для запроса подходящий контролер(HelloController) – это работа механизма **Handler Mapping**.

*<bean name="/hello.htm" class="stepbystep.HelloController"/>*

- контролер в нашем простом примере создает модель с вьюшкой, соответствующей файлу hello.jsp

- скрытый от глаз ViewResolver достает нужное представление (hello.jsp) и он наверно бы его и данными бы наполнил, если бы модель была побогаче.

При этом в web.xml прописано:

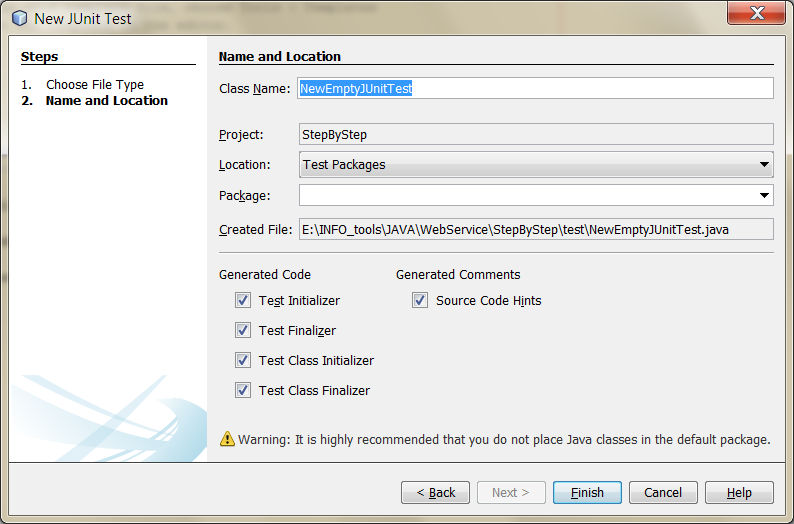
Какой сервлет-диспатчер использовать (<servlet>) и как сопоставлять запросы с сервлетом(<servlet-mapping>), а именно определяет общий паттерн для запросов(<url-pattern>)

# 2. Step-by-step: начало

<https://sites.google.com/site/springmvcnetbeans/step-by-step/#step1.8>

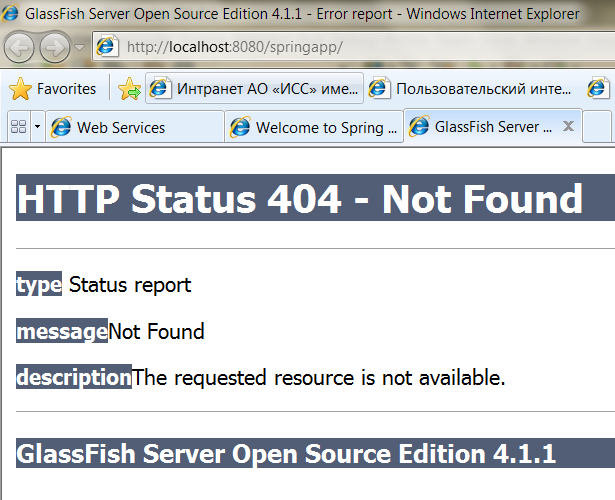
1) Создал stepbystep.controller. HelloController.java

2) Cоздал класс и протестил …



добавляет тестовые библиотеки …

|  |  |
| --- | --- |
|  |  |



# 1. Неудачный пример от Netbeans

<https://netbeans.org/kb/docs/web/quickstart-webapps-spring.html#overview>

Дошел до создания **Simple Form Controller**  и оказалось что он уже не поддерживается..

Перешел на другой пример(step-by-step)