**今天完成的事情**：

**今天尝试搭建MVC模型，模型实例为：Servlet+JSP+JavaBean**

按照Spring MVC的入门实例搭建，第一步是配置Web工程的web.xml文件

1. **web.xml配置文件**

**web.xml配置文件分为几个部分：**

* **文件头**
* **Spring IoC配置**
* **Dispatcher-Servlet配置**

1. **文件头**

我发现建立web项目的时候没法直接建立文件头模版

所以搜索了一些不同版本的文件头，地址如下：

**https://www.cnblogs.com/cowshed/p/7990130.html**

1. **Spring IoC配置**

* **contextConfigLocatiion**

用于定位Spring的IoC控制反转机制的配置路径（依赖注入等功能）

* **contextLoaderListener**

用于初始化Spring IoC容器

1. **Dispatcher-Servlet配置**

**Servlet主要负责调度工作，可以有多个配置，根据拦截器的不同url规则配置，指定访问不同的controller**

Spring MVC框架会根据servlet-name来寻找/WEB-INF/dispatcher-servlet.xml配置文件（注意servler-name和文件名的关系），将其加载入web工程中

* **servlet**

<servlet>标签首先声明一个servlet，包括servlet的名字和对应的java类名

* **servlet-mapping拦截器**

<servlet-mapping>标签声明了与该servlet相应的匹配规则，每个<url-pattern>标签代表1个匹配规则

当浏览器发起一个url请求后，该请求发送到servlet容器的时候，容器先会将请求的url减去当前应用上下文的路径作为servlet的映射url，比如url是http://10.43.11.143/myapp/kata/detail.html，其应用上下文是myapp，容器会将http://10.43.11.143/myapp去掉，剩下的/kata/detail.html部分拿来做servlet的映射匹配。

这个映射匹配过程是有优先顺序的(具体的优先顺序规则后面介绍)，而且当有一个servlet匹配成功以后，就不会去理会剩下的servlet了

1. **代码：TestMVC**
2. **dispatcher-servlet.xml配置**

**dispatcher-servlet.xml配置分为几个部分：**

* **自动扫描@Controller注解，选择controller处理器**
* **使用注解驱动**
* **处理器映射器**
* **处理器适配器**
* **视图解析器**

1. **自动扫描@Controller注解**

<context:component-scan base-package="controller"/>

1. **使用注解驱动**

<mvc:annotation-driven/>

1. **处理器映射器**

处理器接收url请求时，将url请求减去上下文路径后作为servlet的映射路径

<bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"/>

1. **处理器适配器**

<bean class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter"/>

1. **视图解析器**

<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<!-- 找到Web工程/WEB-INF/jsp文件夹，且将结尾为.jsp的文件配置为映射 -->

<!-- jsp文件夹下放置网页文件，将文件夹内的.jsp文件作为视图对象来解析 -->

<property name="prefix" value="/WEB-INF/jsp/"/>

<property name="suffix" value=".jsp"/>

</bean>

1. **Controller.java配置**

Controller控制器负责处理DispatcherServlet分发的请求，把用户请求的数据经过业务处理层处理之后封装成一个Model，再把Model返回给对应的ViewResolver进行展示

**Controller.java主题分为几个部分：**

* **@controller注解**
* **@RequestMapping注解**
* **主体逻辑**

1. **@controller注解**

通过@controller注解方式表示为处理器

1. **@RequestMapping注解**

通过@RequestMapping注解方式定义URL请求和Controller方法之间的映射

@RequestMapping(value = "/a/test",method = RequestMethod.GET)

1. **主体代码逻辑**

以ModelAndView类为模版，建立新类select()

使用ModelAndView类为类型，使用ModelAndView()构造方法，构建对象mv

调用对象方法mv.addObject ()给前台页面传值

调用对象方法mv.setViewName()给

返回mv对象

public ModelAndView select(){

ModelAndView mv= new ModelAndView();

/\* 给前台页面传值 \*/

mv.addObject("msg", "dhkjsahdk");

mv.setViewName("a");

return mv;

}

**明天计划完成的事情：**

今天晚上看一看页面语言，明天尝试构建一个登录页面出来，完成页面跳转，传参接参

**遇到的问题：**

今天花了很多时间来理清代构件的逻辑和跳转的顺序，搞懂每段的代码的意思，下午基本搞懂MVC构件的功能和运行逻辑，稍微落后了计划的进度

**收获：**

理清了MVC的代码逻辑，也搞懂了各构件的功能和作用，对MVC有了整体的理解