SpringMVC

1. **导入Jar包**



1. **在web.xml中配置DisPatcherServlet**

Alt+/ 会出现提示，倒数有一个**dispatcherservle**t，选择后会自动配置，需要写两个地方，黄标位置

<!-- 配置DispatcherServlet，负责流程控制 -->

<servlet>

<servlet-name>springDispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

**配置springMVC配置文件的路径，这里需要修改**

<param-name>contextConfigLocation</param-name>

<param-value>classpath:springmvc.xml</param-value>

</init-param> **classpath表示src下的路径**

**这个param可以省略，默认的路径为**

**“/WEB-INF/<servlet-name>” + “-serlvet.xml”**

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

**url需要写，这里是应答所有请求**

<servlet-name>springDispatcherServlet</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

1. **加入SpringMVC的配置文件**

<context:component-scan base-package=*"springmvc.handlers"*>

</context:component-scan>

**加入开启注解扫描**

**配置视图解析器 ，如何把Handler的方法返回值解析成一个物理视图**

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

<!-- 需要配置两个属性 -->

<!-- **前缀** -->

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

<!-- **后缀** -->

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

</bean>

1. **编写处理请求的处理器，并标识为处理器**

把类标识为@Controller

使用注解映射请求，直接在方法上，无需再写serlvet

**@RequestMapping**("/helloworld")

**public** String hello() {

**return** "success";

}

该方法就可以响应helloworld的请求了，要求这个类必须在IOC容器中，然后会**转发**到到/WEB-INF/views/success.jsp页面

**返回值可以通过视图解析器，解析成实际的物理视图**

视图解析器会把返回值解析成

**前缀+返回值+后缀**

前缀后缀在配置文件中配置

1. **编写视图**