参看博客：<https://blog.csdn.net/eson_15/article/details/51689648>

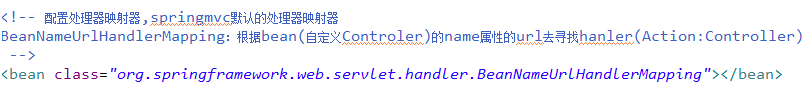
https://blog.csdn.net/eson\_15/article/details/51699103

# 处理器映射器

## BeanNameUrlHandlerMapping

功能：寻找Controller

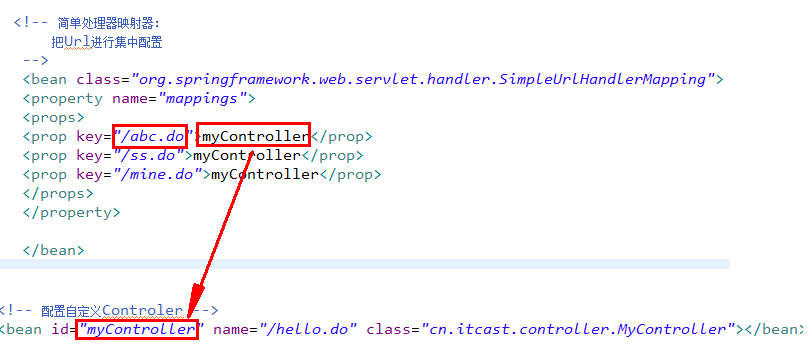
根据url请求去匹配bean的name属性url，从而获取Controller



## SimpleUrlHandlerMaping

功能：寻找Controller

根据浏览器url匹配简单url的key，key又Controller的id找到Controller



## ControllerClassNameHandlerMapping

功能：寻找Controller

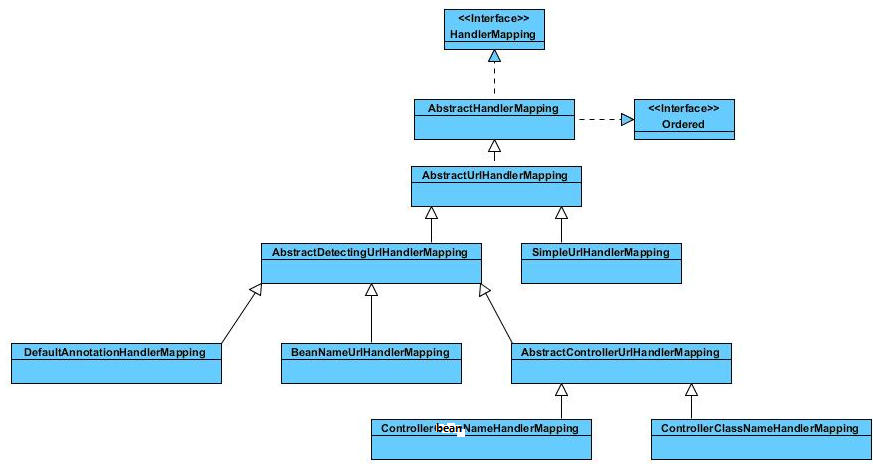
根据类名（MyController）类名.do来访问,类名首字母小写



## 映射器之间能不能共存？

3个处理器映射器可以共存。

## HandlerMapping架构图

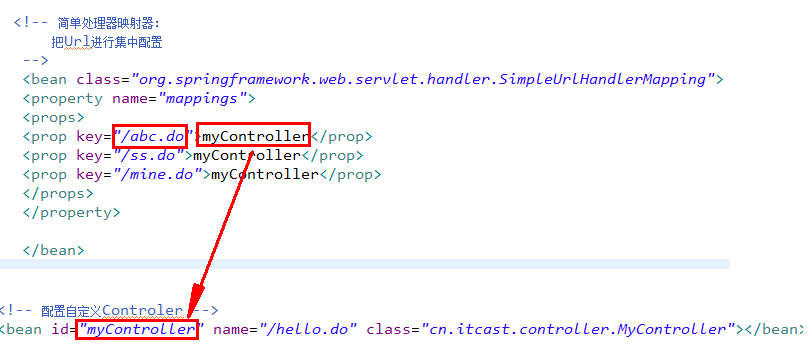


# 处理器适配器

## SimpleControllerHandlerAdapter

功能：执行controller

调用controller里面方法，返回modelAndView。



## HttpRequestHandlerAdapter

功能：执行controller



## 2个处理器适配器能共存？

可以共存

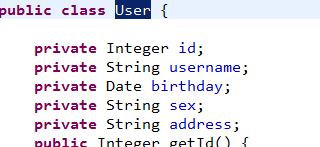
# 命令控制器—封装参数

Springmvc通过命令设计模式接受页面参数。

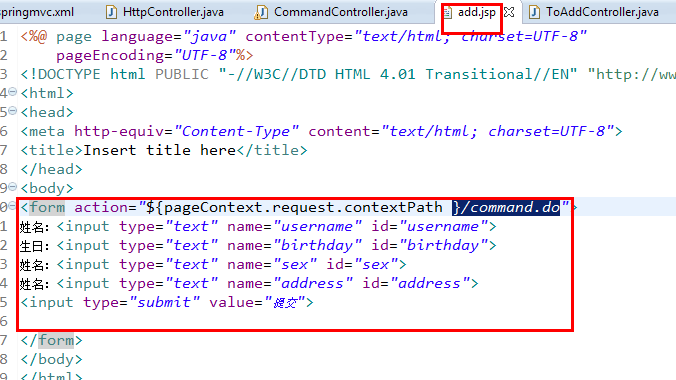
## 自定义命令控制器



## 定义javaBean



## 封装参数页面

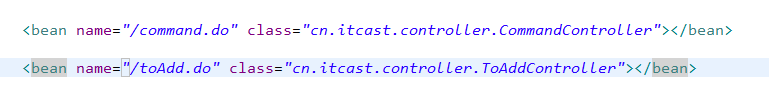


## 跳转到add页面

由于add页面在WEB-INF下面不能直接访问，需要通过Controller来访问。

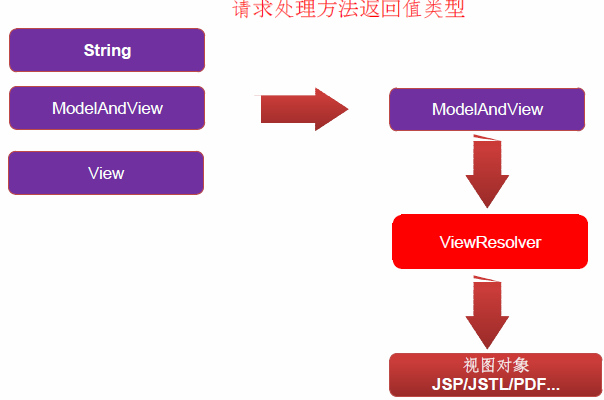


## 在springmvc配置bean



# 视图和视图解析器

## Spring MVC如何解析视图



## 视图

视图的作用是渲染模型数据，将模型里的数据以某种形式呈现给客户。

为了实现视图模型和具体实现技术的解耦，Spring在org.springframework.web

.servlet 包中定义了一个高度抽象的 View接口：



视图对象由视图解析器负责实例化。由于视图是无状态的，所以他们不会有线程安全的问题

常用的视图实现类

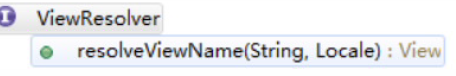


## 视图解析器

SpringMVC 为逻辑视图名的解析提供了不同的策略，可以在 Spring WEB 上下文中配置一种或多种解析策略，并指定他们之间的先后顺序。每一种映射策略对应一个具体的视图解析器实现类。

视图解析器的作用比较单一：将逻辑视图解析为一个具体的视图对象。

所有的视图解析器都必须实现 ViewResolver 接口：



常用的视图解析器实现类



## InternalResourceViewResolver

若项目中使用了 JSTL，则 SpringMVC 会自动把视图由InternalResourceView 转为 JstlView



若使用 JSTL 的 fmt 标签则需要在 SpringMVC 的配置文件中配置国际化资源文件

<!-- 配置国际化资源文件 -->

<bean id=*"messageSource"* class=*"org.springframework.context.support.ResourceBundleMessageSource"*>

<property name=*"basename"* value=*"i18n"*></property>

</bean>

<br>国际化：

<fmt:message key=*"i18n.username"*></fmt:message> <br>

<fmt:message key=*"i18n.password"*></fmt:message> <br>

若希望直接响应通过 SpringMVC 渲染的页面，可以使用 mvc:view-controller 标签实现

<!-- 配置直接转发的页面 -->

<!-- 可以直接相应转发的页面, 而无需再经过 Handler 的方法. -->

<mvc:view-controller path=*"/success"* view-name=*"success"*/>

<!-- 在实际开发中通常都需配置 mvc:annotation-driven 标签 -->

<mvc:annotation-driven></mvc:annotation-driven>

## 自定义视图

@RequestMapping("/testView")

**public** String testView(){

System.*out*.println("testView");

**return** "helloView";

}

<!-- 配置视图 BeanNameViewResolver 解析器: 使用视图的名字来解析视图 -->

<!-- 通过 order 属性来定义视图解析器的优先级, order 值越小优先级越高 -->

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

<property name=*"order"* value=*"100"*></property>

</bean>

@Component

**public** **class** HelloView **implements** View{

@Override

**public** String getContentType() {

**return** "text/html";

}

@Override

**public** **void** render(Map<String, ?> model, HttpServletRequest request,

HttpServletResponse response) **throws** Exception {

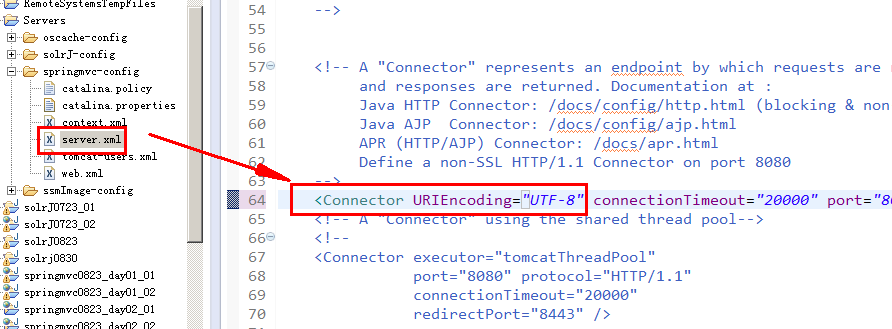
response.getWriter().print("hello view, time: " + **new** Date());

}

}

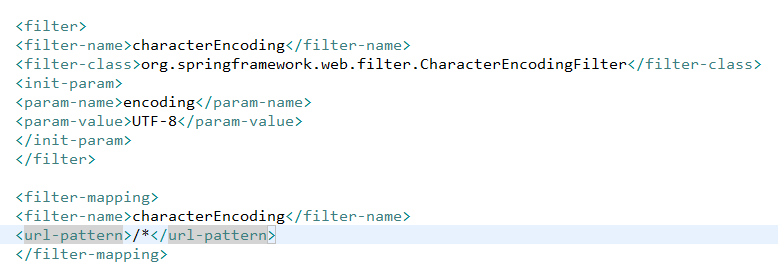
# 中文乱码解决

## Get请求乱码



## Post乱码

Spring编码过滤器：在web.xml配置



# 时间类型转换

