# 1 非注解的映射器和适配器

## 1.1 映射器（HandlerMapping）

- `org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping`
- `org.springframework.web.servlet.handler.SimpleUrlHandlerMapping`

`SimpleUrlHandlerMapping` 是增强版本，可以将 `bean` 的 `id` 进行统一的映射配置

```
<bean class = "org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property>
        <props>
            <prop key = "/simple1.action">sim1</prop>
        </props>
    </property>
</bean>
```

## 1.2 适配器（HandlerAdapter）

- `org.springframework.web.servlet.mvc.SimpleControlleHandlerAdapter`
- `org.springframework.web.servlet.handler.HttpRequestHandlerAdapter`

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

# 2 注解的映射器和适配器

## 2.1 映射器

`RequestMappingHandlerMapping`

- 对类中标记了 `@RequestMapping` 的方法进行映射
- 根据 `RequestMapping` 定义的 `url` 匹配 `@RequestMapping` 标记的方法
- 匹配成功返回 `HandlerMethod` 对象给前端控制器

```
<bean class =
"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping/>

自动声明 Bean ：
<mvc:annotation-driven/>
```

### 2.2 适配器

`RequestMapping, HandlerAdapter`

- 对类中标记了 `@RequestMapping` 的方法进行适配

```
<bean class =
"org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter/>

自动进行适配:
<mvc:annotation-driven></mvc:annotation-driven>
```

## 3 视图解析器

即在解析 `view` 路径时作拼接

默认识别的数据类型:

- `HttpServletRequest`，通过 request 对象获取请求信息
- `HttpServletResponse`，通过 response 处理响应信息
- `HttpSession`，通过session对象得到session中存放的对象
- `Model/ModelMap`，将model数据填充到request域，向页面传递数据

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

## 4 @RequestMapping 注解

- 即请求映射，为控制器指定可以处理的 `URL`
- 可以标记在类定义处和方法定义处
    - 类定义处：提供初步的映射信息，相当于 `Web` 应用根目录
    - 方法处：提供进一步的细分映射信息，相当于类定义处的 `URL`
- 若类定义处未标记 `@RequestMapping`，则方法处标记的 `URL` 相对于 `WEB` 应用的根目录
- `@RequestMapping` 除了使用请求 `URL` 映射请求外，还可以使用请求方法，请求参数，请求头映射请求
- 其中的 `value`、 `method`、 `params`、 `headers` 分别表示请求 `URL`、请求方法、请求参数、请求头的映射条件

# 5 控制器方法

返回值有 `ModelAndView` 、 `String` 、 `void`

- 普通字符串，表示返回的逻辑视图名
- 以 `forward` 开头字符串，表示转发
- 以 `redirect` 开头字符串，表示重定向

```
@RequestMapping(value="/f6")
public void f6(HttpServletRequest req, HttpServletResponse res) {
  req.getRequestDispatcher("/WEB-INF/jsp/1.jsp").forward(req, res);
  req.getRequestDispatcher("f4").forward(req, res);
}
```

# 6 参数绑定注解

## 6.1 Spring MVC 的参数注解

- 位于包 `org.springframework.web.bind.annotation`

根据处理内容的不同分为

- 处理 `request body` 部分的注解， `@RequestParam` 、 `@RequestBody`
- 处理 `request uri` 部分注解， `@PathVariable`
- 处理 `request header` 部分的注解： `@RequestHeader` 、 `@CookieValue`
- 处理 `attribute` 类型的注解： `@SessionAttribute` 、 `@ModelAttribute`

## 6.2 @RequestParam

`@RequestParam` 用于将指定的请求参数赋值给方法在形参中

| 属性 | 类型 | 是否必须 | 说明 |
| --- | --- | --- | --- |
| name | String | 否 | 指定请求头绑定的名称 |
| value | String | 否 | name属性的别名 |
| required | Boolean | 否 | 当前参数是否绑定，默认为true |
| defaultValue | String | 否 | 默认值 |

- 不用 `@RequestParam` ， request 传入的参数名和 `controller` 方法的形参名一致才能绑定成功
- 使用 `@RequestParam` ， request 传入的参数名和 `@RequestParam` 的 `value` 值一致才能绑定成功

`@RequestBody` ： 获取请求传递的参数对

## 6.3 @PathVariable

- 是 Spring3.0 之后加入的，是Spring 支持REST 风格的url 的一个重要标志
- 能够方便的获取 `URL` 中的动态参数，只有一个 `value` 属性
- url 中的占位符，即 `/delete/{id}` ， `id` 为占位符
- `@PathVariable` 标记的参数传入

传统的 `url`

- 添： `http://localhost:8080/addUser` `post`
- 删： `http://localhost:8080/deleteUser?id=1` `get`
- 改： `http://localhost:8080/modifyUser` `post`
- 查： `http://localhost:8080/queryUser?id=1` `get`

`REST` 的 `url`

- 添： `http://localhost:8080/user` `post` 提交
- 删： `http://localhost:8080/user/1` `delete` 提交
- 改： `http://localhost:8080/user` `put` 提交
- 查： `http://localhost:8080/user/1` `get` 提交

## 6.4 @RequestHeader

- 用于将请求头数据映射到处理方法的参数上、获取请求头数据

| 属性 | 类型 | 是否必须 | 说明 |
| --- | --- | --- | --- |
| name | String | 否 | 指定请求头绑定的名称 |
| value | String | 否 | name属性的别名 |
| required | Boolean | 否 | 当前参数是否绑定 |
| defaultValue | String | 否 | 默认值 |

## 6.5 @SessionAttributes

- 默认情况下 `SpringMVC` 将模型中的数据存储到 `request` 域，即当一个请求结束后，数据就失效了，跨请求使用，需要将数据存到 `Session` 中
- `SessionAttributes` 只注解到类上，将指定的 Model 的键值对保存在 `Session` 中

| 属性 | 类型 | 是否必须 | 说明 |
| --- | --- | --- | --- |
| names | String[] | 否 | Model 中属性的别称，即在HttpSession存储的属性名称 |
| value | String[] | 否 | names属性别名 |
| types | Class<?>[] | 否 | 存储在HttpSession中的对象类型，所指定类型的实例均保存到HttpSession对象中 |

- `Model` 默认的作用范围与 `Request` 相同，当 `Request` 请求失效时，`Model` 中的数据也被清空
- 使用 `@SessionAttributes` 将`Model` 中数据转移到 `SessionAttributes` 中时，`Model` 的作用范围与`SessionAttributes` 作用范围相同，直到 `sessionStatus.setComplete()` 方法清除数据后，`Model` 数据才被清空

### 6.6 @ModelAttribute

- 用于将请求参数绑定到 Model 对象
- 使用 `@ModelAttribute` 注解的方法将在`Controller` 的请求处理方法之前被调用
- 修饰参数时，表示给该参数赋值

## 7 过滤器

- 过滤器解决中文乱码问题

```
<filter>
      <filter-name>CharacterEncodingFilter</filter-name>
      <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
      <init-param>
          <param-name>encoding</param-name>
          <param-value>UTF-8</param-value>
      </init-param>
      <init-param>
          <param-name>forceEncoding</param-name>
          <param-value>true</param-value>
      </init-param>
  </filter>
  <filter-mapping>
      <filter-name>CharacterEncodingFilter</filter-name>
      <url-pattern>/*</url-pattern>
  </filter-mapping>
```

# 8 参数绑定

## 8.1 数组

- 在 jsp 页面使用进行绑定时直接使用数组名即可，不要用 `[]`，否则无效

## 8.2 集合

- `List`，需要绑定在 pojo 上，不能直接写在 Controller 方法的参数中
- `Map`，需要绑定在 pojo 上，不能直接写在 Controller 方法的参数中，使用 `[]` 访问 key