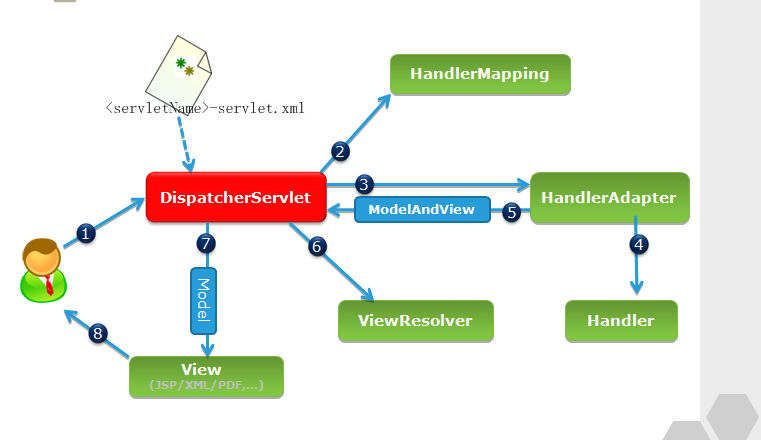
SpringMVC 框架

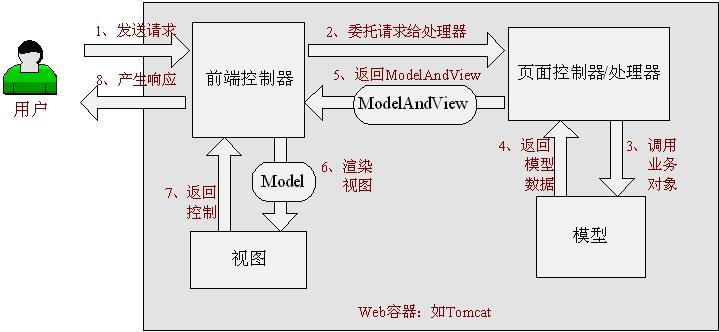
# Spring MVC框架简介

## 1、Spring MVC 3.0新特性

* 添加更多注解，可完全注解驱动
* 引入HTTP输入输出转换器（HttpMessageConverter）
* 和数据转换、格式化、验证框架无缝集成
* 对静态资源处理提供特殊支持
* 更加灵活的控制器方法签名，可完全独立于Servlet API

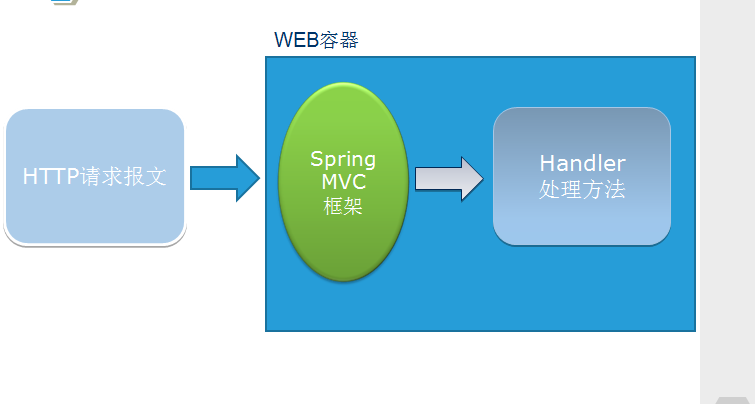
## 2、Spring MVC框架结构





# HTTP请求地址映射

## 1、HTTP请求映射原理





## 2、通过URL限定:URL表达式

@RequestMapping不但支持标准的URL，还支持Ant风格（即?、\*和\*\*的字符，参见3.3.2节的内容）的和带{xxx}占位符的URL。以下URL都是合法的：

* /user/\*/createUser

匹配/user/aaa/createUser、/user/bbb/createUser等URL。

* /user/\*\*/createUser

匹配/user/createUser、/user/aaa/bbb/createUser等URL。

* /user/createUser??

匹配/user/createUseraa、/user/createUserbb等URL。

* /user/{userId}

匹配user/123、user/abc等URL。

* /user/\*\*/{userId}

匹配user/aaa/bbb/123、user/aaa/456等URL。

* company/{companyId}/user/{userId}/detail

匹配company/123/user/456/detail等的URL。

## 3、通过请求方法限定:请求方法

请求方法，在HTTP中这被叫做动词（verb），除了两个大家熟知的（GET和POST）之外，标准方法集合中还包含PUT、DELETE、HEAD和OPTIONS。这些方法的含义连同行为许诺都一起定义在HTTP规范之中。一般浏览器只支持GET和POST方法。

|  |  |  |
| --- | --- | --- |
| 序号 | 请求方法 | 说明 |
| 1 | GET | 使用GET方法检索一个表述（representation）——也就是对资源的描述。多次执行同一GET请求，不会对系统造成影响，GET方法具有幂等性[指多个相同请求返回相同的结果]。GET请求可以充分使用客户端的缓存。 |
| 2 | POST | POST方法，通常表示“创建一个新资源”，但它既不安全也不具有幂等性（多次操作会产生多个新资源）。 |
| 3 | DELETE | DELETE，表示删除一个资源，你也可以一遍又一遍地操作它，直到得出结果：删除不存在的东西没有任何问题 |
| 4 | PUT | 幂等性同样适用于PUT（基本的含义是“更新资源数据，如果资源不存在的话，则根据此URI创建一个新的资源”） |

**代码示例：**

## 4、请求头参数限定

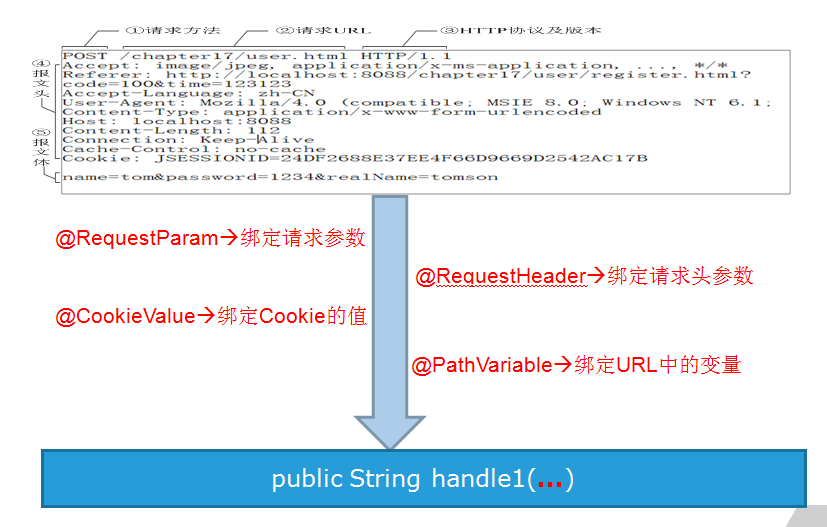
params和headers分别通过请求参数及报文头属性进行映射，它们支持简单的表达式，下面以params表达式为例说明，headers可以参照params进行理解之。

* **"param1"：**表示请求必须包含名为param1的请求参数。
* **"!param1"：**表示请求不能包含名为param1的请求参数。
* **"param1!=value1"：**表示请求包含名为param1的请求参数，但其值不能为value1。
* **{"param1=value1","param2"}：**请求必须包含名为param1和param2的两个请求参数，且param1参数的值必须为value1。

# 三、HTTP请求数据的绑定

## 1、通过路径参数限定：

## 2、通过注解绑定



@RequestParam有以下三个参数。

* value：参数名。
* required：是否必需，默认为true，表示请求中必须包含对应的参数名，如果不存在将抛出异常。
* defaultValue：默认参数名，设置该参数时，自动将required设为false

上面的处理方法 ，如果HTTP请求不包含“userName”参数时，将产生异常！！  
 因此，如果不能保证存在”userName”的参数，必须使用：

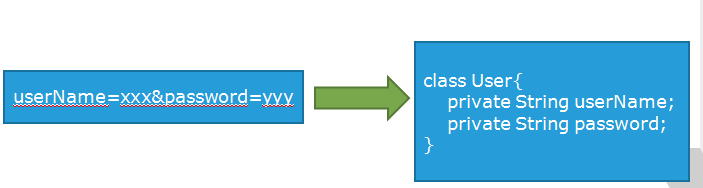
@RequestParam(value = "userName", **required = false**)

## 3、使用命令/表单对象绑定

所谓命令/表单对象并不需要实现任何接口，仅是一个拥有若干属性的POJO。Spring MVC按：

**“HTTP请求参数名 = 命令/表单对象的属性名”**

的规则，自动绑定请求数据，支持“级联属性名”，自动进行基本类型数据转换。



## 4、使用Servlet API对象作为入参

## 5、使用Spring的Servlet API代理类

Spring MVC在org.springframework.web.context.request包中定义了若干个可代理Servlet原生API类的接口，如WebRequest和NativeWebRequest，它们也允许作为处理类的入参，通过这些代理类可访问请求对象的任何信息。

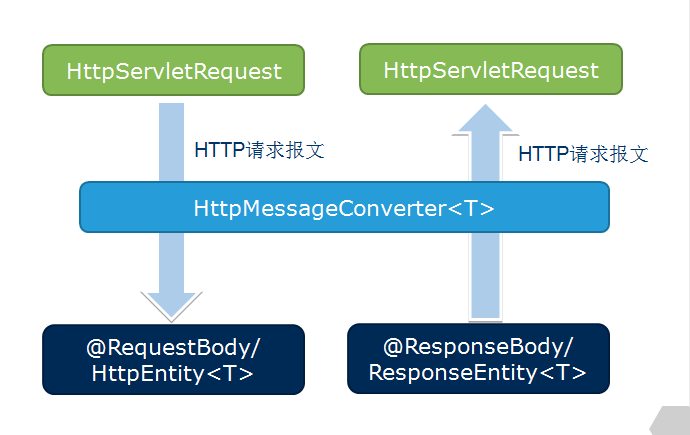
## 6、使用IO对象作为入参

Spring MVC允许控制器的处理方法使用java.io.InputStream/java.io.Reader及java.io.OutputStream/java.io.Writer作为方法的入参

Spring MVC将获取ServletRequest的InputStream/Reader或ServletResponse的OutputStream/Writer，然后按类型匹配的方式，传递给控制器的处理方法入参。

# 数据转换、格式化、校验

## 消息转化器



### 使用@RequestBody/@ResponseBody

将HttpServletRequest的getInputStream()内容绑定到入参，将处理方法返回值写入到HttpServletResponse的getOutputStream()中。

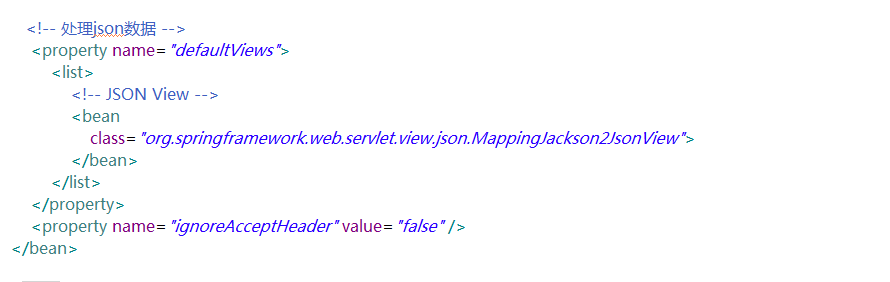
特点：

优点：处理方法签名灵活不受限

缺点：只能访问报文体，不能访问报文头

### 输出XML和JSON

1>需要在SpringMVC配置文件添加json视图解析器

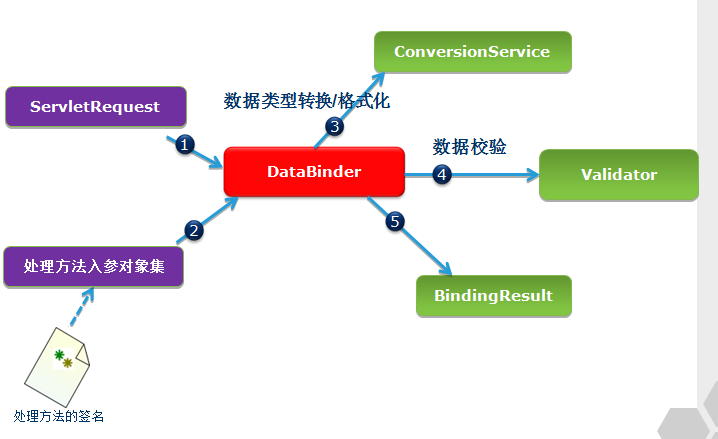


2>在controller的请求方法中返回ResponseEntity对象

对于服务端的处理方法而言，除使用@RequestBody/@ResponseBody或HttpEntity<T> /ResponseEntity<T>进行方法签名外，不需要进行任何额外的处理，借由Spring MVC中装配的HttpMessageConverter，它即拥有了处理XML及JSON的能力了。

## 2、数据绑定

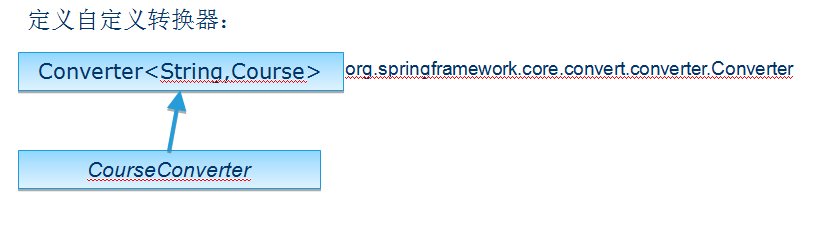
### 1、数据绑定机理



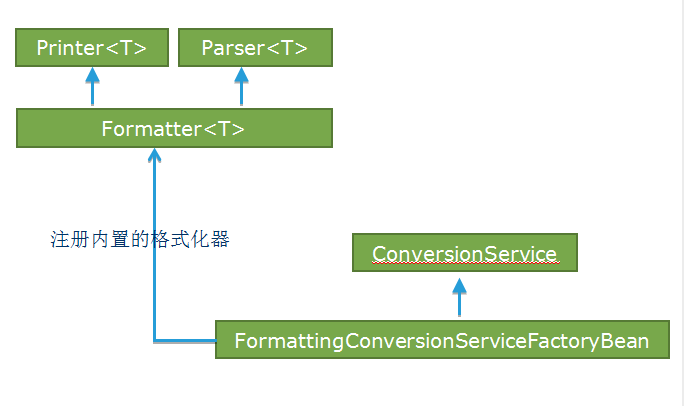
由于ConversionService在进行类型转换时，可以使用到Bean所在宿主类的上下文信息（包括类结构，注解信息），所以可以实施更加高级的类型转换，如注解驱动的格式化等功能。

以上User类，通过一个@DateTimeFormat注解，为类型转换提供了一些“额外”的信息，即代表日期的“源字符器”格式是“yyyy-MM-dd”

### 2、自定义的类型转换器



### 格式化



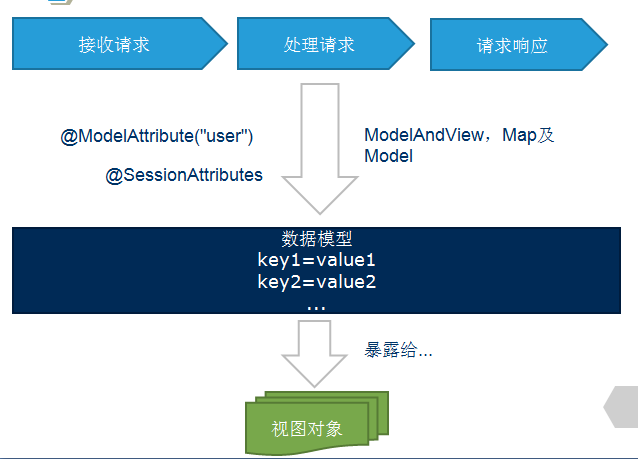
### 使用支持格式化的转换器

值得注解的是，<mvc:annotation-driven/>标签内部默认创建的ConversionService实例就是一个FormattingConversionServiceFactoryBean，自动支持如下的格式化注解：

* @NumberFormatter：用于数字类型对象的格式化。
* @CurrencyFormatter：用于货币类型对象的格式化。
* @PercentFormatter：用于百分数数字类型对象的格式化。

# 数据模型控制

## 1、数据模型访问结构



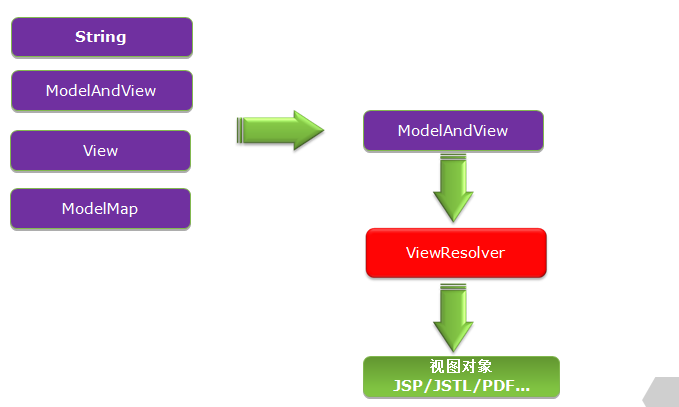
## 通过ModelAndView

## 3、通过Map及Model

Spring MVC一旦发现处理方法有Map或Model类型的入参，就会将请求内在的隐含模型对象的引用传给这些入参。

# 视图及解析器

## 解析器原理



## 视图解析器类型

### 1、完成单一功能的视图解析器

如下：

* InternalResourceViewResolver
* FreeMarkerViewResolver
* BeanNameViewResolver
* XmlViewResolver

### 基于协商的视图解析器

* ContentNegotiatingViewResolver

该解析器是Spring 3.0新增的，它不负责具体的视图解析，而是作为一个中间人的角色根据请求所要求的MIME类型，从上下文中选择一个适合的视图解析器，再将视图解析工作委托其负责。

# 静态资源处理

## 使REST风格的URL成为实现

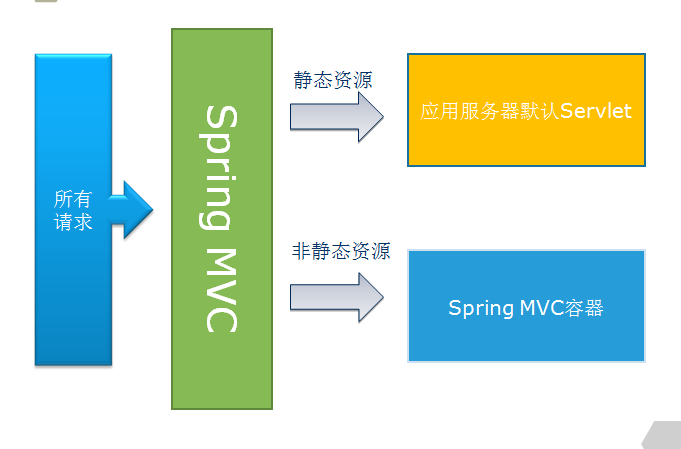
优雅REST风格的资源URL不希望带.html或.do等后缀，以下是几个优雅的URL。

⦁ /blog/tom：用户tom的blog资源。

⦁ /forum/java：java论坛板块资源。

⦁ /order/4321：订单号为4321的订单资源；

## 处理原理



1. 配置方式

第一步：web.xml让所有请求都由Spring MVC处理

第二步： mvc-dispatcher-servlet.xml 让Web应用服务器处理静态资源

实例：

物理静态资源路径映射逻辑资源路径

