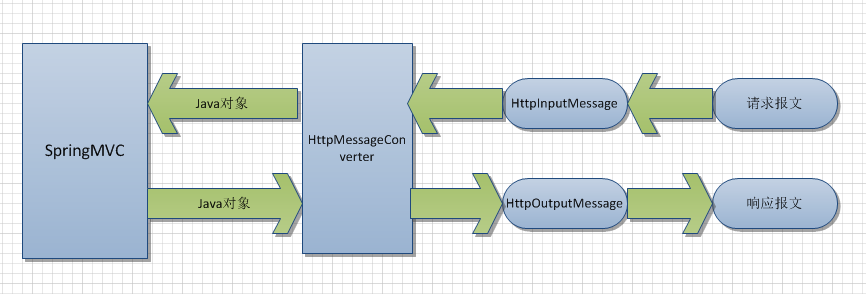
# 消息转换器HttpMessageConverter



## SpringMVC消息转换机制

1. 在SpringMVC中，可以使用**@RequestBody**和**@ResponseBody**两个注解，分别完成请求报文到对象和对象到响应报文的转换，底层这种灵活的消息转换机制，就是Spring3.x中新引入的HttpMessageConverter即消息转换器机制。
2. @RequestBody作用：用于读取Http Request中Body部分的数据，使用系统默认配置的消息转换器HttpMessageConverter进行解析，染回将数据绑定在要返回的对象上。消息转换器HttpMessageConverter再将返回对象的数据绑定在controller方法的参数上。
3. @ResposeBody作用：将Controller方法返回的对象，通过适当的HttpMessageConverter转换成指定的格式后，然后写入到Respose返回对象的正文当中，不会再经过视图解析器的处理。

## Http请求的抽象

* 1. 请求体的解析

当web应用接收到Http请求的时候，在servlet标准中，可以用ServletRequest接口中的以下方法：



来得到一个ServletInputStream。这个ServletInputStream中，可以读取到一个原始请求报文的所有内容。

### 响应报文

在web应用中，可以通过ServletResponse接口中的方法：



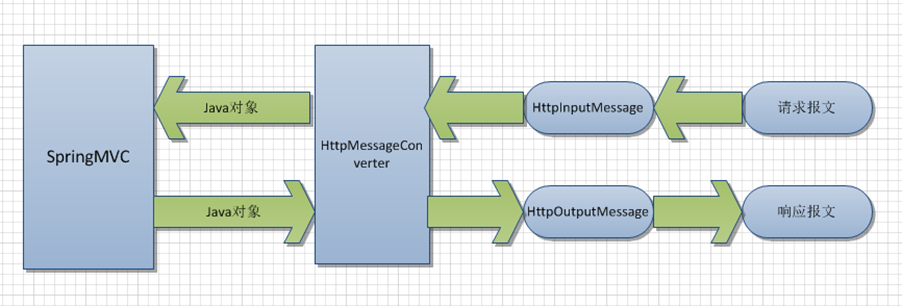
来得到一个ServletOutputStream，这个ServletOutputSteam，继承自java中的OutputStream，可以让你输出Http的响应报文内容。

### 需要解决的问题

在报文到达SpringMVC和从SpringMVC出去，都存在一个字符串到java对象的阻抗问题，需要框架来进行处理。

1. 在Struts2中，采用了OGNL来解决这个问题。
2. 在SpringMVC中，它是HttpMessageConverter机制。

## HttpMessageConverter消息转换器接口

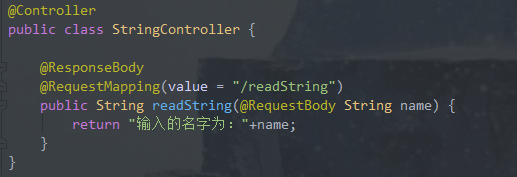


### 基本处理逻辑

(HttpMessageConverter接口)

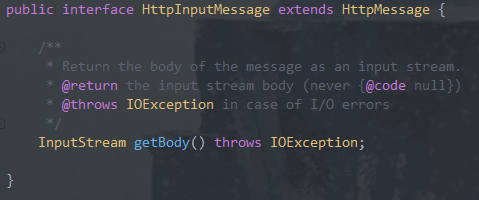


(Controller方法实现)



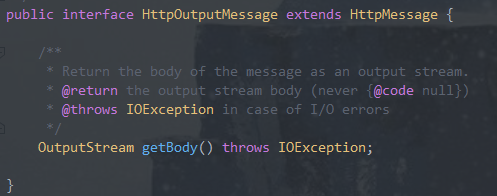
1. 在SpringMVC进入readString方法前，会根据@RequestBody注解选择适当的HttpMessageConverter实现类来将请求参数解析到string变量中，具体来说是使用了StringHttpMessageConverter类。它的canRead()方法返回 true，然后它的read()方法会从请求中读出请求参数，绑定到readString()方法的string变量中。
2. **read()方法使用到HttpInputMessage进行从http请求中读取参数的处理。**
3. 当SpringMVC执行readString方法后，由于返回值标识了@ResponseBody，SpringMVC将使用 StringHttpMessageConverter的write()方法，将结果作为String值写入响应报文，当然，此时canWrite()方 法返回true。
4. **write()方法使用到HttpOutputMessage进行将结果写入http请求中的处理。**

### HttpInputMessage接口



作用：SpringMVC的消息转换器所作用的受体“请求消息”的内部抽象，消息转换器从“请求消息”中按照规则提取消息，转换为方法形参中声明的对象。

### HttpOutputMessage接口

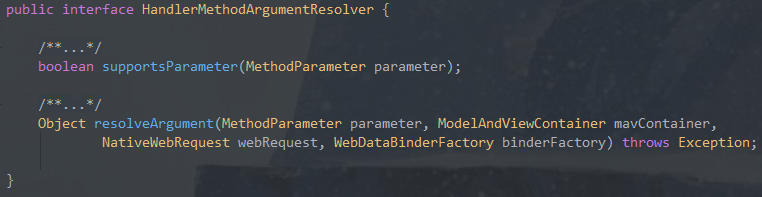


作用：SpringMVC的消息转换器所作用的受体“响应消息”的内部抽象，消息转换器将“响应消息”按照一 定的规则写到响应报文中。

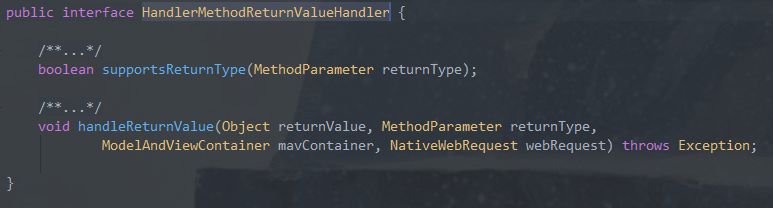
## RequestResponseBodyMethodProcessor类

### 作用

1. 充当了方法参数解析和返回值处理两种角色，是将消息转换过程集中体现的一个类。
2. 实现了接口HandlerMethodArgumentResolver，是一个可以将请求报文绑定到处理方法形参的策略接口。



1. 实现了接口HandlerMethodReturnValueHandler，是一个对处理方法返回值进行处理的策略接口。



1. 两个接口的实现，分别是以是否有@RequestBody和@ResponseBody为条件，然后分别调用HttpMessageConverter来进行消息的读写。

## SpirngMVC初始化7个消息转换器

### 5.1 消息转换器的注册



标签<mvc:annotation-driven/>作用

1. 会自动注册RequestMappingHandlerMapping、RequestMappingHandlerAdapter、ExceptionHandlerExceptionResolver三个bean支持使用了像@RquestMapping、ExceptionHandler等等的注解的controller 方法去处理请求。
2. 支持使用了ConversionService]的实例对表单参数进行类型转换。
3. 支持使用@NumberFormat、@NumberFormat注解对数据类型进行格式化。
4. 支持使用@Valid对javaBean进行JSR-303验证。
5. 支持使用@RequestBody、@ResponseBody。

### 5.2 初始化的7个消息转换器

* ByteArrayHttpMessageConverter
* StringHttpMessageConverter
* ResourceHttpMessageConverter
* SourceHttpMessageConverter
* XmlAwareFormHttpMessageConverter
* Jaxb2RootElementHttpMessageConverter
* MappingJacksonHttpMessageConverter

解析JSON数据

注册消息转换器一般分为2个步骤

1. 导入响应的处理包



1. 在web.xml中开启



## 消息转换器使用步骤

1. 定义一个消息转换器，使用自带的或者自定义都可以。
2. 向springMVC注册消息转换器，可以通过xml和配置类两种方式。
3. 使用@RequestBody或者@ResponseBody注解，使用消息转换器。

新建一个配置类，实现WebMvcConfigurer接口或者继承WebMvcConfigurationSupport

