SpringMVC

回顾:

mvc:annotation-driven/是告知Spring，启用注解驱动。然后Spring会自动为我们注册多个Bean到工厂中，来处理我们的请求。

主要有两个：RequestMappingHandlerMapping RequestMappingHandlerAdapter 第一个是HandlerMapping的实现类，它会处理@RequestMapping 注解，并将其注册到请求映射表中。 第二个是HandlerAdapter的实现类，它是处理请求的适配器，就是确定调用哪个类的哪个方法，并且构造方法参数，返回值。 --支持使用 @RequestBody 和 @ResponseBody 注解

@RequestBody 将HTTP请求正文转换为适合的**HttpMessageConverter**对象。 **@ResponseBody 将内容或对象作为 HTTP 响应正文返回，**

@RequestBody

@RequestBody是作用在形参列表上，用于将前台发送过来固定格式的数据【xml 格式或者 json等】封装为对应的 JavaBean 对象，封装时使用到的一个对象是系统默认配置的 HttpMessageConverter进行解析，然后封装到形参上。

```
  $.ajax({
              url:"user/login",
              type:"POST",
              data:'{"userName":"admin","pwd","admin123"}',
              content-type:"application/json charset=utf-8",
              success:function(data){
                  alert(data);
              }
       });
```

```
@RequestMapping("/login")
public void login(@RequestBody String userName,@RequestBody String pwd){
    System.out.println(userName+" : "+pwd);
}
```

这种情况是将JSON字符串中的两个变量的值分别赋予了两个字符串，**(不是很常用)**

@ResponseBody

produces={"application/json;charset=UTF-8"} 用于处理响应请求编码（建议改为application/text)

```
@ResponseBody
@RequestMapping(value="t1",produces="application/json;charset=UTF-8")
public String t1(){
    return "Hello ResponseBody ";
}
```

@ResponseBody是作用在方法上的，@ResponseBody 表示该方法的返回结果直接写入 HTTP response body 中，一般在异步获取数据时使用【也就是AJAX】，

在使用 @RequestMapping后，返回值通常解析为跳转路径，但是加上 @ResponseBody 后返回结果不会被解析为跳转路径，而是直接写入 HTTP response body 中。 比如异步获取 json 数据，加上 @ResponseBody 后，会直接返回 json 数据。

@RequestBody 将 HTTP 请求正文插入方法中，使用适合的 HttpMessageConverter 将请求体写入某个对象。

HttpMessageConverter

Spring 3.X系列增加了新注解@ResponseBody，@RequestBody HttpMessageConverter接口，需要开启 <mvc:annotation-driven /> AnnotationMethodHandlerAdapter将会初始化7个转换器，

ByteArrayHttpMessageConverter

StringHttpMessageConverter

ResourceHttpMessageConverter

SourceHttpMessageConverter

XmlAwareFormHttpMessageConverter

Jaxb2RootElementHttpMessageConverter

MappingJacksonHttpMessageConverter

只要有对应协议的解析器，就可以通过注解完成协议——对象的转换工作

编写对应消息解析器

导入相应JAR (Spring默认的json协议解析由Jackson完成加入jar包)

```
jackson-annotations-2.8.7.jar
jackson-core-2.8.7.jar
jackson-databind-2.8.7.jar
```

编写目标方法，使其返回 JSON 对应的对象或集合

在方法上添加 @ResponseBody 注解

```
@ResponseBody
@RequestMapping("t2")
public List<User> t2(){
    List<User> list=userService.selectList();
    return list;
}
```

Jackson JSON操作

Jackson ObjectMapper类用于Java对象与JSON的互换

```
//Jackson中有个ObjectMapper类很是实用，用于Java对象与JSON的互换
/*
 * writeValue(File arg0, Object arg1)把arg1转成json序列，并保存到arg0文件中。
 * writeValue(OutputStream arg0, Object arg1)把arg1转成json序列，并保存到arg0输出流中。
 * writeValueAsBytes(Object arg0)把arg0转成json序列，并把结果输出成字节数组。
 * writeValueAsString(Object arg0)把arg0转成json序列，并把结果输出成字符串。
 */
List<Users> list=new ArrayList<Users>();
list.add(new Users("guo","1"));
list.add(new Users("guo1","李四1"));

ObjectMapper mapper=new ObjectMapper();
//集合转JSON字符串
String listJson=mapper.writeValueAsString(list);
//对象转JSON字符串
Users u=new Users("guo3","李四3");
String objJson=mapper.writeValueAsString(u);
//返序列化 将JSON转对象
Users u2=mapper.readValue(objJson,Users.class);
//返序列化 将JSON转集合
List<Users> list1=mapper.readValue(listJson,ArrayList.class);
```