**SpringBoot 使用@RestController返回JSON数据事项**

1. [直接使用@RestController](mailto:1.直接使用@RestController)有时可以之际返回JSON格式。（暂时没找到异常情况）
2. [使用@RestController](mailto:2.使用@RestController)返回JSON数据报异常：

org.springframework.http.converter.HttpMessageNotReadableException

解决方案如下：

# [springboot自定义消息转换器HttpMessageConverter](https://www.cnblogs.com/hhhshct/p/9676604.html)

　　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利用HttpMessageConverter来实现的，Spring内置了很多HttpMessageConverter，比如MappingJackson2HttpMessageConverter，StringHttpMessageConverter等，下面我们来自定义自己的消息转换器来满足自己特定的需求，有两种方式：1、使用spring或者第三方提供的现成的HttpMessageConverter，2、自己重写一个HttpMessageConverter。

# 配置使用FastJson插件返回json数据

　　在springboot项目里当我们在控制器类上加上@RestController注解或者其内的方法上加入@ResponseBody注解后，默认会使用jackson插件来返回json数据，下面我们利用fastjson为我们提供的FastJsonHttpMessageConverter来返回json数据。

　　首先要引入fastjson的依赖：

　　　　　<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

<version>1.2.31</version>

</dependency>

　　接下来通过实现WebMvcConfigurer接口来配置FastJsonHttpMessageConverter，springboot2.0版本以后推荐使用这种方式来进行web配置，这样不会覆盖掉springboot的一些默认配置。配置类如下：

[复制代码](javascript:void(0);)

package com.example.demo;

import java.util.List;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.converter.HttpMessageConverter;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import com.alibaba.fastjson.serializer.SerializerFeature;

import com.alibaba.fastjson.support.config.FastJsonConfig;

import com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter;

@Configuration

public class MyWebmvcConfiguration implements WebMvcConfigurer{

@Override

public void extendMessageConverters(List<HttpMessageConverter<?>> converters) {

FastJsonHttpMessageConverter fjc = new FastJsonHttpMessageConverter();

FastJsonConfig fj = new FastJsonConfig();

fj.setSerializerFeatures(SerializerFeature.DisableCircularReferenceDetect);

fjc.setFastJsonConfig(fj);

converters.add(fjc);

}

}

[复制代码](javascript:void(0);)

　　fastJson配置实体调用setSerializerFeatures方法可以配置多个过滤方式，常用的如下：

　　1、WriteNullListAsEmpty  ：List字段如果为null,输出为[],而非null  
　　2、WriteNullStringAsEmpty ： 字符类型字段如果为null,输出为"",而非null  
　　3、DisableCircularReferenceDetect ：消除对同一对象循环引用的问题，默认为false（如果不配置有可能会进入死循环）  
　　4、WriteNullBooleanAsFalse：Boolean字段如果为null,输出为false,而非null  
　　5、WriteMapNullValue：是否输出值为null的字段,默认为false。

　　其它的相关类，我们引入了lombok插件

[复制代码](javascript:void(0);)

package com.example.demo;

import java.util.ArrayList;

import java.util.List;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class UserController {

@RequestMapping(value="/get",method=RequestMethod.GET)

public Object getList(){

List<UserEntity> list= new ArrayList<UserEntity>();

UserEntity u1 = new UserEntity(null, "shanghai");

list.add(u1);

return list;

}

}

[复制代码](javascript:void(0);)

[复制代码](javascript:void(0);)

package com.example.demo;

import lombok.AllArgsConstructor;

import lombok.Data;

@Data

@AllArgsConstructor

public class UserEntity {

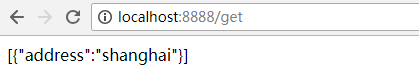
private String name;

private String address;

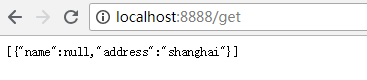
}

[复制代码](javascript:void(0);)

　　设置端口为8888，启动项目访问http://localhost:8888/get，我们代码中没有配置WriteMapNullValue，所以如果返回结果中有null值则不显示，结果如下：



　　我们注释掉fastjson配置，重新启动项目并访问，从结果可以看出我们配置的消息转换器起作用了。



# 重写HttpMessageConverter

　　接下来我们继承AbstractHttpMessageConverter来实现一个自己的消息转换器，示例如下：

[复制代码](javascript:void(0);)

package com.example.demo;

import org.springframework.http.HttpInputMessage;

import org.springframework.http.HttpOutputMessage;

import org.springframework.http.MediaType;

import org.springframework.http.converter.AbstractHttpMessageConverter;

import org.springframework.http.converter.HttpMessageNotReadableException;

import org.springframework.http.converter.HttpMessageNotWritableException;

import org.springframework.util.StreamUtils;

import java.io.IOException;

import java.nio.charset.Charset;

public class MyMessageConverter extends AbstractHttpMessageConverter<UserEntity> {

public MyMessageConverter() {

// 新建一个我们自定义的媒体类型application/xxx-junlin

super(new MediaType("application", "xxx-junlin", Charset.forName("UTF-8")));

}

@Override

protected boolean supports(Class<?> clazz) {

// 表明只处理UserEntity类型的参数。

return UserEntity.class.isAssignableFrom(clazz);

}

/\*\*

\* 重写readlntenal 方法，处理请求的数据。代码表明我们处理由“-”隔开的数据，并转成 UserEntity类型的对象。

\*/

@Override

protected UserEntity readInternal(Class<? extends UserEntity> clazz,

HttpInputMessage inputMessage) throws IOException,

HttpMessageNotReadableException {

String temp = StreamUtils.copyToString(inputMessage.getBody(), Charset.forName("UTF-8"));

String[] tempArr = temp.split("-");

return new UserEntity(tempArr[0],tempArr[1]);

}

/\*\*

\* 重写writeInternal ，处理如何输出数据到response。

\*/

@Override

protected void writeInternal(UserEntity userEntity,

HttpOutputMessage outputMessage)

throws IOException, HttpMessageNotWritableException {

String out = "hello: " + userEntity.getName() + "-" + userEntity.getAddress();

outputMessage.getBody().write(out.getBytes());

}

}

[复制代码](javascript:void(0);)

　　将自定义的消息转换器加入到springmvc容器中，以便被使用。

[复制代码](javascript:void(0);)

package com.example.demo;

import java.util.List;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.converter.HttpMessageConverter;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import com.alibaba.fastjson.serializer.SerializerFeature;

import com.alibaba.fastjson.support.config.FastJsonConfig;

import com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter;

@Configuration

public class MyWebmvcConfiguration implements WebMvcConfigurer{

@Override

public void extendMessageConverters(List<HttpMessageConverter<?>> converters) {

FastJsonHttpMessageConverter fjc = new FastJsonHttpMessageConverter();

FastJsonConfig fj = new FastJsonConfig();

fj.setSerializerFeatures(SerializerFeature.DisableCircularReferenceDetect);

fjc.setFastJsonConfig(fj);

converters.add(fjc);

converters.add(converter());

}

@Bean

public MyMessageConverter converter() {

return new MyMessageConverter();

}

}

[复制代码](javascript:void(0);)

　　UserController中加入测试的代码

[复制代码](javascript:void(0);)

package com.example.demo;

import java.util.ArrayList;

import java.util.List;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class UserController {

@RequestMapping(value="/get",method=RequestMethod.GET)

public Object getList(){

List<UserEntity> list= new ArrayList<UserEntity>();

UserEntity u1 = new UserEntity(null, "shanghai");

list.add(u1);

return list;

}

@RequestMapping(method = RequestMethod.POST, value = "/convert")

public @ResponseBody UserEntity converter(@RequestBody UserEntity user) {

return user;

}

}

[复制代码](javascript:void(0);)

　　启动项目，使用postman来测试，从响应来看我们的消息转换器已经起作用了，如下：

