原 spring mvc整合json

赞 0

发表于3年前(2013-01-13 23:59) 阅读（19626） | 评论（11） 48 人收藏此文章, 我要收藏

4月23日，武汉源创会火热报名中，期待您的参与>>>>> HOT

spring mvc

最近team要开发一个App，负责server端。原本准备是用SSH2框架的，但是感觉struts2还是比较适合用来与jsp结合使用，想了又想决定用spring mvc整合json来做。网上查了很多资料，调试了一整天，终于把调通。在这里分享一下我的经验：

1. 第一步当然创建一个新项目，加入spring啦^-^ 貌似现在只有3.0版本以上才支持整合json哦。
2. 在项目中加入两个json的jar包：jackson-core-asl-x.x.x.jar和jackson-mapper-asl.x.x.x.jar。其中xxx是jar包的版本，网上很多说用1.4.2，但是我加入1.4.2版本后，部署会出错。在请求中会出现ObjectMapper:NoClassDefFoundException的错，仔细查看原来tomcat启动的时候就错了，报的是isFullyTyped:NoSuchMethodError的错误。这个错真心纠结了很久，最后在官网上看到原来是core包里里面一个类没有这个方法。查看jar包，1.4.2真的没有这个方法。仔细查看官网API，尼玛json竟然不是向下兼容的，伤不起....最后看到了1.1版本还是支持的，于是就下载了1.1.2的版本。于是两个jar包就是：jackson-core-asl-1.1.2.jar和jackson-mapper-asl.1.1.2.jar。
3. 在web.xml添加spring mvc

```
<context-param>
    <param-name>contextConfigLocation</param-name>
        <span></span>         <span></span> <param-value>
        /WEB-INF/classes/applicationContext.xml,
        /WEB-INF/classes/spring-servlet.xml
    </param-value>
  </context-param>
  <listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
  </listener>
  <!-- spring mvc的servlet，处理所有.json请求 -->
  <servlet>
    <servlet-name>spring</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/classes/spring-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>spring</servlet-name>
    <url-pattern>*.json</url-pattern>
  </servlet-mapping>
```

4. 编写spring-servlet.xml，配置spring mvc

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.spring
                        http://www.springframework.org/schema/context http://www.spr

    <context:annotation-config />
    <!-- Controller 类注解标识 -->
    <context:component-scan base-package="com.gswtek.controller" />

    <bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver
        <property name="mediaTypes">
            <map>
                <!-- 告诉视图解析器，返回的类型为json格式 -->
                <entry key="json" value="application/json" />
            </map>
        </property>
        <property name="defaultViews">
            <list>
                <!-- ModelAndView里的数据变成JSON -->
                <bean class="org.springframework.web.servlet.view.json.MappingJackso
            </list>
        </property>
    </bean>
</beans>
```

5. 编写Controller类，返回的ModelAndView会自动将数据转换成json数据

```
package com.gswtek.controller;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.apache.log4j.Logger;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.portlet.ModelAndView;

import com.gswtek.service.UserService;
import com.gswtek.service.UserServiceImpl;

@Controller        /注解这个是个Controller
public class UserController {

    public static Logger log = Logger.getLogger(UserController.class);
    @Autowired
    @Qualifier("UserServiceImpl")
    private UserServiceImpl userService;

    public void setUserService(UserServiceImpl userService) {
        this.userService = userService;
    }

        /*
        * @RequestMapping配置请求地址
        * @RequestParam将请求中的参数注入
        */
        <span></span> @RequestMapping(value="/register")    <span></span> public Mod
                                  ,@RequestParam("password") String password) {
        log.debug("register a new user");
        ModelAndView modelView = new ModelAndView();
        Map<String, Object> modelMap = new HashMap<String, Object>();
        boolean status = userService.addUser(username, password);
        /*if(status) {
            modelMap.put("status", true);
        } else {
            modelMap.put("status", false);
        }*/
                modelMap.put("status",status);//感谢二楼的指正，已修改
                modelMap.put("date", new Date());
```

```
            modelView.addAllObjects(modelMap);
            return modelView;
        }

    }
```

6. service层的代码就不贴了，大家都懂的^-^
7. 部署测试，测试的地址是http://localhost:8080/spring_mvc/register.json?username=kevin&password=hahahah浏览器中返回数据是

```
{"modelAndView":{"empty":false,"reference":false,"viewName":null,"view":null,"modelMap":{
```

ok，成功！确实是json数据。

8. 观察一下，数据冗余有木有！没关系，定义我们自己的数据转换类：

```
package com.gswtek.util;

import java.util.Map;

import org.springframework.web.portlet.ModelAndView;
import org.springframework.web.servlet.view.json.MappingJacksonJsonView;

public class MappingJacksonJsonViewExd extends MappingJacksonJsonView {

    @Override
    protected Object filterModel(Map<String, Object> model) {
        Map<?, ?> result = (Map<?, ?>) super.filterModel(model);
        ModelAndView mav =  (ModelAndView) (result.size() != 1 ? result : result.val
        return mav.getModelMap();
    }

}
```

更改spring-servlet.xml：

```
                <property name="defaultViews">
            <list>
                <!-- ModelAndView里的数据变成JSON -->
                <bean class="com.gswtek.util.MappingJacksonJsonViewExd" />
            </list>
        </property>
```

9. 再次布局，测试：

```
{"status":true,"date":1358092422728}
```

ok,这次数据就比较简洁了。