**1、本章面试题**

      什么是restful风格，springmvc如何支持

springmvc 如何使用 ajax （jquery的ajax）

**2、知识点**

**2.1、课程回顾**

     文件上传（单文件和多文件上传），

注意点：

页面 : method="post" enctype="multipart/form-data"

controller: 输入框的name值必须和上传方法形参完全一致 @RequestParam

配置: id="multipartResolver" CommsMultipartResolver

统一异常处理 : HandlerExceptionResolver

**2.2、本章重点**

         postman 工具的使用

springmvc 的restful风格的使用

ajax 调用restful接口

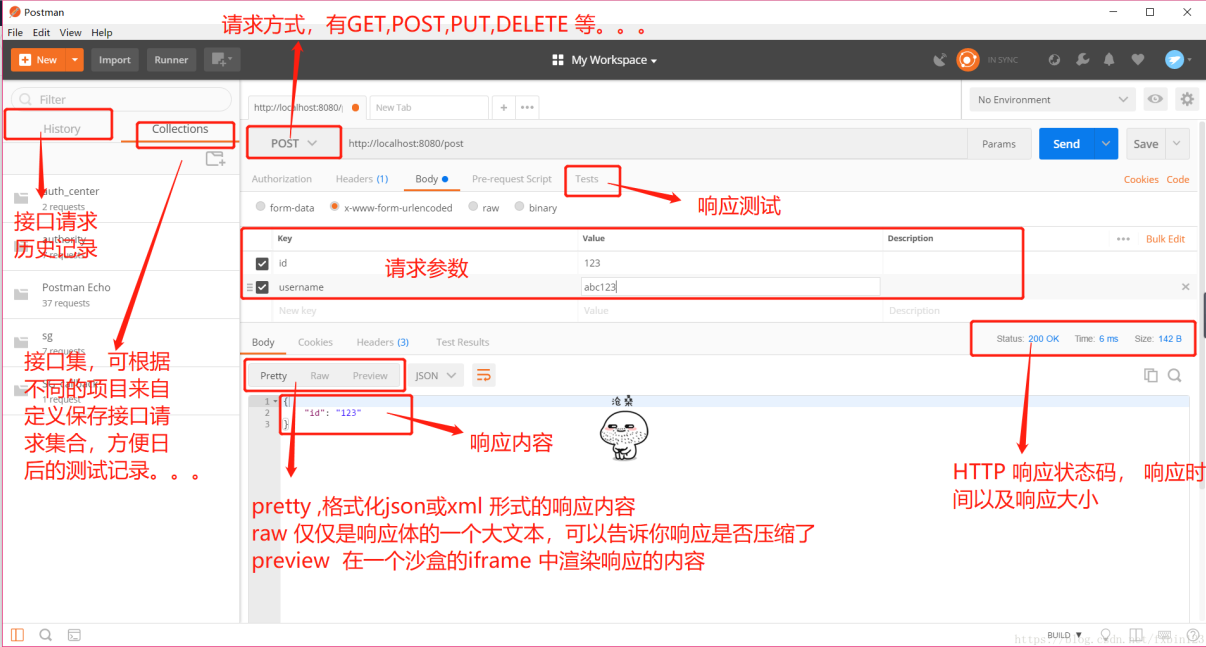
**3、具体内容E**

**3.1 postman简介和下载**

Postman一款非常流行的API调试工具。其实，开发人员用的更多。因为测试人员做接口测试会有更多选择 。不过，对于开发过程中去调试接口，Postman确实足够的简单方便，而且功能强大。

             官方网站：<https://www.getpostman.com/>

使用：



**3.2 Restful**

**3.2.1 概念**

RESTFUL是一种网络应用程序的设计风格和开发方式，基于HTTP，可以使用XML格式定义或JSON格式定义。

REST全称是Representational State Transfer，中文意思是表述性 状态 转移。 它首次出现在2000年Roy Fielding的博士论文中，Roy Fielding是HTTP规范的主要编写者之一。 他在论文中提到："我这篇文章的写作目的，就是想在符合架构原理的前提下，理解和评估以网络为基础的应用软件的架构设计，得到一个功能强、性能好、适宜通信的架构。REST指的是一组架构约束条件和原则。" 如果一个架构符合REST的约束条件和原则，我们就称它为RESTful架构。

**3.2.2 特点**

1、每一个URI代表1种资源； uri=/ssm/test/list.do 和 url =http://localhost:8080/ssm/test/list.do

2、客户端使用GET、POST、PUT、DELETE4个表示操作方式的动词对服务端资源进行操作： GET用来获取资源，POST用来新建资源 ，PUT用来更新资源，DELETE用来删除资源；

3、通过操作资源的表现形式来操作资源；

4、资源的表现形式是XML或者HTML或者JSON....；

5、 客户端与服务端之间的交互在请求之间是无状态的，从客户端到服务端的每个请求都必须包含理解请求所必需的信息。

**3.2.3 使用**

**1，在springmvc配置文件加入如下配置：**

<!--mvc:annotation-driven 作用：

Spring会默认帮我们注册处理请求，参数和返回值的类

会自动注册RequestMappingHandlerMapping、RequestMappingHandlerAdapter 与ExceptionHandlerExceptionResolver 三个bean。

支持使用 ConversionService 实例对表单参数进行类型转换；

支持使用 @NumberFormat annotation、@DateTimeFormat；

支持使用 @RequestBody 和 @ResponseBody 注解

-->

<mvc:annotation-driven>

<mvc:message-converters>

<!--完成请求报文到字符串和字符串到响应报文的转换-->

<bean class="org.springframework.http.converter.StringHttpMessageConverter"/>

<bean class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter">

<property name="supportedMediaTypes">

<list>

<value>text/html;charset=UTF-8</value>

<value>application/json;charset=UTF-8</value>

</list>

</property>

</bean>

</mvc:message-converters>

</mvc:annotation-driven>

**2，编写添加方法**

package com.aaa.ssm.controller;

import com.aaa.ssm.entity.Dept;

import com.aaa.ssm.service.DeptService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.util.Map;

/\*\*

\* fileName:DeptController

\* description:

\* author:zz

\* createTime:2019/11/5 9:28

\* version:1.0.0

\*/

@Controller //把该类交给spring管理 自定义控制器

@RequestMapping("/dept") //所有请求映射的父路径

public class DeptController {

//依赖注入

@Autowired

private DeptService deptService;

/\*\*

\* 查询

\* @param map

\* @return

\*/

@ResponseBody //默认让方法的返回json格式

@RequestMapping(method = RequestMethod.GET)

public Object list(@RequestBody Map map){

return deptService.getList(map);

}

/\*\*

\* 部门添加

\* @param dept

\* @return

\*/

@ResponseBody

@RequestMapping(method = RequestMethod.PUT)

// RequestBody 默认让方法的接受参数为json格式

public Object add(@RequestBody Dept dept){

return deptService.add(dept);

}

/\*\*

\* 部门更新

\* @param dept

\* @return

\*/

@ResponseBody

@RequestMapping(method = RequestMethod.POST)

// RequestBody 默认让方法的接受参数为json格式

public Object update(@RequestBody Dept dept){

return deptService.update(dept);

}

/\*\*

\* 部门删除

\* @param deptNo

\* @return

\*/

@ResponseBody

@RequestMapping(value = "/{deptId}.do",method = RequestMethod.DELETE)

// RequestBody 默认让方法的接受参数为json格式

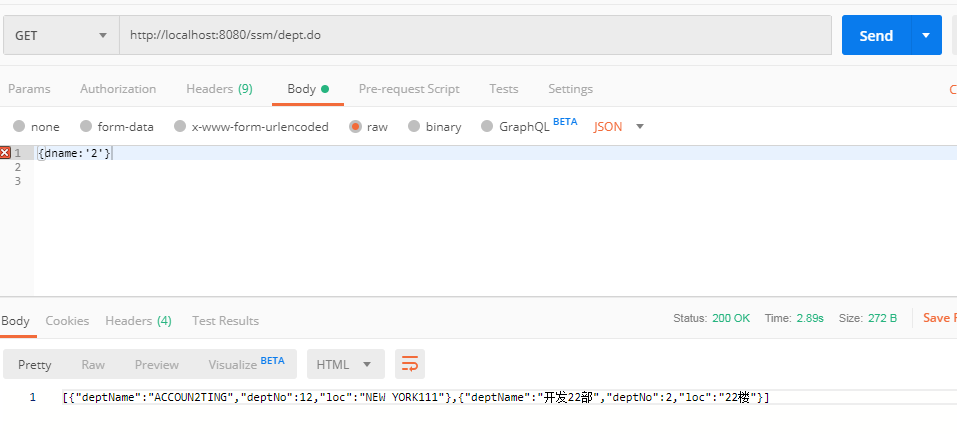
public Object delete(@PathVariable("deptId") Integer deptNo){

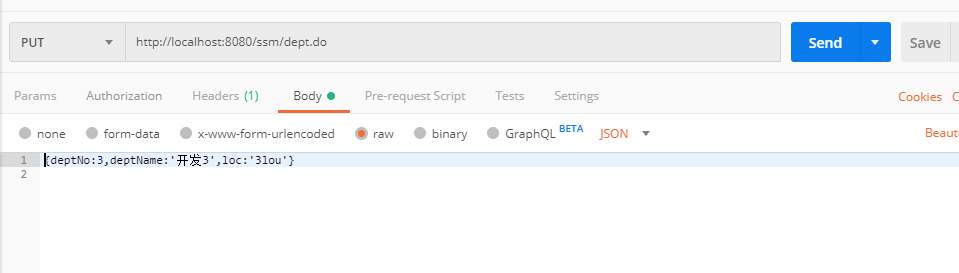
return deptService.delete(deptNo);

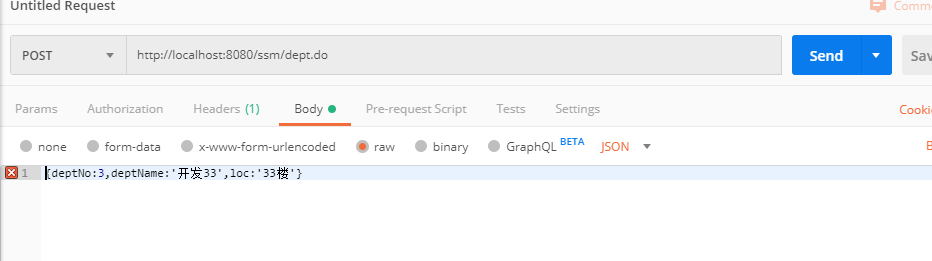
}

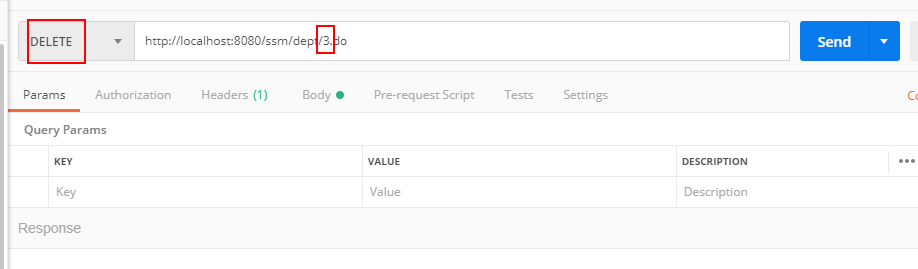
}

**3，postman**









**3.3 jquery ajax**

**get:**

$.ajax({

url:"/deptRestful/list.do",//请求地址

type:"GET",//请求方式

dataType:"json",//返回值类型

success:function (data) {

if(data!=null&&data!=''&&data!='null'){

//清空原有数据

$("#tbd").empty();

//循环追加每一行数据

for (var i = 0; i < data.length; i++) {

var dept = data[i];

$("#tbd").append("<tr><td>"+dept.deptName+"</td><td>"+dept.loc+"</td>" +

"<td><a href=\"/html/dept/update.html?deptNo="+dept.deptNo+"\">更新</a>&nbsp;&nbsp;" +

"<a href='javascript:del("+dept.deptNo+")'>删除</a></td></tr>");

}

}

},

error:function () {

alert("有错误！！！")

}

});

**put:**

$.ajax({

url:"/deptRestful.do",

type:"POST",

contentType:"application/json;charset=utf-8",//设置请求参数的格式

data:JSON.stringify(jsonData),

dataType:"json",//设置返回值格式

success:function (data) { //回调函数 成功

// == 1=='1' true 值相等为true === 1==='1' 类型和值都要相等

if(data==1){

location.href='/html/dept/list.html';

}

}

});

**post:**

$("#btn").click(function () {

var jsonData ={deptName:$("#deptName").val(),loc:$("#loc").val()};

//JSON.stringify(json) 把json对象转换为json字符串

//JSON.parse(jsonStr) 把json格式的字符串转换为json对象

alert(jsonData.deptName);

$.ajax({

url:"/deptRestful.do",

type:"POST",

contentType:"application/json;charset=utf-8",//设置请求参数的格式

data:JSON.stringify(jsonData),

dataType:"json",//设置返回值格式

success:function (data) { //回调函数 成功

// == 1=='1' true 值相等为true === 1==='1' 类型和值都要相等

if(data==1){

location.href='/html/dept/list.html';

}

}

});

});

**delete(不使用**PathVariable传递不了参数**):**

$.ajax({

url:"/deptRestful/"+deptNo+".do",//请求地址

type:"DELETE",//请求方式

dataType:"json",//返回值类型

success:function (data) {

if(data==1){

location.href="/html/dept/list.html";

}

},

error:function () {

alert("有错误！！！")

}

});

@ResponseBody

@RequestMapping(value="/{deptId}.do",method=RequestMethod.DELETE)

public Object delete(@PathVariable("deptId") Integer deptNo){

return deptService.delete(deptNo);

}

**4、本章总结**

**4.1 总结本章知识点**

**4.2 面试题答案**

**4.3 预习下一章重点**