# 0 复习Vue

1 MVVM：前端的设计模式，他实现了双向数据绑定，他与MVC有什么框架？MVC是没有实现双向数据绑定

2 双向数据绑定：当model数据发生改变之后，页面view层自动改动；当页面数据发生改变的时候，model层数据也会接收到变化

3 生命周期钩子函数：created：new vue（）执行完成，在双向数据绑定之前，通常用来获取数据，紧跟着就可以进行双向绑定

注：回顾下vue的8个钩子函数

4 指令：

* 插值表达式{{}}
* v-text，v-html
* v-model：绑定表单元素的input、radio，CheckBox、textarea、select
* v-on：事件修饰符 简写 @ ，有@click @mouseover @mouseout @keydown @keyup...
* js的冒泡事件：阻止冒泡的发生 .stop
* 阻止js的默认事件的发生 .prevent
* v-if
* v-for
* v-bind：绑定的是属性，简写 ：
* 计算属性：本质是方法，但是我们可以以属性的方式调用
* axios：发送ajax
* 组件：父子组件传值

# 1 Vue复习-商品管理系统

目的：1 回顾Vue的基本用法

2 以这个练习为切入点，讲解前后端分离的开发方式

## 1.1 案例需求

完成商品的查询、修改、删除等操作

前端（浏览器端）：vue

后端（服务器端）：spring boot+spring mvc+mybatis

## 1.2 数据库设计与表结构

CREATE TABLE `t\_goods` (   
 `id` int(11) NOT NULL AUTO\_INCREMENT,   
 `name` varchar(255) DEFAULT NULL,   
 `image` varchar(255) DEFAULT NULL,   
 `price` double(20,0) DEFAULT NULL,   
 PRIMARY KEY (`id`)   
)

导入数据：

INSERT INTO `t\_goods` VALUES ('1', '华为 G9 青春版 白色 移动联通电信4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '84900');  
INSERT INTO `t\_goods` VALUES ('2', '华为 G9 青春版 金色 移动联通电信4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '84900');  
INSERT INTO `t\_goods` VALUES ('3', '三星 Galaxy C5（SM-C5000）4GB+32GB 枫叶金 移动联通电信4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '119900');  
INSERT INTO `t\_goods` VALUES ('4', '三星 Galaxy C5（SM-C5000）4GB+32GB 蔷薇粉 移动联通电信4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '119900');  
INSERT INTO `t\_goods` VALUES ('5', '三星 Galaxy C5（SM-C5000）4GB+32GB 烟雨灰 移动联通电信4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '119900');  
INSERT INTO `t\_goods` VALUES ('6', '三星 Galaxy C5（SM-C5000）4GB+32GB 皎洁银 移动联通电信4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '119900');  
INSERT INTO `t\_goods` VALUES ('7', '华为 G9 Plus 32GB 月光银 移动联通4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '119900');  
INSERT INTO `t\_goods` VALUES ('8', '华为 麦芒5 全网通 4GB+64GB版 香槟金 移动联通电信4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '139900');  
INSERT INTO `t\_goods` VALUES ('9', '努比亚（nubia）Z11 百合金 4GB+64GB 全网通 移动联通电信4G手机双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '159900');  
INSERT INTO `t\_goods` VALUES ('10', '华为 HUAWEI nova 4GB+64GB版 香槟金（白）移动联通电信4G手机 双卡双待', 'https://img12.360buyimg.com/n1/jfs/t17050/106/1124838205/250590/7e63050a/5abb6be2N853106d9.jpg', '139900');

## 1.3 服务器端

### 1.3.1 创建SprignBoot项目

过程略。

### 1.3.2 导入jar包

pom.xml

<!-- 统一版本维护 -->  
<properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
 <mybatis.starter.version>1.3.2</mybatis.starter.version>  
 <mapper.starter.version>2.0.2</mapper.starter.version>  
 <mysql.version>5.1.32</mysql.version>  
 <pageHelper.starter.version>1.2.5</pageHelper.starter.version>  
 <durid.starter.version>1.1.10</durid.starter.version>  
</properties>  
<dependencies>  
 <!-- SpringBoot整合SpringMVC的启动器 -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
   
 </dependency>  
 <!-- SpringBoot整合jdbc和事务的启动器 -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-jdbc</artifactId>  
 </dependency>  
 <!-- mybatis启动器 -->  
 <dependency>  
 <groupId>org.mybatis.spring.boot</groupId>  
 <artifactId>mybatis-spring-boot-starter</artifactId>  
 <version>${mybatis.starter.version}</version>  
 </dependency>  
 <!-- 通用Mapper启动器 -->  
 <dependency>  
 <groupId>tk.mybatis</groupId>  
 <artifactId>mapper-spring-boot-starter</artifactId>  
 <version>${mapper.starter.version}</version>  
 </dependency>  
 <!-- 分页助手启动器 -->  
 <dependency>  
 <groupId>com.github.pagehelper</groupId>  
 <artifactId>pagehelper-spring-boot-starter</artifactId>  
 <version>${pageHelper.starter.version}</version>  
 </dependency>  
 <!-- mysql驱动 -->  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 <version>${mysql.version}</version>  
 </dependency>  
 <!-- Druid连接池 -->  
 <dependency>  
 <groupId>com.alibaba</groupId>  
 <artifactId>druid-spring-boot-starter</artifactId>  
 <version>${durid.starter.version}</version>  
 </dependency>  
</dependencies>

### 1.3.3 配置文件（application.properties）

application.properties文件

#Tomcat   
server.port=8090   
#DB configuration   
spring.datasource.driverClassName=com.mysql.jdbc.Driver   
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/vue01?useUnicode=true&characterEncoding=utf8   
spring.datasource.username=root   
spring.datasource.password=123456   
#druid   
spring.datasource.driver-class-name=com.mysql.jdbc.Driver   
spring.datasource.druid.initial-size=1   
spring.datasource.druid.min-idle=1   
spring.datasource.druid.max-active=20   
spring.datasource.druid.test-on-borrow=true   
spring.datasource.druid.stat-view-servlet.allow=true

### 1.3.4 创建springboot引导类

@SpringBootApplication  
public class UserManagementApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.run(UserManagementApplication.class, args);  
 }  
}

### 1.3.5 创建实体

创建包com.czxy.pojo，创建类User.java

@Table(name="t\_goods")  
public class Goods {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.IDENTITY)  
 public Integer id;  
 private String name;  
 private String image;  
 private Double price;  
}

### 1.3.6 创建Dao

创建包com.czxy.dao，创建接口GoodsMapper.java

@org.apache.ibatis.annotations.Mapper  
public interface GoodsMapper extends Mapper<Goods> {  
  
}

### 1.3.7 创建Service

创建包com.czxy.service，创建接口GoodsService.java

@Service  
@Transactional  
public class GoodsService {  
  
 @Autowired  
 private GoodsMapper goodsMapper;  
  
}

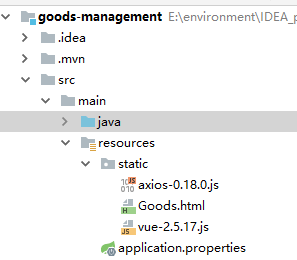
### 1.3.8 创建Controller

创建包com.czxy.controller，创建类GoodsController.java

@RestController  
@RequestMapping("/goods")  
public class GoodsController {  
 @Autowired  
 private GoodsService goodsService;  
}

## 1.4 前端

拷贝页面到resources下的static中



### 1.4.1 Goods.html

我们把所有的vue的内容放置到页面中

<script src="js/vue-2.6.10.js"></script>  
<script src="js/axios-0.18.0.js"></script>

### 1.4.2 查询所有

思路：后端查询所有的商品数据

前端显示所有

#### 1.4.2.1 GoodsService

public List<Goods> findAllGoods(){  
 return goodsMapper.selectAll();  
}

#### 1.4.2.2 GoodsController

@GetMapping  
public ResponseEntity<List<Goods>> findAll(){  
 List<Goods> list = goodsService.findAllGoods();  
 return new ResponseEntity<>(list,HttpStatus.OK);  
}

#### 1.4.2.3 Goods.html页面

第一步：引入vue库文件

<script src="js/vue-2.6.10.js"></script>  
<script src="js/axios-0.18.0.js"></script>

第二步：页面模板，将所有内容包含到一个父容器中

<div id="app">  
</div>

第三步：编写vue代码

var vm = new Vue({  
 el:"#app",  
 data:{  
 goods:[]  
 },  
 created(){  
 //发送ajax请求  
 axios.get("/goods").then(res => {  
 console.log(res)  
 this.goods = res.data  
 })  
 }  
})

第四步：编写Goods.html

<div id="app">  
 <table border="1" cellspacing="0" cellpadding="0" width="800" height="400">  
 <tr>  
 <th>编号</th>  
 <th>商品名字</th>  
 <th>商品图片</th>  
 <th>商品价格</th>  
 </tr>  
 <tr v-for="(goods,index) in goods">  
 <td>{{goods.id}}</td>  
 <td>{{goods.name}}</td>  
 <td><img src="{{goods.image}}"/></td>  
 <td>{{goods.price}}</td>  
 </tr>  
 </table>  
  
</div>

测试结果：



#### 1.4.2.3 优化一：图片不显示的问题



修改：

<td><img :src="goods.image"/></td>

效果如下：



#### 1.4.2.3 优化二：图片太大的问题

<td><img :src="goods.image" width="100" height="100"/></td>

效果：



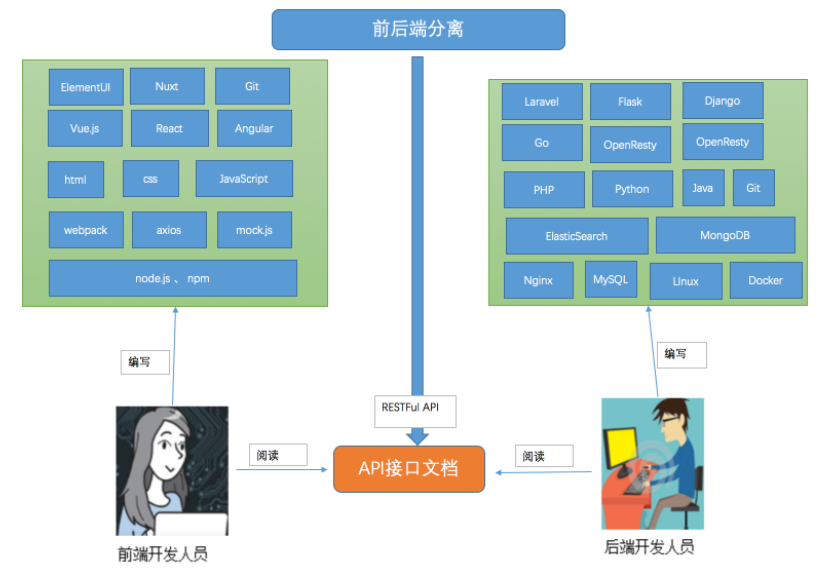
# 2 前后端分离介绍

## 2.1 前后端分离

前后端分离是目前一种非常流行的开发模式，它使项目的分工更加明确：

* 后端：负责处理、存储数据
* 前端：负责显示数据

前端和后端开发人员通过 **接口** 进行数据的交换。



好处：

1. 一套后端接口为所有前端提供数据（PC端、APP端、桌面端等）
2. 前、后端代码彼此独立互不影响

坏处：

1. 当接口改变的时候，非常麻烦--
2. 需要前后端人员联调--联调开发的时间（开发+测试+联调）占项目的 15%--60%

## 2.2 JavaScript框架

随着前端的发展，原来很多由服务器端完成的功能现在都转移到了前端来实现，这就导致了前端会编写出成千上万行的JavaScript代码，用它们来管理各种各样的HTML和CSS文件，但是缺乏正式的组织形式：



为了更好的组织前端的代码，开发者们越来越多的开始使用JavaScript框架，最流行的几个前端框架有：

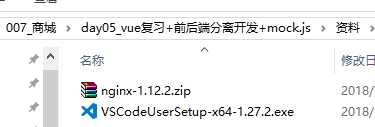
1. Google 的 Angularjs
2. Facebook的React
3. 尤雨溪的Vue

# 3 商品管理改造

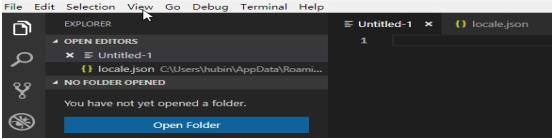
## 3.1 前后端分离开发具体实现

### 3.1.1 Visual Studio Code开发工具的安装

资料位置：



双击安装，即可成功

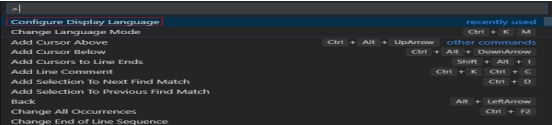


### 3.1.2 Visual Studio Code开发工具的配置

#### 3.1.2.1、设置中文环境

* 设置中文环境

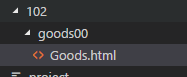
安装完毕后，vscode全部都是英文版本的，需要把vscode环境调整为中文版本。

* 调整步骤如下：
  + 第一步：快捷键ctrl+shift+p
  + 选择Configure Display Language
  + 
  + 第二步：选择安装"其他语言"
  + ![1571691696535](assets/1571691696535.png)
  + 第三步：设置完毕保存文件，重启vscode，就会显示中文。
  + ![1541954970803](assets/1541954970803.png)
  + 好了，到此vscode已经变成中文模式了，现在就可以进入到开发中了。

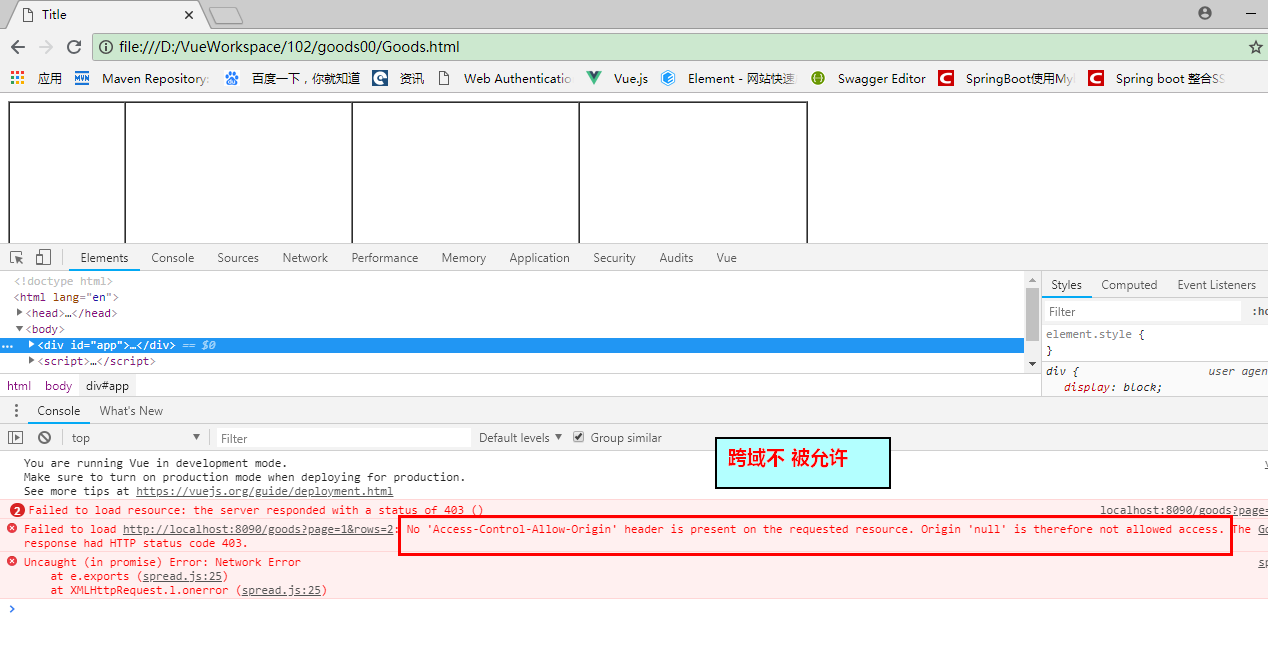
#### 3.1.2.2 安装Vetur插件

* 功能表述
* 总：综合比较应该是目前 VSCode 上面最好用的 Vue 插件
* 分：
  + 语法错误检查，包括 CSS/SCSS/LESS/Javascript/TypeScript
  + 语法高亮，包括 html/jade/pug css/sass/scss/less/stylus js/ts
  + emmet 支持
  + 代码自动补全（目前还是初级阶段），包括 HTML/CSS/SCSS/LESS/JavaScript/TypeScript
  + .....
* 安装步骤，如下图操作
* 

## 3.2 将商品开发的页面放到vs中



复制地址，直接在浏览器打开：



## 3.3.跨域问题

### 3.3.1.什么是跨域

跨域是指跨域名的访问，以下情况都属于跨域：

|  |  |
| --- | --- |
| 跨域原因说明 | 示例 |
| 域名不同 | www.jd.com 与 www.taobao.com |
| 域名相同，端口不同 | www.jd.com:8080 与 www.jd.com:8081 |
| 二级域名不同/子域名不同 | item.jd.com 与 miaosha.jd.com |
| 协议不同 | http://jd.com与https://jd.com |
| 域名和ip地址 | http://jd.com与192.168.23.12 |

如果**域名和端口都相同，但是请求路径不同**，不属于跨域，如：

www.jd.com/item

www.jd.com/goods

而我们刚才是从file:///D:/VueWorkspace/102/goods00/Goods.html去访问http://localhost:8090/，这属于域名不同，跨域了。

### 3.3.2.为什么有跨域问题？

1 浏览器的**同源策略**引起跨域问题

2 跨域不一定会有跨域问题。

答：a.跨域没有引起问题：

b.跨域引起了问题  
  
 因为跨域问题是浏览器对于ajax请求的一种安全限制：\*\*一个页面发起的ajax请求，只能是于当前页同域名的路径\*\*，这能有效的阻止跨站攻击。

因此：**跨域问题 是针对ajax的一种限制**。

但是这却给我们的开发带来了不便，而且在实际生成环境中，肯定会有很多台服务器之间交互，地址和端口都可能不同，怎么办？

### 3.3.3.解决跨域问题的方案

目前比较常用的跨域解决方案有3种：

* Jsonp
* 最早的解决方案，利用script标签（不受限）可以跨域的原理实现。
* 限制：
  + 需要服务的支持
  + 只能发起GET请求
* nginx反向代理
* 思路是：利用nginx反向代理把跨域为不跨域，支持各种请求方式
* CORS
* 规范化的跨域请求解决方案，安全可靠。是w3c组织的标准
* 优势：
  + 在服务端进行控制是否允许跨域，可自定义规则
  + 支持各种请求方式：get、post、put、delete
  + 限制访问的电脑的IP地址
* 缺点：
  + 会产生额外的请求（可能发一次/二次请求）

我们这里会采用nginx搭建静态页面服务器+cors的跨域方案。

## 3.4.cors解决跨域

### 3.4.1.什么是cors

CORS是一个W3C标准，全称是"跨域资源共享"（Cross-origin resource sharing）。

它允许浏览器向跨源服务器，发出[XMLHttpRequest](http://www.ruanyifeng.com/blog/2012/09/xmlhttprequest_level_2.html)请求，从而克服了AJAX只能[同源](http://www.ruanyifeng.com/blog/2016/04/same-origin-policy.html)使用的限制。

CORS需要浏览器和服务器同时支持。目前，所有浏览器都支持该功能，IE浏览器不能低于IE10。

* 浏览器端：
* 目前，所有浏览器都支持该功能（IE10以下不行）。整个CORS通信过程，都是浏览器自动完成，不需要用户参与。
* 服务端：
* 一般**通过过滤器**完成功能。

### 3.4.2.CORS原理有点复杂(了解)

浏览器会将ajax请求分为两类，其处理方案略有差异：简单请求、特殊请求。

#### 简单请求

只要同时满足以下两大条件，就属于简单请求。：

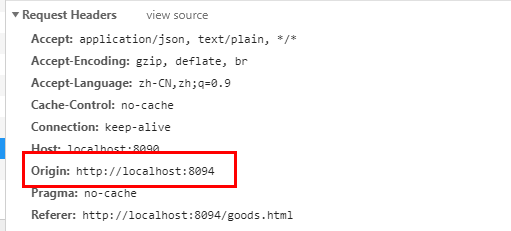
（1) 请求方法是以下三种方法之一：

* HEAD
* GET
* POST

（2）HTTP的头信息不超出以下5种字段：

* Accept
* Accept-Language
* Content-Language
* Last-Event-ID
* Content-Type：只限于三个值application/x-www-form-urlencoded、multipart/form-data、text/plain

当浏览器发现发现的ajax请求是简单请求时，会在请求头中携带一个字段：Origin.



Origin中会指出当前请求属于哪个域（协议+域名+端口）。服务会根据这个值决定是否允许其跨域。

如果服务器允许跨域，需要在返回的响应头中携带下面信息：

Access-Control-Allow-Origin: http://localhost:8094  
Access-Control-Allow-Credentials: true  
Content-Type: text/html; charset=utf-8

* Access-Control-Allow-Origin：可接受的域，是一个具体域名或者\*，代表任意
* Access-Control-Allow-Credentials：是否允许携带cookie，默认情况下，cors不会携带cookie，除非这个值是true

注意：

如果跨域请求要想操作cookie，需要满足3个条件：

* 服务的响应头中需要携带Access-Control-Allow-Credentials并且为true。
* 浏览器发起ajax需要指定withCredentials 为true
* 响应头中的Access-Control-Allow-Origin一定不能为\*，必须是指定的域名

#### 特殊请求

不符合简单请求的条件，会被浏览器判定为特殊请求,，例如请求方式为PUT。

预检请求

特殊请求会在正式通信之前，增加一次HTTP查询请求，称为"预检"请求（preflight）。

浏览器先询问服务器，当前网页所在的域名是否在服务器的许可名单之中，以及可以使用哪些HTTP动词和头信息字段。只有得到肯定答复，浏览器才会发出正式的XMLHttpRequest请求，否则就报错。

一个“预检”请求的样板：

OPTIONS /cors HTTP/1.1  
Origin: http://localhost:8094  
Access-Control-Request-Method: PUT  
Access-Control-Request-Headers: X-Custom-Header  
Host: http://localhost:8094  
Accept-Language: en-US  
Connection: keep-alive  
User-Agent: Mozilla/5.0...

与简单请求相比，除了Origin以外，多了两个头：

* Access-Control-Request-Method：接下来会用到的请求方式，比如PUT
* Access-Control-Request-Headers：会额外用到的头信息

预检请求的响应

服务的收到预检请求，如果许可跨域，会发出响应：

HTTP/1.1 200 OK  
Date: Mon, 01 Dec 2008 01:15:39 GMT  
Server: Apache/2.0.61 (Unix)  
Access-Control-Allow-Origin: http://localhost:8094  
Access-Control-Allow-Credentials: true  
Access-Control-Allow-Methods: GET, POST, PUT  
Access-Control-Allow-Headers: X-Custom-Header  
Access-Control-Max-Age: 1728000  
Content-Type: text/html; charset=utf-8  
Content-Encoding: gzip  
Content-Length: 0  
Keep-Alive: timeout=2, max=100  
Connection: Keep-Alive  
Content-Type: text/plain

除了Access-Control-Allow-Origin和Access-Control-Allow-Credentials以外，这里又额外多出3个头：

* Access-Control-Allow-Methods：允许访问的方式
* Access-Control-Allow-Headers：允许携带的头
* Access-Control-Max-Age：本次许可的有效时长，单位是秒，**过期之前的ajax请求就无需再次进行预检了**

如果浏览器得到上述响应，则认定为可以跨域，后续就跟简单请求的处理是一样的了。

### 3.4.3.实现非常简单

虽然原理比较复杂，但是前面说过：

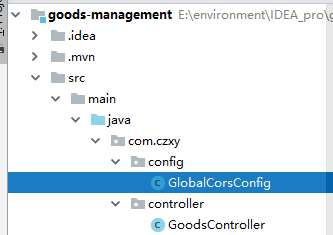
* 浏览器端都有浏览器自动完成，我们无需操心
* 服务端可以通过拦截器统一实现，不必每次都去进行跨域判定的编写。

事实上，SpringMVC已经帮我们写好了CORS的跨域过滤器：CorsFilter ,内部已经实现了刚才所讲的判定逻辑，我们直接用就好了。

在config目录中编写一个配置类，并且注册CorsFilter：

import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.web.cors.CorsConfiguration;  
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;  
import org.springframework.web.filter.CorsFilter;  
  
@Configuration  
public class GlobalCorsConfig {  
 @Bean  
 public CorsFilter corsFilter() {  
 //1.添加CORS配置信息  
 CorsConfiguration config = new CorsConfiguration();  
 //1) 允许的域,不要写\*，否则cookie就无法使用了  
 config.addAllowedOrigin("http://localhost:8094");  
 //2) 是否发送Cookie信息  
 config.setAllowCredentials(true);  
 //3) 允许的请求方式  
 config.addAllowedMethod("OPTIONS");  
 config.addAllowedMethod("HEAD");  
 config.addAllowedMethod("GET");  
 config.addAllowedMethod("PUT");  
 config.addAllowedMethod("POST");  
 config.addAllowedMethod("DELETE");  
 config.addAllowedMethod("PATCH");  
 // 4）允许的头信息  
 config.addAllowedHeader("\*");  
  
 //2.添加映射路径，我们拦截一切请求  
 UrlBasedCorsConfigurationSource configSource = new UrlBasedCorsConfigurationSource();  
 configSource.registerCorsConfiguration("/\*\*", config);  
  
 //3.返回新的CorsFilter.  
 return new CorsFilter(configSource);  
 }  
}

结构：



4.5.4.重启测试：还是无法访问

## 总结：

0 前后端分离开发的好处

* **后端追求的是：三高（高并发，高可用，高性能），安全，存储，业务等等。**
* **前端追求的是：页面表现，速度流畅，兼容性，用户体验等等。**

让前后端更加专注自己的业务领域

1 什么是跨域

2 为什么会有跨越问题

3 跨域解决方案：解决方案的优缺点（JSONP 和cors的优缺点）

4 nginx：反向代理服务器，通过域名访问项目；搭建静态资源服务器

4 cors是什么

5 cors的解决方案：如何实现？

6 前后端分离之后，html页面放到了vs下，后端java只提供接口，那么vs下的html页面如何访问后端接口？

答：要将html静态资源放到服务器下，而nginx就是最优秀的静态资源服务器

7 nginx搭建HTML静态资源服务器

8 CORS原理总结（了解）：

* 简单请求
  + 只能是head、get、post请求
  + 头信息不超过5种数据
  + 只需要发送一次请求
* 复杂请求
  + 发送一次或者两次请求
  + put、delete等请求
  + 第一次请求是预检请求，判断服务器是否会处理该请求，第二次请求是预检通过之后，向服务器发送真正的请求