**品优购电商系统开发**

**第2章**

**AngularJS&Nginx&品牌管理**

传智播客.黑马程序员

# 课程目标

目标1：掌握代码生成器的使用

目标2：运用AngularJS前端框架常用指令

目标3：完成安装Nginx和配置nginx.conf

目标4：实现品牌列表的前端代码

目标5：实现增加品牌的功能

目标6：实现修改品牌的功能

# 分页PageHelper与通用Mapper

## 1.1集成分页助手PageHelper

使用分页助手PageHelper的好处？

1. 简化分页查询。select \* from tb\_user limit ?,?
2. 简化统计查询。Select count(1) from tb\_user
3. 不用关心底层数据库(mysql、oracle)，因为分页助手会根据方言自动生成数据库相关分页、统计SQL语句。

### 1.1.1配置部分

修改pinyougou-mapper的pom.xml文件，加入依赖

|  |
| --- |
| *<!-- 配置分页插件 -->* <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**> </**dependency**> |

修改pinyougou-mapper/src/main/resources/mybatis-config.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>*** <**configuration**>  *<!-- 全局的设置信息-->* <**settings**>  *<!-- 开启驼峰命名(方便自动映射) dept\_id deptId -->* <**setting name="mapUnderscoreToCamelCase" value="true"**/>  </**settings**>  *<!-- 配置插件 -->* <**plugins**>  *<!-- 配置分页拦截器(拦截SqL语句，在你的sql语句后面加上limit) -->* <**plugin interceptor="com.github.pagehelper.PageInterceptor"**>  *<!-- 配置数据库方言(mysql、oracle、sqlserver等) -->* <**property name="helperDialect" value="mysql"**/>  *<!-- 配置开启统计查询 count -->* <**property name="rowBoundsWithCount" value="true"**/>  </**plugin**>  </**plugins**> </**configuration**> |

### 1.1.2代码部分

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl/BrandServiceImpl.java

|  |
| --- |
| @Override **public** List<Brand> findAll() {  *// 开始分页* PageInfo<Brand> pageInfo = PageHelper.*startPage*(1, 10)  .doSelectPageInfo(**new** ISelect() {  @Override  **public void** doSelect() {  **brandMapper**.findAll();  }  });  System.***out***.println(**"总记录数："** + pageInfo.getTotal());  System.***out***.println(**"总页数："** + pageInfo.getPages());  **return** pageInfo.getList(); } |

**测试地址为：[http://localhost:9101/brand/f](http://localhost:9091/rest/item/cat/query/1?rows=5)indAll**

PageInfo对象常用方法：

* pageInfo.getTotal(): 获取总记录数
* pageInfo.getPages(): 获取总页数
* pageInfo.getList(): 获取分页数据集合

## 1.2集成通用Mapper

使用通用Mapper的好处？

1. 简化单表CRUD操作(不需要写SQL语句)，添加、删除、修改、查询是经常需要用到的功能，加快项目开发进度。

### 1.2.1配置部分

修改pinyougou-pojo项目的pom.xml文件，加入依赖

|  |
| --- |
| <**dependencies**>  *<!-- 配置通用mapper -->* <**dependency**>  <**groupId**>tk.mybatis</**groupId**>  <**artifactId**>mapper</**artifactId**>  </**dependency**> </**dependencies**> |

pinyougou-mapper/src/main/resources/applicationContext-mapper.xml添加如下配置信息：

|  |
| --- |
| *<!-- 配置通用Mapper集成Spring4 -->* <**bean class="tk.mybatis.spring.mapper.MapperScannerConfigurer"**>  *<!-- 设置数据访问接口类 -->* <**property name="basePackage" value="com.pinyougou.mapper"**/> </**bean**> |

### 1.2.2代码部分

修改pinyougou-pojo/src/main/java/com.pinyougou.pojo/Brand.java，加入JPA注解，通用Mapper需要该类与数据库表之间的映射关系

|  |
| --- |
| @Table(name=**"tb\_brand"**) **public class** Brand **implements** Serializable{  */\*\* 主键ID \*/* @Id @GeneratedValue(strategy=GenerationType.***IDENTITY***)  @Column(name=**"id"**)  **private** Long **id**;  */\*\* 品牌名称 \*/* @Column(name=**"name"**)  **private** String **name**;  */\*\* 品牌首字母 \*/* @Column(name=**"first\_char"**)  **private** String **firstChar**;  ......  } |

修改pinyougou-mapper/src/main/java/com.pinyougou.mapper包下的

BrandMapper.java，继承Mapper接口(Mapper接口有CRUD方法)

|  |
| --- |
| */\*\* 品牌数据访问接口 \*/* **public interface** BrandMapper **extends** Mapper<Brand> { } |

修改pinyougou-sellergoods-service/src/main/java/

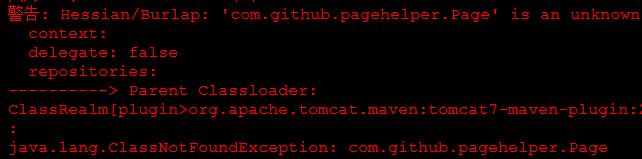
com.pinyougou.sellergoods.service.impl包下的BrandServiceImpl.java：

|  |
| --- |
| @Override **public** List<Brand> findAll() {  *// 开始分页* PageInfo<Brand> pageInfo = PageHelper.*startPage*(1, 10)  .doSelectPageInfo(**new** ISelect() {  @Override  **public void** doSelect() {  **brandMapper.selectAll();**  }  });  System.***out***.println(**"总记录数："** + pageInfo.getTotal());  System.***out***.println(**"总页数："** + pageInfo.getPages());  **return** pageInfo.getList(); } |

**测试地址为：[http://localhost:9101/brand/f](http://localhost:9091/rest/item/cat/query/1?rows=5)indAll**

Mapper接口常用到得四批方法：

1. mapper.selectXxx(): 查询方法
2. mapper.insertXxx(): 添加方法
3. mapper.updateXxx(): 修改方法
4. mapper.deleteXxx(): 删除方法



注意：因为pinyougou-manager-web工程也需要用到Page类，所以需要在pinyougou-interface工程的pom.xml文件加入以下依赖：

|  |
| --- |
| *<!-- 配置分页插件 -->* <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**> </**dependency**> |

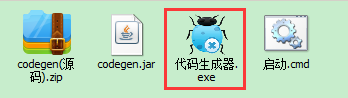
# 代码生成器【快速开发】

## 2.1拷贝实体类

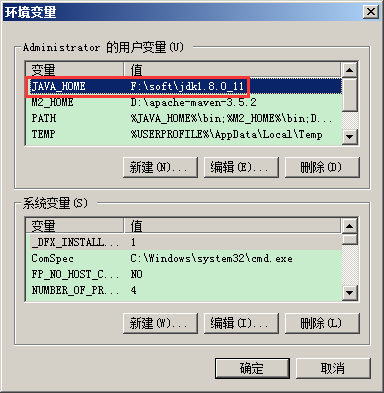
将【资料\实体类】下面所有的类 拷贝到pinyougou-pojo/src/main/java的com.pinyougou.pojo包下。

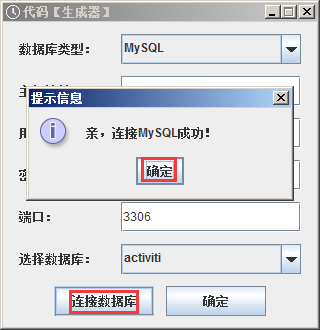
## 2.2代码生成器

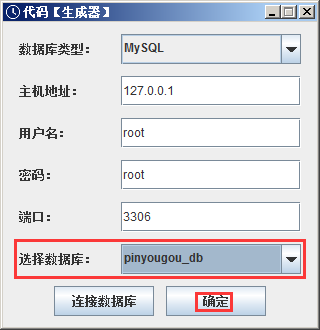
在【资料\生成器】文件夹：



在使用“代码生成器.exe”之前，需配置环境变量：**JAVA\_HOME**







## 2.3生成数据访问层



**注意：**

* **拷贝pinyougou\pinyougou-mapper\src\main\java\com\pinyougou\**

**Mapper目录下全部XxxMapper.java数据访问接口类到pinyougou-mapper/src/main/java/com.pinyougou.mapper/包下。**

* **拷贝pinyougou\pinyougou-mapper\src\main\resources\mappers目录下全部XxxMapper.xml映射文件到pinyougou-mapper/src/main/**

**resources/mappers/包下。**

## 2.4生成服务接口类



**注意：**

**第一步：**拷贝pinyougou\pinyougou-service\src\main\java\com\pinyougou

\service\目录下面全部的XxxService.java服务接口类到pinyougou-interface/src/main/java/com.pinyougou.service/包下。

**第二步：**pinyougou-sellergoods-service/src/main/java/com.pinyougou

.sellergoods.service.impl/BrandServiceImpl.java实现接口中的方法：

|  |
| --- |
| @Service(interfaceName=**"com.pinyougou.service.BrandService"**) */\*\* 上面指定接口名，产生服务名，不然会用代理类的名称 \*/* @Transactional **public class** BrandServiceImpl **implements** BrandService {  */\*\* 注入数据访问接口代理对象 \*/* @Autowired  **private** BrandMapper **brandMapper**;  @Override  **public void** save(Brand brand) {   }  @Override  **public void** update(Brand brand) {  }  @Override  **public void** delete(Serializable id) {  }  @Override  **public void** deleteAll(Serializable[] ids) {  }  @Override  **public** Brand findOne(Serializable id) {  **return null**;  }  @Override  **public** List<Brand> findAll() {  *// 开始分页* PageInfo<Brand> pageInfo = PageHelper.*startPage*(1, 10)  .doSelectPageInfo(**new** ISelect() {  @Override  **public void** doSelect() {  **brandMapper**.selectAll();  }  });  System.***out***.println(**"总记录数："** + pageInfo.getTotal());  System.***out***.println(**"总页数："** + pageInfo.getPages());  **return** pageInfo.getList();  }  @Override  **public** List<Brand> findByPage(Brand brand, **int** page, **int** rows) {  **return null**;  } } |

# 前端框架【AngularJS】

## 3.1 AngularJS简介

AngularJS  诞生于2009年，由Misko Hevery 等人创建，后为Google所收购。是一款优秀的前端JS框架，已经被用于Google的多款产品当中。AngularJS有着诸多特性，最为核心的是：MVC、模块化、自动化双向数据绑定、依赖注入等等。



官方网站：https://angularjs.org/

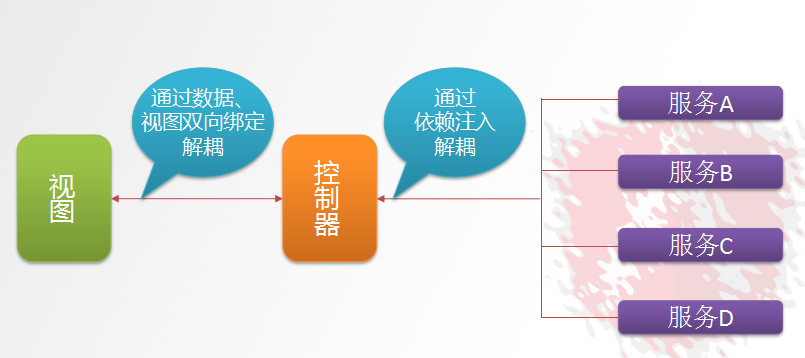
下载地址：<https://code.angularjs.org/>

学习版本：angular-1.6.9.zip(2018-02-02)

## 3.2 AngularJS四大特征

### **3.2.1 MVC 模式**

AngularJS遵循软件工程的MVC模式,并鼓励展现，数据，和逻辑组件之间的松耦合.通过依赖注入（dependency injection），AngularJS为客户端的Web应用带来了传统服务端的服务，例如独立于视图的控制。为后端减少了许多负担。



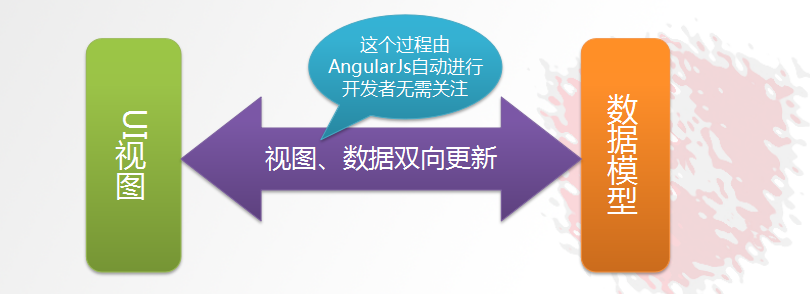
Model:数据,其实就是angular变量($scope.XX);

View: 数据的呈现,Html+Directive(指令);

Controller:操作数据,就是function,数据的增删改查;

### **3.2.2双向绑定**

AngularJS是建立在这样的信念上的：即声明式编程应该用于构建用户界面以及编写软件构建，而指令式编程非常适合来表示业务逻辑。框架采用并扩展了传统HTML，通过双向的数据绑定来适应动态内容，双向的数据绑定允许模型和视图之间的自动同步。因此，AngularJS使得对DOM的操作不再重要并提升了可测试性。



### **3.2.3依赖注入**

依赖注入(Dependency Injection,简称DI)是一种设计模式, 指某个对象依赖的其他对象无需手工创建，只需要“吼一嗓子”，则此对象在创建时，其依赖的对象由框架来自动创建并注入进来,其实就是最少知识法则，模块中所有的service和provider两类对象，都可以根据形参名称实现DI.

### **3.2.4模块化设计**

高内聚低耦合法则：

1)官方提供的模块  ng、ngRoute、ngAnimate

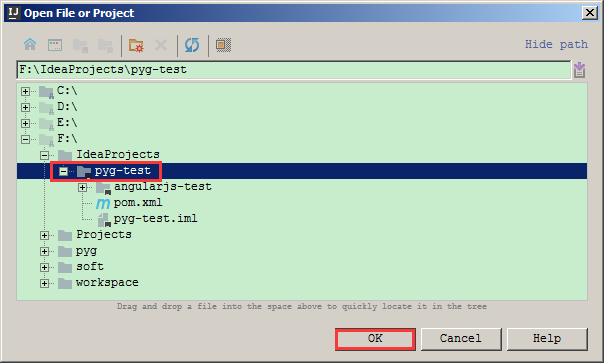
  2)用户自定义的模块 angular.module('模块名',[ ])

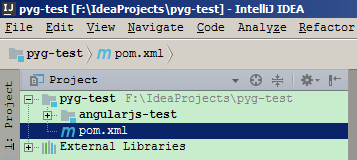
## 3.3基础指令

拷贝【资料\angularjs\pyg-test】测试项目到IdeaProjects目录下面：









拷贝【**资料\angularjs\angular.min.js**】到angularjs-test/src/main/webapp

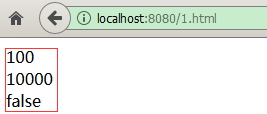
/js/目录下。

### **3.3.1表达式**

**语法格式：{{变量名}}|{{对象.变量名}}|{{表达式}}**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Angular表达式</title>  <meta charset=*"UTF-8"*/>  <meta http-equiv=*"pragma"* content=*"no-cache"*/>  **<script src=*"js/angular.min.js"*></script>**  </head>  <body **ng-app**>  {{100 + 100 - 100}}<br/>  {{100 \* 100}}<br/>  {{100 > 100}}<br/>  </body>  </html> |

执行结果如下：



说明：表达式可以是变量或运算式。

**ng-app 指令定义了 Angular应用的根元素，在根元素的所有子元素中用到指令，angular会自动识别。**

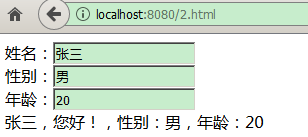
**ng-app 指令在网页加载完毕时会自动初始化应用中的angular的指令。**

### **3.3.2 双向绑定**

**语法格式：ng-model=”变量名”|ng-model=”对象.变量名”**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Angular双向绑定</title>  <meta charset=*"UTF-8"*/>  <meta http-equiv=*"pragma"* content=*"no-cache"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  </head>  <body ng-app>  姓名：<input type=*"text"* ng-model=*"name"*/><br/>  性别：<input type=*"text"* ng-model=*"user.sex"*/><br/>  年龄：<input type=*"text"* ng-model=*"user.age"*/><br/>  {{name}}，您好！，性别：{{user.sex}}，年龄：{{user.age}}  </body>  </html> |

运行效果如下：



说明：ng-model 指令用于绑定变量,这样用户在文本框输入的内容会绑定到变量上，而表达式可以实时地输出变量。

### **3.3.3 初始化指令**

**语法格式：ng-init=”变量名=’变量值’;变量名=’变量值’”**

使用ng-init指令来对变量初始化：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Angular变量初始化</title>  <meta charset=*"UTF-8"*/>  <meta http-equiv=*"pragma"* content=*"no-cache"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  </head>  <body ng-app ng-init=*"name='李小华';user.sex='男';user.age=18"*>  姓名：<input type=*"text"* ng-model=*"name"*/><br/>  性别：<input type=*"text"* ng-model=*"user.sex"*/><br/>  年龄：<input type=*"text"* ng-model=*"user.age"*/><br/>  {{name}}，您好！，性别：{{user.sex}}，年龄：{{user.age}}  </body>  </html> |

### **3.3.4 控制器**

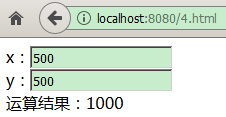
**定义模块语法格式：var 变量名 = angular.module(“模块名”, []);**

**定义控制器语法格式：**

**模块变量名.controller(“控制器名”, function($scope){});**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Angular控制器</title>  <meta charset=*"UTF-8"*/>  <meta http-equiv=*"pragma"* content=*"no-cache"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  <script type=*"text/javascript"*>  /\*\* 定义myApp模块 \*/  **var** myApp = angular.module("myApp",[]);  /\*\* 为myApp模块添加控制器 \*/  myApp.controller("myController", **function**($scope){  /\*\* 定义add方法 \*/  $scope.add = **function**(){  **return** parseInt($scope.x) + parseInt($scope.y);  };  });  </script>  </head>  <body ng-app=*"myApp"* ng-controller=*"myController"*>  x：<input type=*"text"* ng-model=*"x"*/><br/>  y：<input type=*"text"* ng-model=*"y"*/><br/>  运算结果：{{add()}}  </body>  </html> |

运行结果如下：



ng-controller指定所使用的控制器。

理解 $scope作用域：

$scope 贯穿整个 Angular App应用,它与数据模型相关联,同时也是表达式执行的上下文.有了$scope相当于在视图和控制器之间建立了一个通道,基于作用域视图在修改数据时会立刻更新 $scope,同样的$scope 发生改变时也会立刻重新渲染视图。

注意：mg-model绑定的变量数据都存放在$scope作用域对象中。

### **3.3.5 事件指令**

* ng-click：单击事件
* ng-blur：失去焦点事件
* ng-focus：获取焦点事件
* ng-change：对应onchange改变事件
* ng-keydown：键盘按键按下事件
* ng-keyup：键盘按键按下并松开
* ng-mousedown：鼠标按下事件
* ng-mouseup：鼠标按下弹起
* ng-mousemove：鼠标移动事件
* ng-mouseover：鼠标进入事件

**语法格式：ng-xxx=”控制器中$scope中定义的方法名();”;**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Angular事件指令</title>  <meta charset=*"UTF-8"*/>  <meta http-equiv=*"pragma"* content=*"no-cache"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  <script type=*"text/javascript"*>  /\*\* 定义模块 \*/  **var** myApp = angular.module("myApp",[]);  /\*\* 模块中添加控制器 \*/  myApp.controller("myController", **function**($scope){  /\*\* 定义方法 \*/  $scope.add = **function**(){  $scope.count = parseInt($scope.x) +  parseInt($scope.y);  };  /\*\* 定义方法 \*/  $scope.blur = **function**(){  alert($scope.x);  };  /\*\* 定义方法 \*/  $scope.keyup = **function**(){  alert($scope.y);  };  });  </script>  </head>  <body ng-app=*"myApp"* ng-controller=*"myController"*>  x：<input type=*"text"* ng-model=*"x"* ng-blur=*"blur();"*/><br/>  y：<input type=*"text"* ng-model=*"y"* ng-keyup=*"keyup()"*/><br/>  <input type=*"button"* value=*"计算"* ng-click=*"add();"*/>  运算结果：{{count}}  </body>  </html> |

运行结果：



说明：ng-xxx事件指令，绑定控制器的某个方法。

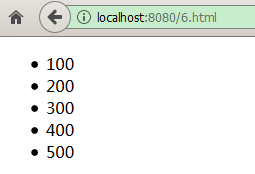
### **3.3.6 循环数组**

**语法格式：ng-repeat=”变量名 in 集合或数组”;**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Angular循环数组</title>  <meta charset=*"UTF-8"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  <script type=*"text/javascript"*>  /\*\* 定义模块 \*/  **var** myApp = angular.module("myApp",[]);  /\*\* 模块中添加控制器 \*/  myApp.controller("myController", **function**($scope){  /\*\* 定义数组 \*/  $scope.list = [100,200,300,400,500];  });  </script>  </head>  <body ng-app=*"myApp"* ng-controller=*"myController"*>  <ul>  <li ng-repeat=*"item in list"*>  {{item}}  </li>  </ul>  </body>  </html> |

说明：这里的ng-repeat指令用于循环数组变量。

运行结果如下：

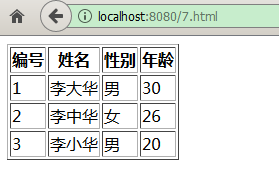


### **3.3.7 循环对象数组**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Angular循环对象数组</title>  <meta charset=*"UTF-8"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  <script type=*"text/javascript"*>  /\*\* 定义模块 \*/  **var** myApp = angular.module("myApp",[]);  /\*\* 模块中添加控制器 \*/  myApp.controller("myController", **function**($scope){  /\*\* 定义对象数组 \*/  $scope.list =[  {name : "李大华", sex : "男", age : 30},  {name : "李中华", sex : "女", age : 26},  {name : "李小华", sex : "男", age : 20}];  });  </script>  </head>  <body ng-app=*"myApp"* ng-controller=*"myController"*>  <table border=*"1"*>  <tr>  <th>编号</th>  <th>姓名</th>  <th>性别</th>  <th>年龄</th>  </tr>  <tr ng-repeat=*"u in list"*>  <td>{{$index + 1}}</td>  <td>{{u.name}}</td>  <td>{{u.sex}}</td>  <td>{{u.age}}</td>  </tr>  </table>  </body>  </html> |

**$index：获取迭代时的索引号。**

运行结果如下：

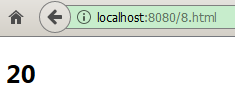


### **3.3.8 条件指令**

**语法格式：ng-if=”条件表达式”;**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>Angular条件指令</title>  <meta charset=*"UTF-8"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  <script type=*"text/javascript"*>  /\*\* 定义模块 \*/  **var** myApp = angular.module("myApp",[]);  /\*\* 模块中添加控制器 \*/  myApp.controller("myController", **function**($scope){  /\*\* 定义对象 \*/  $scope.user = {age : 20};  });  </script>  </head>  <body ng-app=*"myApp"* ng-controller=*"myController"*>  <h2 ng-if=*"user.age <= 20"*>{{user.age}}</h2>  </body>  </html> |

运行效果如下：



### **3.3.9下拉列表框**

* ng-options="元素变量.键 as 元素变量.键 for 元素变量in 数组"

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>select</title>  <meta charset=*"UTF-8"*/>  <meta http-equiv=*"pragma"* content=*"no-cache"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  <script type=*"text/javascript"*>  /\*\* 定义模块 \*/  **var** myApp = angular.module("myApp",[]);  /\*\* 模块中添加控制器 \*/  myApp.controller("myController", **function**($scope){  // 定义下拉列表框需要的数据变量  $scope.cities = [{id:1, name:'广州市'},  {id:2, name:'深圳市'},  {id:3, name:'珠海市'},  {id:4, name:'东莞市'}];    });  </script>  </head>  <body ng-app=*"myApp"* ng-controller=*"myController"*>  <select ng-model=*"code"*  ng-options=*"city.id as city.name for city in cities"*>  <option value=*""*>==请选择城市==</option>  </select>  {{code}}  </body>  </html> |

运行结果如下：



## 3.4内置服务

### **3.4.1 $http异步请求服务**

我们的数据一般都是从后端获取的，那么如何获取数据呢？我们一般使用内置服务$http服务来实现。

#### 3.4.1.1 $http()发送get|post请求

**语法格式：**

|  |
| --- |
| $http({  method : 'get|post', // 请求方式  url : '', // 请求URL  params : {'name':'admin'} // 请求参数  }).then(**function**(response){ // 请求成功  // response: 响应对象封装了响应数据、状态码  },**function**(response){ // 请求失败  // response: 响应对象封装了响应状态码  }); |

发送异步请求：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>$http</title>  <meta charset=*"UTF-8"*/>  <meta http-equiv=*"pragma"* content=*"no-cache"*/>  <script type=*"text/javascript"* src=*"js/angular.min.js"*></script>  <script type=*"text/javascript"*>  /\*\* 定义模块 \*/  **var** myApp = angular.module("myApp",[]);  /\*\* 模块中添加控制器 \*/  myApp.controller("myController", **function**($scope, $http){  // 发送异步请求  $scope.httpGet = **function**(){  $http({  method : "GET", // 请求方式  url : "/find", // 请求URL  params : {"id":10} // 请求参数  }).then(**function**(response){ // 请求成功  // response响应对象  $scope.user = response.data;  },**function**(response){ // 请求失败  alert("数据加载失败！");  });  };  });  </script>  </head>  <body ng-app=*"myApp"* ng-controller=*"myController"*  ng-init=*"httpGet();"*>  {{user.id}} ---> {{user.name}}  </body>  </html> |

angular-test项目中TestController.java：

|  |
| --- |
| @RestController  **public** **class** TestController {  /\*\* 处理get请求与post请求 \*/  @RequestMapping("/find")  **public** Map<String,String> find(@RequestParam("id")Long id){  System.***out***.println("请求参数id: " + id);  Map<String,String> user = **new** HashMap<>();  user.put("id", "1");  user.put("name", "李小华");  **return** user;  }  } |

说明：再把异步请求方式改成post请求测试，效果完全一样。

#### 3.4.1.2 $http.get()发送get请求

**语法格式：**

|  |
| --- |
| // 第一种格式  $http.get(URL,{  params: {  "id":id  }  }).then(**function**(response){// 请求成功  // response: 响应对象封装了响应数据、状态码  }, **function**(response){ // 请求失败  // response: 响应对象封装了响应状态码  });  // 第二种格式  $http.get(URL).then(**function**(response){ // 请求成功  // response: 响应对象封装了响应数据、状态码  },**function**(response){ // 请求失败  // response: 响应对象封装了响应状态码  }); |

发送get请求：

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <title>$http.get()请求</title>  <meta charset=*"UTF-8"*/>  <meta http-equiv=*"pragma"* content=*"no-cache"*/>  <script src=*"js/angular.min.js"*></script>  <script type=*"text/javascript"*>  /\*\* 定义模块 \*/  **var** myApp = angular.module("myApp",[]);  /\*\* 模块中添加控制器 \*/  myApp.controller("myController", **function**($scope, $http){  // 发送异步请求  $scope.sendGet = **function**(){  $http.get("/data.json").then(  **function**(response){ // 请求成功  // response响应对象，封装了响应数据、状态码等  $scope.list = response.data;  },**function**(response){ // 请求失败  alert("数据加载失败！");  }  );  };  });  </script>  </head>  <body ng-app=*"myApp"* ng-controller=*"myController"*  ng-init=*"sendGet()"*>  <table border=*"1"*>  <tr>  <td>姓名</td>  <td>性别</td>  <td>年龄</td>  </tr>  <tr ng-repeat=*"user in list"*>  <td>{{user.name}}</td>  <td>{{user.sex}}</td>  <td>{{user.age}}</td>  </tr>  </table>  </body>  </html> |

创建文件 data.json：

|  |
| --- |
| [{"name" : "李大华", "sex" : "男", "age" : 30},  {"name" : "李中华", "sex" : "女", "age" : 26},  {"name" : "李小华", "sex" : "男", "age" : 20}] |

#### 3.4.1.3 $http.post()发送post请求

**语法格式：**

|  |
| --- |
| // 第一种方式  $http.post(URL,{  "id" : id  }).then(**function**(response){ // 请求成功  // response: 响应对象封装了响应数据、状态码  },**function**(response){ // 请求失败  // response: 响应对象封装了响应状态码  }); |

发送异步请求：

|  |
| --- |
| <!DOCTYPE **html**> <**html**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Angular $http.post()方法</**title**>  <**meta name="Copyright" content="All Rights Reserved"**/>  <**script src="js/angular.min.js"**></**script**>  <**script type="text/javascript"**>  *// 定义一个模块* **var *app*** = **angular**.module(**'myApp'**,[]);  *// 模块添加控制器层* ***app***.controller(**'myController'**, **function**($scope, $http){  $scope.sendPost = **function**(){  *// $http.get()发送异步请求* $http.post(**"/user"**, {**"name"** : **"张三"**})  .then(**function**(response){ *// 请求成功* **if**(response.status == 200){  $scope.**res** = response.**data**;  }  }, **function**(response){ *// 请求失败  alert*(**"数据加载失败！"**);  });  };  });  </**script**> </**head**> <**body ng-app="myApp" ng-controller="myController"  ng-init="sendPost();"**>  {{**res**.**msg**}} </**body**> </**html**> |

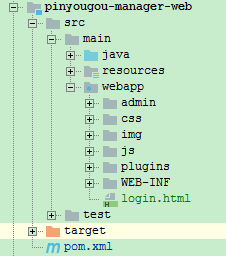
TestController.java控制器：

|  |
| --- |
| */\*\* $http.post()请求 \*/* @PostMapping(**"/user"**) **public** Map<String,String> user(@RequestBody Map<String,String> map){  System.***out***.println(**"请求参数name: "** + map.get(**"name"**));  Map<String,String> data = **new** HashMap<>();  data.put(**"msg"**, map.get(**"name"**) + **"，您好！"**);  **return** data; } |

# 使用域名访问后台系统

## 4.1拷贝资源

将“资料\运营商管理后台静态资源”下的全部文件拷贝到pinyougou-manager-web的src/main/webapp/目录下。



其中plugins文件夹中包括了angularJS 、bootstrap、jQuery等常用前端库，我们将在项目中用到。

**品优购-运营商后台请求地址：<http://localhost:9101/admin/index.html>**

## 4.2问题

现在访问的路径：http://127.0.0.1:9101/admin/index.html

存在的问题：

1. 开发环境和测试环境的ip不一样，每次环境变化时，都需要修改访问地址；
2. 页面加载资源文件，有可能使用url的全路径，一旦更换环境（ip变了），资源文件则无法加载；
3. ip地址没有记忆意义，不容易记忆，用户不通过ip访问，一般通过域名访问。

## 4.3域名访问

可以修改windows/system32/drivers/etc/hosts文件，实现域名和ip地址的映射。

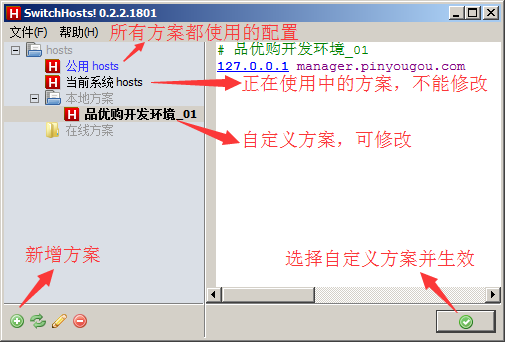
使用域名访问后的流程为：



## 4.4 SwitchHosts工具

手动修改hosts文件，如果以后ip多了，配置和切换都比较麻烦；一般可以使用工具进行管理。而SwitchHosts则可以管理hosts文件；SwitchHosts安装文件在“资料\SwitchHosts.exe”中。

功能说明：



问题：ip不用输入了，但是端口还在。可使用nginx解决。

# 5.Nginx反向代理服务器

## 5.1 Nginx 简介

* Nginx ("engine x") 是一个高性能的HTTP和反向代理服务器；
* 可实现负载均衡；
* Rewrite功能强大；
* 电商架构大部分都采用Nginx + Tomcat的架构。

反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器。

## 5.2安装与命令

拷贝资料中“nginx-1.11.13.zip”到任意一个磁盘路径解压，解压路径不能包含中文。

用cmd进入nginx解压所在的根目录：

* 启动： start nginx
* 停止：nginx -s stop
* 重新启动：nginx -s reload

注意：启动nginx后在任务管理器中要能看到两个nginx服务启动才是正确的。

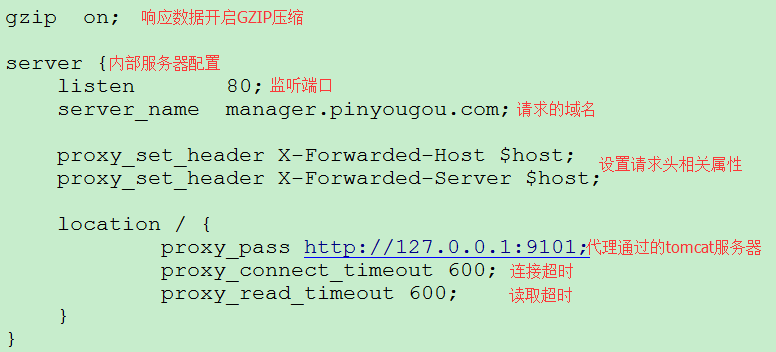


如果启动报错,查看日志文件发现80端口被占用；可以使用命令netstat -ano命令查看端口占用情况。再打开任务管理器，查看pid然后停止程序即可。

netstat -ano|findstr "80"

## 5.3相关配置

修改“nginx-1.11.13\conf\nginx.conf”文件，配置内部服务器：



对于运营商后台管理系统，可以添加如下nginx配置：

|  |
| --- |
| server {  listen 80;  server\_name manager.pinyougou.com;  proxy\_set\_header X-Forwarded-Host $host;  proxy\_set\_header X-Forwarded-Server $host;  location / {  proxy\_pass http://127.0.0.1:9101;  proxy\_connect\_timeout 600;  proxy\_read\_timeout 600;  }  } |

**请求地址：<http://manager.pinyougou.com/admin/index.html>**

**参考：【资料\Nginx反向代理流程图.png】**

# 6.品牌管理【品牌列表查询】

## 6.1需求分析

实现品牌列表的查询（不用分页和条件查询）效果如下：



## 6.2前端代码

### **6.2.1引入JS**

修改pinyougou-manager-web/src/main/webapp/admin/brand.html，引入js：

|  |
| --- |
| <**script src="/plugins/angularjs/angular.min.js"**></**script**> |

### **6.2.2指定模块和控制器**

|  |
| --- |
| <**body class="hold-transition skin-red sidebar-mini"  ng-app="pinyougou" ng-controller="brandController"**> |

ng-app 指令中定义的就是模块的名称

ng-controller 指令用于为你的应用添加控制器。

在控制器中，你可以编写代码，制作函数和变量，并使用 $scope 对象来访问。

### **6.2.3编写JS代码**

|  |
| --- |
| <**script type="text/javascript"**>  */\*\* 定义模块 \*/* **var *app*** = **angular**.**module**(**"pinyougou"**,[]);  */\*\* 添加控制器 \*/* ***app***.controller(**"brandController"**, **function**($scope, $http){  */\*\* 读取列表数据绑定到表格中 \*/* $scope.findAll = **function**(){  */\*\* 发送异步请求查询数据 \*/* $http.get(**"/brand/findAll"**).then(**function**(response){  $scope.**dataList** = response.data;  });  };  }); </**script**> |

### **6.2.4初始化调用**

|  |
| --- |
| <**body class="hold-transition skin-red sidebar-mini"  ng-app="pinyougou" ng-controller="brandController"  ng-init="findAll();"**> |

### **6.2.5循环显示表格数据**

|  |
| --- |
| <**tbody**>  <**tr ng-repeat="entity in dataList"**>  <**td**><**input type="checkbox"** ></**td**>   <**td**>{{entity.**id**}}</**td**>  <**td**>{{entity.**name**}}</**td**>  <**td**>{{entity.**firstChar**}}</**td**>  <**td class="text-center"**>   <**button type="button" class="btn bg-olive btn-xs"**  **data-toggle="modal" data-target="#editModal"**>修改</**button**>   </**td**>  </**tr**> </**tbody**> |

## 6.3后端代码

### **6.3.1控制器层**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller的BrandController.java：



### **6.3.2服务实现层**

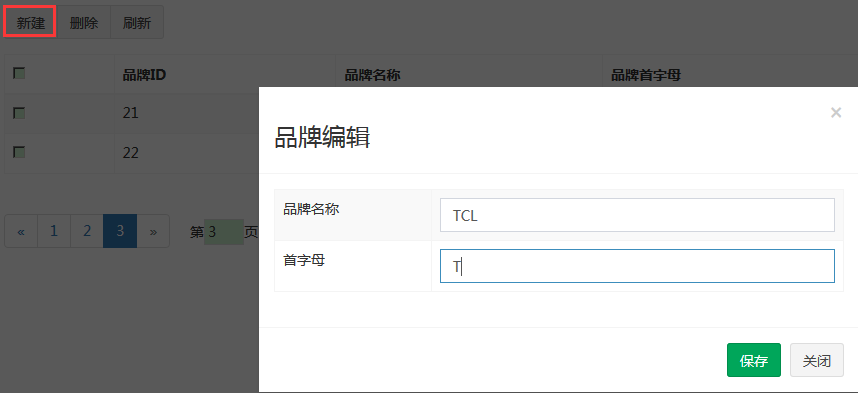
pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl的BrandServiceImpl.java：

|  |
| --- |
| @Override **public** List<Brand> findAll() {  **return brandMapper**.selectAll(); } |

# 7.品牌管理【增加品牌】

## 7.1需求分析

实现品牌增加功能



## **7.2前端代码**

### **7.2.1页面**

绑定表单元素，我们用ng-model指令，绑定按钮的单击事件我们用ng-click

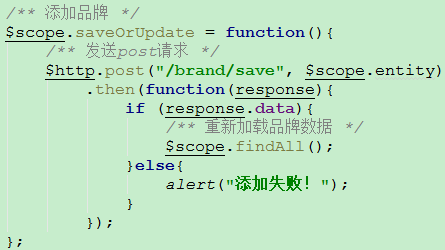
|  |
| --- |
| <**div class="modal-body"**>   <**table class="table table-bordered table-striped" width="800px"**>  <**tr**>  <**td**>品牌名称</**td**>  <**td**>  <**input class="form-control"  ng-model="entity.name" placeholder="品牌名称"**/>  </**td**>  </**tr**>   <**tr**>  <**td**>首字母</**td**>  <**td**>  <**input class="form-control"  ng-model="entity.firstChar" placeholder="首字母"**/>  </**td**>  </**tr**>   </**table**>  </**div**> <**div class="modal-footer"**>   <**button class="btn btn-success" data-dismiss="modal"  aria-hidden="true" ng-click="saveOrUpdate();"**>保存</**button**>  <**button class="btn btn-default" data-dismiss="modal"  aria-hidden="true"**>关闭</**button**> </**div**> |

为了每次打开窗口没有遗留上次的数据，我们可以修改新建按钮，对entity变量进行清空操作：

|  |
| --- |
| <**button type="button" class="btn btn-default" title="新建"  data-toggle="modal" data-target="#editModal"  ng-click="entity={}"**>  <**i class="fa fa-file-o"**></**i**>新建 </**button**> |

### **7.2.2 JS代码**

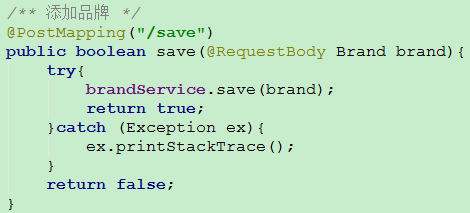
pinyougou-manager-web/src/main/webapp/admin/brand.html添加js代码：



## 7.3后端代码

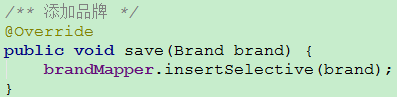
### **7.3.1控制器层**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller的BrandController.java新增方法：



### **7.3.2服务实现层**

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl的BrandServiceImpl.java实现方法：



# 8.品牌管理【修改品牌】

## 8.1 需求分析

点击列表的修改按钮，弹出窗口，修改数据后点“保存”执行保存操作



## **8.2 前端代码**

### **8.2.1实现数据绑定**

在pinyougou-manager-web/src/main/webapp/admin/brand.html页面中，增加JS代码显示修改窗口：

|  |
| --- |
| */\*\* 为修改按钮绑定点击事件 \*/* $scope.show = **function**(entity){  *// 把entity的json对象转化成一个新的json对象* $scope.**entity** = ***JSON***.parse(***JSON***.stringify(entity));  }; |

修改列表中的“修改”按钮，调用此方法执行表单数据绑定：

|  |
| --- |
| <**button type="button" class="btn bg-olive btn-xs"  data-toggle="modal"  data-target="#editModal"  ng-click="show(entity);"**>修改</**button**> |

### **8.2.2保存数据**

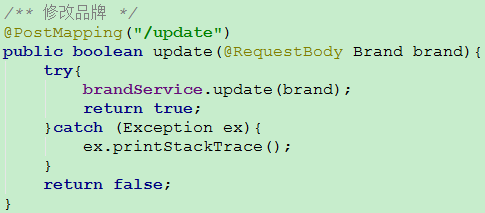
修改JS的$scope.saveOrUpdate方法：

|  |
| --- |
| */\*\* 添加或修改 \*/* $scope.saveOrUpdate = **function**(){  */\*\* 定义请求URL \*/* **var** url = **"save"**; *// 添加品牌* **if** ($scope.**entity**.**id**){  url = **"update"**; *// 修改品牌* }  */\*\* 发送post请求 \*/* $http.**post**(**"/brand/"** + url, $scope.**entity**)  .then(**function**(response){  **if** (response.data){  */\*\* 重新加载品牌数据 \*/* $scope.findAll();  }**else**{  *alert*(**"操作失败！"**);  }  }); }; |

## 8.3 后端代码

### **8.3.1控制器层**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller的BrandController.java新增方法：



### **8.3.2服务实现层**

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl的BrandServiceImpl.java实现方法：

