



品优购电商系统开发

第2章

品牌管理

传智播客·黑马程序员

课程目标

目标 1: 运用 AngularJS 前端框架的常用指令

目标 2: 完成品牌管理的功能（增删改查）

1.前端框架 AngularJS 入门

1.1 AngularJS 简介

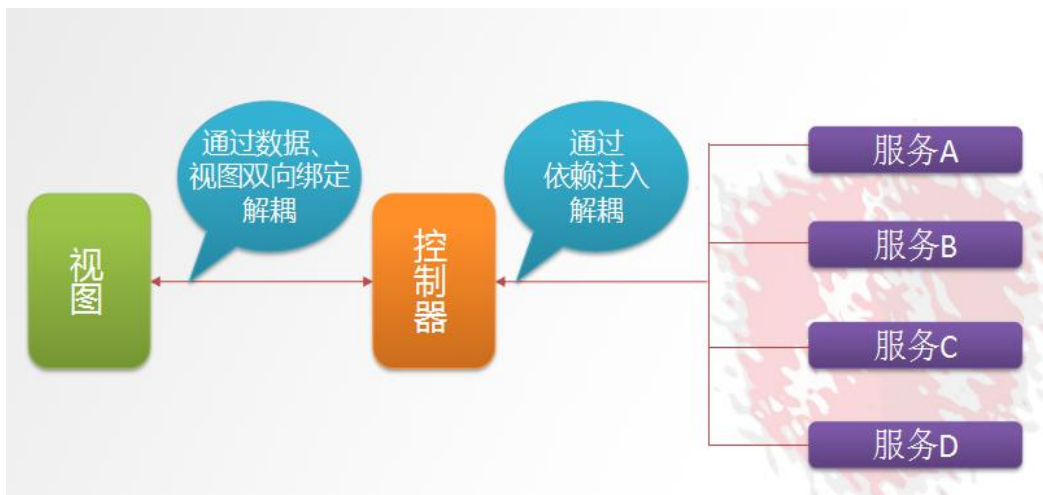
AngularJS 诞生于 2009 年，由 Misko Hevery 等人创建，后为 Google 所收购。是一款优秀的前端 JS 框架，已经被用于 Google 的多款产品当中。AngularJS 有着诸多特性，最为核心的是：MVC、模块化、自动化双向数据绑定、依赖注入等等。



1.2 AngularJS 四大特征

1.2.1 MVC 模式

Angular 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），Angular 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。因此，后端减少了许多负担，产生了更轻的 Web 应用。



Model:数据,其实就是 angular 变量(\$scope.XX);

View: 数据的呈现,Html+Directive(指令);(眼睛能看见的,基本上是视图)

Controller:操作数据,就是 function,数据的增删改查(业务逻辑,控制逻辑)

mvc 分层结构对比:

```
<script type="text/javascript">

    function HelloController($scope) {
        $scope.greeting = {
            text : "hello" 模型(数据)
        };
    }

</script>
</head>
<body ng-app>
    <div ng-controller="HelloController">
        <p>{{greeting.text}},angular!</p>
    </div>
</body>
```

控制器

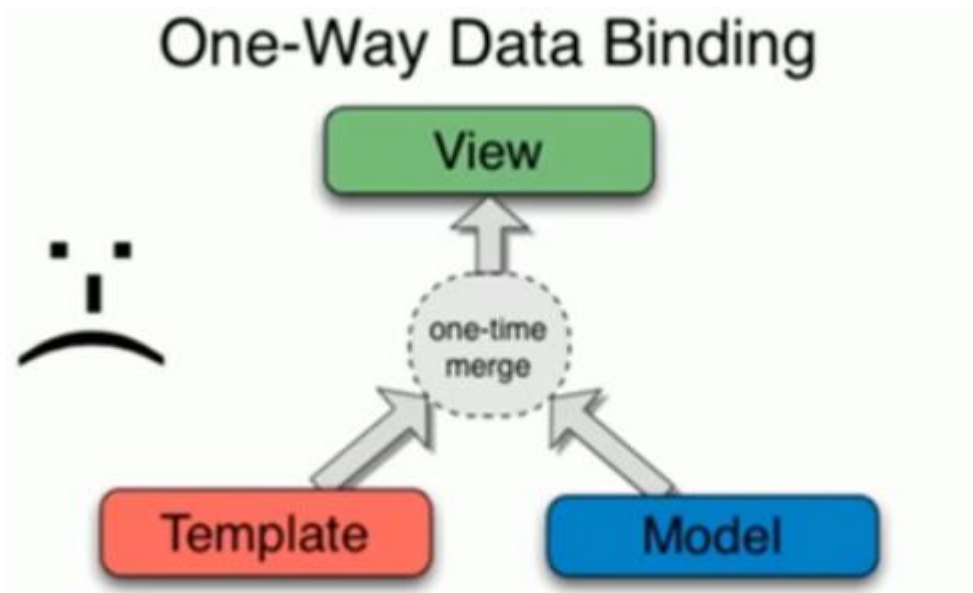
视图

1.2.2 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。

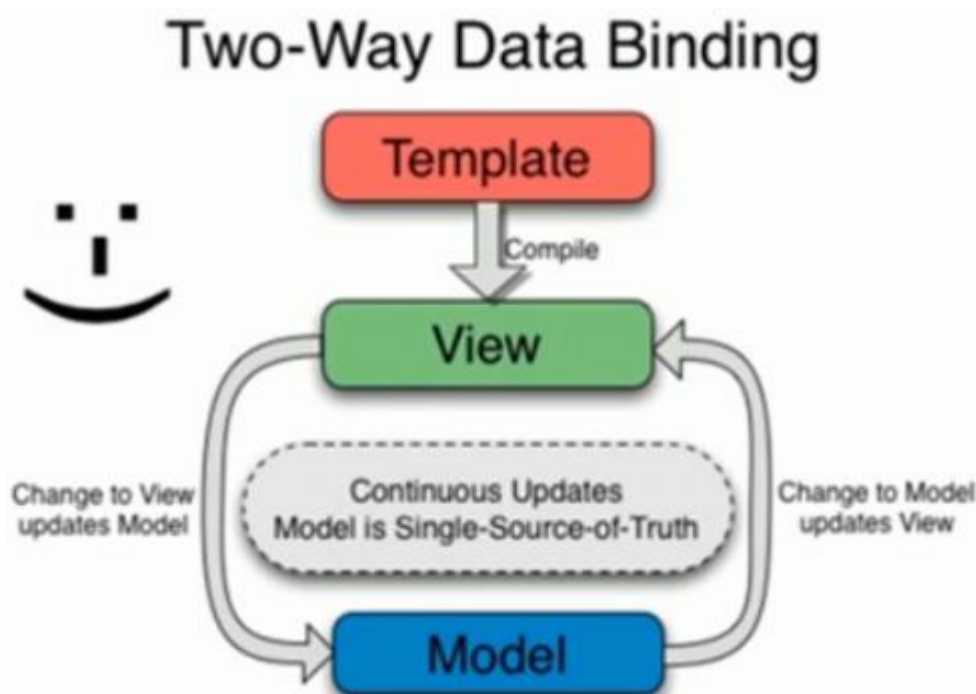


单向绑定:



单向绑定使用模版+模型数据结合,通过数据绑定机制生成视图,那么他的缺点就是有新的数据生成,html 就没法发改变了,只能重新进行.

双向绑定:



数据双向绑定,当模型数据发生改变以后,angular.js 数据绑定机制会自动改变更新视图.

1.2.3 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.

1.2.4 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',[])

模块设计实例:

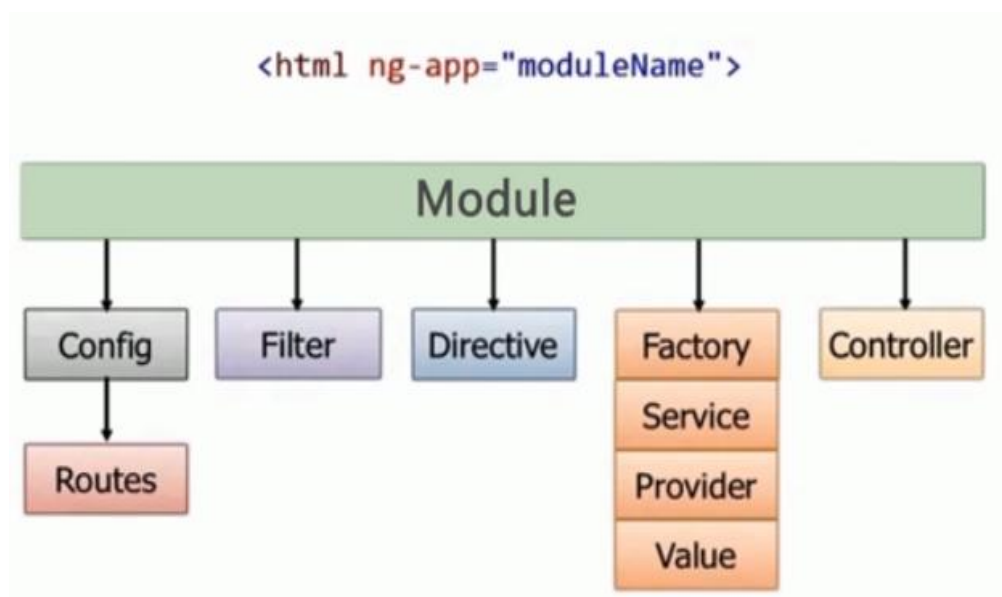
```
<script type="text/javascript">
    //定义一个叫oneModule名称的模块
    var myModule = angular.module('oneModule',[]);
    //使用模块调用方法
    myModule.controller('helloController',function($scope){
        $scope.greeting = function(){
            return "hello,angular!";
        }
    })
</script>
```

相当于java语言的主函数
每个页面只能有一个

```
<body ng-app="oneModule">
    <div ng-controller="helloController">
        <p>{{greeting()}}</p>
    </div>
</body>
```

调用模块中函数

模块示意图:





1.3 AngularJS 快速入门

1.3.1 表达式

```
<html>

<head>

  <title>入门小 Demo-1</title>

  <script src="angular.min.js"></script>

</head>

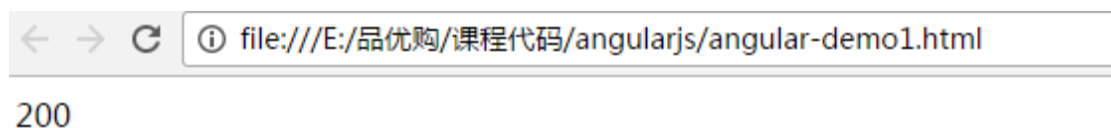
<body ng-app>

  {{100+100}}

</body>

</html>
```

执行结果如下：



表达式的写法是{{表达式 }} 表达式可以是变量或是运算式

ng-app 指令 作用是告诉子元素一下的指令是归 angularJs 的,angularJs 会识别的

ng-app 指令定义了 AngularJS 应用程序的 根元素。

ng-app 指令在网页加载完毕时会自动引导（自动初始化）应用程序。

1.3.2 双向绑定

```
<html>
```



```
<head>

  <title>入门小 Demo-1 双向绑定</title>

  <script src="angular.min.js"></script>

</head>

<body ng-app>

请输入你的姓名: <input ng-model="myname">

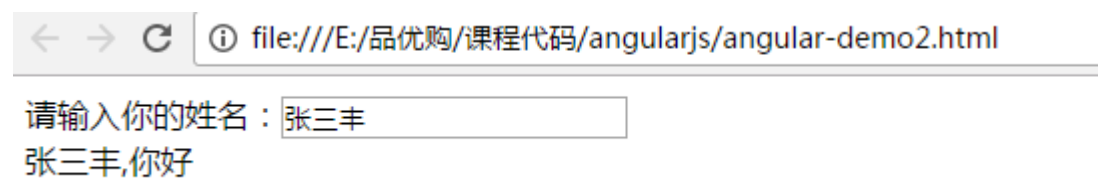
<br>

{{myname}},你好

</body>

</html>
```

运行效果如下:



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。

1.3.3 控制器

```
<html>

<head>

  <title>入门小 Demo-3 初始化</title>

  <script src="angular.min.js"></script>

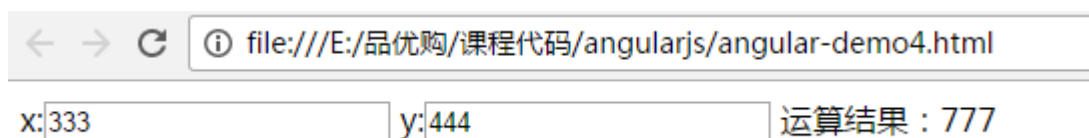
  <script type="text/javascript">

    //定义函数
```



```
function helloController($scope) {  
  
    $scope.greeting = {  
  
        text : "hello"  
  
    };  
  
}  
  
</script>  
  
</head>  
  
<body ng-app>  
  
    <div ng-controller="helloController">  
  
        <p>{{greeting.text}},angular!</p>  
  
    </div>  
  
</body>  
  
</html>
```

运行结果如下:



ng-controller 用于指定所使用的控制器

理解 \$scope:

\$scope 的使用贯穿整个 AngularJS App 应用,它与数据模型相关联,同时也是表达式执行的上文.有了\$scope 就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.



1.3.4 mvc 模式

Angular 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,提高代码的复用性.angularJS 为后端减轻了很多业务压力,带来了更轻的 web 应用.

那么在 AngularJS 的应用中,怎么提现 mvc 设计模式的呢?下面来看以下代码展示 AngularJS 的应用.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="js/angular.min.js"></script>
<script type="text/javascript">
    //定义函数
    function helloController($scope) {
        $scope.greeting = {
            text : "hello" //模型层
        };
    }
</script>
</head>
<body ng-app>
    <div ng-controller="helloController"> //控制层
        <p>{{greeting.text}},angular!</p> //视图层
    </div>
</body>
</html>
```

1.3.5 模块化设计

js 编程通常把 js 都放在全局环境中,那么这些变量函数,或者是对象都会产生在 window 对象下,会污染全局空间.为了更好管理 js,我们使用模块化的开发方式.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<script type="text/javascript" src="js/angular.min.js"></script>
<script type="text/javascript">
    //定义一个叫oneModule名称的模块
```



```
var myModule = angular.module('oneModule',[]);
//使用模块调用方法
myModule.controller('helloController',function($scope){
    $scope.greeting = function(){

        return "hello,angular!";
    }
})

</script>
</head>
<body ng-app="oneModule">
    <div ng-controller="helloController">
        <p>{{greeting()}}</p>
    </div>
</body>
</html>
```

1.3.6 初始化指令

我们如果希望有些变量具有初始值，可以使用 ng-init 指令来对变量初始化

```
<html>

<head>

    <title>入门小 Demo-3  初始化</title>

    <script src="angular.min.js"></script>

</head>

<body ng-app    ng-init="myname='陈大海'">

    请输入你的姓名: <input ng-model="myname">

    <br>

    {{myname}},你好

</body>

</html>
```



1.3.7 事件指令

```
<html>

<head>

  <title>入门小 Demo-5  事件指令</title>

  <script src="angular.min.js"></script>

  <script>

    var app=angular.module('myApp',[]); //定义了一个叫 myApp 的模块

    //定义控制器

    app.controller('myController',function($scope){

      $scope.add=function(){

        $scope.z= parseInt($scope.x)+parseInt($scope.y);

      }

    });

  </script>

</head>

<body ng-app="myApp" ng-controller="myController">

x:<input ng-model="x" >

y:<input ng-model="y" >

<button ng-click="add()">运算</button>

结果: {{z}}

</body>

</html>
```

运行结果：



← → ↺ 🔍 file:///E:/品优购/课程代码/angularjs/angular-demo5.html

x:

y:

运算

结果：33

ng-click 是最常用的单击事件指令，再点击时触发控制器的某个方法

1.3.8 循环数组

```
<html>

<head>

  <title>入门小 Demo-6  循环数据</title>

  <script src="angular.min.js"></script>

  <script>

    var app=angular.module('myApp',[]); //定义了一个叫 myApp 的模块

    //定义控制器

    app.controller('myController',function($scope){

      $scope.list= [100,192,203,434 ]; //定义数组

    });

  </script>

</head>

<body ng-app="myApp" ng-controller="myController">

<table>

<tr ng-repeat="x in list">

  <td>{{x}}</td>

</tr>

</table>
```

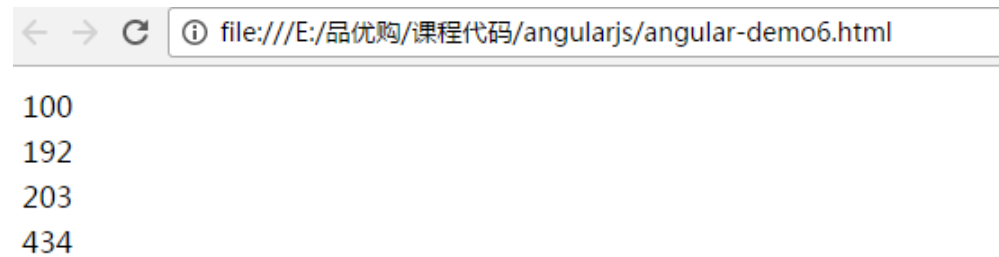


```
</body>

</html>
```

这里的 `ng-repeat` 指令用于循环数组变量。

运行结果如下：



```
100
192
203
434
```

1.3.9 循环对象数组

```
<html>

<head>

  <title>入门小 Demo-7  循环对象数组</title>

  <script src="angular.min.js"></script>

  <script>

    var app=angular.module('myApp',[]); //定义了一个叫 myApp 的模块

    //定义控制器

    app.controller('myController',function($scope){

      $scope.list= [

        {name:'张三',shuxue:100,yuwen:93},

        {name:'李四',shuxue:88,yuwen:87},

        {name:'王五',shuxue:77,yuwen:56}

      ]; //定义数组
```



```
});

</script>

</head>

<body ng-app="myApp" ng-controller="myController">

<table>

<tr>

<td>姓名</td>

<td>数学</td>

<td>语文</td>

</tr>

<tr ng-repeat="entity in list">

<td>{{entity.name}}</td>

<td>{{entity.shuxue}}</td>

<td>{{entity.yuwen}}</td>

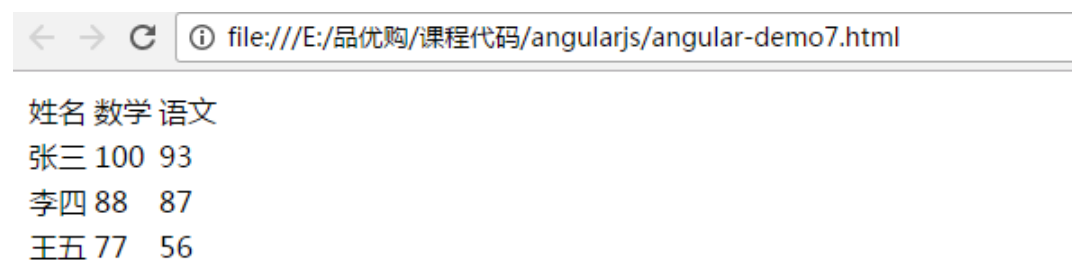
</tr>

</table>

</body>

</html>
```

运行结果如下：





1.3.10 内置服务

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 来实现。注意：以下代码需要在 tomcat 中运行。

```
<html>

<head>

  <title>入门小 Demo-8  内置服务</title>

  <meta charset="utf-8" />

  <script src="angular.min.js"></script>

  <script>

    var app=angular.module('myApp',[]); //定义了一个叫 myApp 的模块

    //定义控制器

    app.controller('myController',function($scope,$http){

      $scope.findAll=function(){

        $http.get('data.json').success(

          function(response){

            $scope.list=response;

          }

        );

      }

    });

  </script>

</head>

<body ng-app="myApp" ng-controller="myController" ng-init="findAll()">
```



```
<table>

<tr>

    <td>姓名</td>

    <td>数学</td>

    <td>语文</td>

</tr>

<tr ng-repeat="entity in list">

    <td>{{entity.name}}</td>

    <td>{{entity.shuxue}}</td>

    <td>{{entity.yuwen}}</td>

</tr>

</table>

</body>

</html>
```

建立文件 data.json

```
[

    {"name": "张三", "shuxue": 100, "yuwen": 93},

    {"name": "李四", "shuxue": 88, "yuwen": 87},

    {"name": "王五", "shuxue": 77, "yuwen": 56},

    {"name": "赵六", "shuxue": 67, "yuwen": 86}

]
```


2.品牌列表的实现

2.1 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：

品牌管理

新建	删除	刷新	品牌名称： <input type="text"/>	查询
<input type="checkbox"/>	品牌ID	品牌名称	品牌首字母	操作
<input type="checkbox"/>	1	联想	L	修改
<input type="checkbox"/>	2	华为	H	修改
<input type="checkbox"/>	3	三星	S	修改
<input type="checkbox"/>	4	小米	X	修改
<input type="checkbox"/>	5	OPPO	O	修改
<input type="checkbox"/>	6	360	S	修改
<input type="checkbox"/>	7	中兴	Z	修改

2.2 前端代码

2.2.1 拷贝资源

将“资源/静态原型/运营商管理后台”下的页面资源拷贝到 pinyougou-manager-web 下

品优购	资源	静态原型	运营商管理后台
帮助(H)			
新建文件夹			
名称	修改日期	类型	大小
admin	2017/7/29 10:57	文件夹	
css	2017/7/29 18:22	文件夹	
img	2017/7/29 18:23	文件夹	
js	2017/7/29 18:33	文件夹	
plugins	2017/7/28 17:26	文件夹	
login.html	2017/7/31 9:33	Chrome HTML D...	4 KB

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



2.2.2 引入 JS

修改 brand.html ， 引入 JS

```
<script type="text/javascript" src="../../plugins/angularjs/angular.min.js"></script>
```

2.2.3 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。

2.2.4 编写 JS 代码

```
var app=angular.module('pinyougou', []);//定义模块

app.controller('brandController' ,function($scope,$http){

    //读取列表数据绑定到表单中

    $scope.findAll=function(){

        $http.get('../brand/findAll.do').success(

            function(response){

                $scope.list=response;

            }

        );

    }

});
```



2.2.5 循环显示表格数据

```
<tbody>

  <tr ng-repeat="entity in list">

    <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>
```

2.2.6 初始化调用

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

3.品牌列表分页的实现

3.1 需求分析

在品牌管理下方放置分页栏，实现分页功能



品牌管理

新建	删除	刷新	品牌名称: <input type="text"/>	查询
<input type="checkbox"/>	品牌ID	品牌名称	品牌首字母	操作
<input type="checkbox"/>	1	联想	L	修改
<input type="checkbox"/>	2	华为	H	修改
<input type="checkbox"/>	3	三星	S	修改
<input type="checkbox"/>	4	小米	X	修改
<input type="checkbox"/>	5	OPPO	O	修改
<input type="checkbox"/>	6	360	S	修改
<input type="checkbox"/>	7	中兴	Z	修改
<input type="checkbox"/>	8	魅族	M	修改
<input type="checkbox"/>	9	苹果	P	修改
<input type="checkbox"/>	10	VIVO	V	修改

[«](#) [1](#) [2](#) [»](#) 第 1 页 每页 10 条 /共11条

3.2 后端代码

3.2.1 分页结果封装实体

在 pinyougou-pojo 工程中创建 entity 包，用于存放通用实体类，创建类 PageResult

```
package entity;

import java.util.List;

/**
 * 分页结果封装对象
 * @author Administrator
 *
 */

public class PageResult implements Serializable{

    private long total;//总记录数

    private List rows;//当前页结果

    public PageResult(long total, List rows) {

        super();
    }
}
```



```
        this.total = total;

        this.rows = rows;

    }

    //getter and setter .....

}
```

3.2.2 服务接口层

在 pinyougou-sellergoods-interface 的 BrandService.java 增加方法定义

```
/**
 * 返回分页列表
 *
 * @return
 *
 */
public PageResult findPage(int pageNum, int pageSize);
```

3.2.3 服务实现层

在 pinyougou-sellergoods-service 的 BrandServiceImpl.java 中实现该方法

```
@Override

public PageResult findPage(int pageNum, int pageSize) {

    PageHelper.startPage(pageNum, pageSize);

    Page<TbBrand> page= (Page<TbBrand>) brandMapper.selectByExample(null);

    return new PageResult(page.getTotal(), page.getResult());

}
```

PageHelper 为 MyBatis 分页插件



3.2.4 控制层

在 pinyougou-manager-web 工程的 BrandController.java 新增方法

```
/**
 * 返回全部列表
 * @return
 */
@RequestMapping("/findPage")
public PageResult findPage(int page,int rows){
    return brandService.findPage(page, rows);
}
```

3.3 前端代码

3.3.1 HTML

在 brand.html 引入分页组件

```
<!-- 分页组件开始 -->
<script src="../../plugins/angularjs/pagination.js"></script>
<link rel="stylesheet" href="../../plugins/angularjs/pagination.css">
<!-- 分页组件结束 -->
```

构建 app 模块时引入 pagination 模块

```
var app=angular.module('pinyougou',['pagination']);//定义品优购模块
```



页面的表格下放置分页组件

```
<!-- 分页 -->

<tm-pagination conf="paginationConf"></tm-pagination>
```

3.3.2 JS 代码

在 brandController 中添加如下代码

```
//重新加载列表 数据

$scope.reloadList=function(){

    //切换页码

    $scope.findPage( $scope.paginationConf.currentPage,
        $scope.paginationConf.itemsPerPage);

}

//分页控件配置

$scope.paginationConf = {

    currentPage: 1,

    totalItems: 10,

    itemsPerPage: 10,

    perPageOptions: [10, 20, 30, 40, 50],

    onChange: function(){

        $scope.reloadList();//重新加载

    }

};

//分页

$scope.findPage=function(page,rows){
```



```
$http.get('../brand/findPage.do?page='+page+'&rows='+rows).success(  
  
    function(response){  
  
        $scope.list=response.rows;  
  
        $scope.paginationConf.totalItems=response.total;//更新总记录数  
  
    }  
  
);  
  
}
```

在页面的 body 元素上去掉 ng-init 指令的调用,添加了分控件后,分页请求将会自动发送

paginationConf 变量各属性的意义:

currentPage: 当前页码

totalItems:总条数

itemsPerPage:每页显示条数

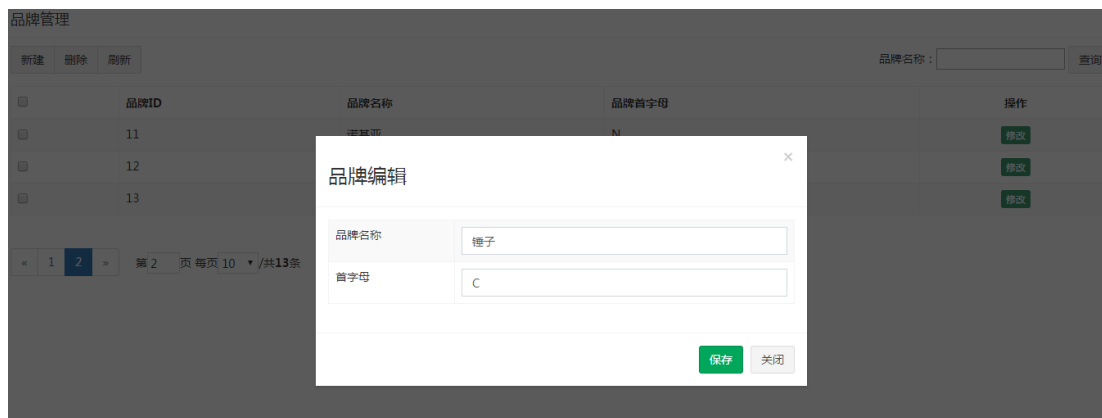
perPageOptions: 页码选项

onChange: 更改页面时触发事件

4.增加品牌

4.1 需求分析

实现品牌增加功能



4.2 后端代码

4.2.1 服务接口层

在 pinyougou-sellergoods-interface 的 BrandService.java 新增方法定义

```
/**
 * 增加
 */
public void add(TbBrand brand);
```

4.2.2 服务实现层

在 com.pinyougou.sellergoods.service.impl 的 BrandServiceImpl.java 实现该方法

```
@Override
public void add(TbBrand brand) {
    brandMapper.insert(brand);
}
```



4.2.3 执行结果封装实体

在 pinyougou-pojo 的 entity 包下创建类 Result.java

```
package entity;

import java.io.Serializable;

/**
 * 返回结果封装
 *
 * @author Administrator
 *
 */
public class Result implements Serializable{

    private boolean success;

    private String message;

    public Result(boolean success, String message) {

        super();

        this.success = success;

        this.message = message;

    }

    //getter and setter.....

}
```

4.2.4 控制层

在 pinyougou-manager-web 的 BrandController.java 中新增方法

```
/**
```



```
* 增加

* @param brand

* @return

*/

@RequestMapping("/add")

public Result add(@RequestBody TbBrand brand){

    try {

        brandService.add(brand);

        return new Result(true, "增加成功");

    } catch (Exception e) {

        e.printStackTrace();

        return new Result(false, "增加失败");

    }

}
```

4.3 前端代码

4.3.1 JS 代码

```
//保存

$scope.save=function(){

    $http.post('../brand/add.do',$scope.entity ).success(

        function(response){

            if(response.success){
```



```
        //重新查询

        $scope.reloadList();//重新加载

    }else{

        alert(response.message);

    }

}

);

}
```

4.3.2 HTML

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-content">

    <div class="modal-header">

        <button type="button" class="close" data-dismiss="modal"
        aria-hidden="true">×</button>

        <h3 id="myModalLabel">品牌编辑</h3>

    </div>

    <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>

        </table>

    </div>

</div>
```



```

        <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="save()">保存</button>

    <button class="btn btn-default" data-dismiss="modal" aria-hidden="true">
关闭</button>

</div>

</div>

</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

```

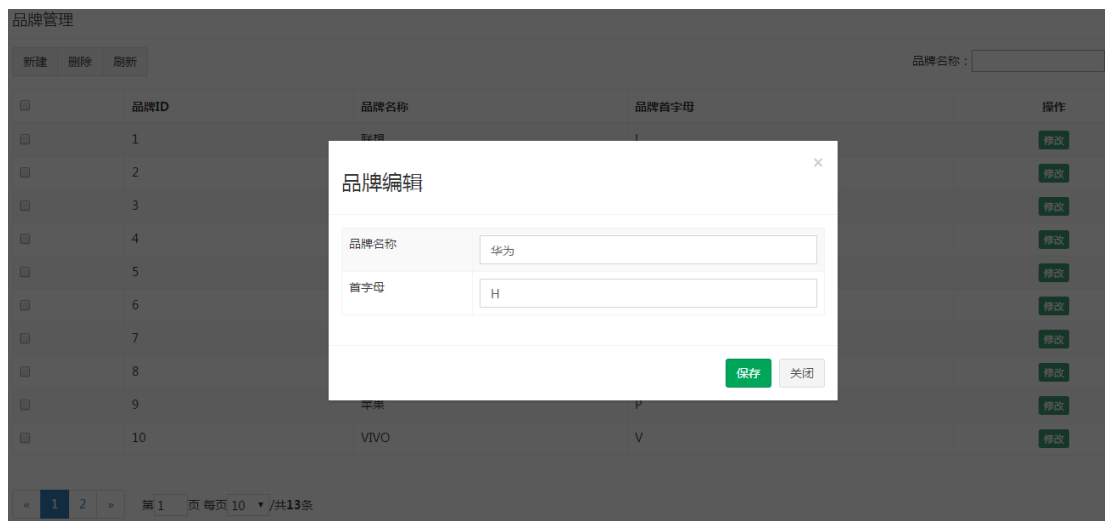
<button type="button" class="btn btn-default" title="新建" data-toggle="modal"
data-target="#editModal" ng-click="entity={}"><i class="fa fa-file-o"></i> 新建
</button>

```

5.修改品牌

5.1 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



5.2 后端代码

5.2.1 服务接口层

在 pinyougou-sellergoods-interface 的 BrandService.java 新增方法定义

```
/**
 * 修改
 */
public void update(TbBrand brand);

/**
 * 根据 ID 获取实体
 * @param id
 * @return
 */
public TbBrand findOne(Long id);
```



5.2.2 服务实现层

在 pinyougou-sellergoods-service 的 BrandServiceImpl.java 新增方法实现

```
/**
 * 修改
 */
@Override
public void update(TbBrand brand){
    brandMapper.updateByPrimaryKey(brand);
}

/**
 * 根据 ID 获取实体
 * @param id
 * @return
 */
@Override
public TbBrand findOne(Long id){
    return brandMapper.selectByPrimaryKey(id);
}
```

5.2.3 控制层

在 pinyougou-manager-web 的 BrandController.java 新增方法

```
/**
```



```
* 修改

* @param brand

* @return

*/

@RequestMapping("/update")

public Result update(@RequestBody TbBrand brand){

    try {

        brandService.update(brand);

        return new Result(true, "修改成功");

    } catch (Exception e) {

        e.printStackTrace();

        return new Result(false, "修改失败");

    }

}

/**

* 获取实体

* @param id

* @return

*/

@RequestMapping("/findOne")

public TbBrand findOne(Long id){

    return brandService.findOne(id);

}
```




5.3 前端代码

5.3.1 实现数据查询

增加 JS 代码

```
//查询实体

$scope.findOne=function(id){

    $http.get('../brand/findOne.do?id='+id).success(

        function(response){

            $scope.entity= response;

        }

    );

}
```

修改列表中的“修改”按钮，调用此方法执行查询实体的操作

```
<button type="button" class="btn bg-olive btn-xs" data-toggle="modal" data-target="#editModal" ng-click="findOne(entity.id)" >修改</button>
```

5.3.2 保存数据

修改 JS 的 save 方法

```
//保存

$scope.save=function(){

    var methodName='add';//方法名称

    if($scope.entity.id!=null){//如果有 ID

        methodName='update';//则执行修改方法

    }
```



```
    }

    $http.post('../brand/' + methodName + '.do', $scope.entity).success(

        function(response){

            if(response.success){

                //重新查询

                $scope.reloadList();//重新加载

            }else{

                alert(response.message);

            }

        }

    );

}
```

6.删除品牌

6.1 需求分析

点击列表前的复选框，点击删除按钮，删除选中的品牌。



品牌管理

新建

删除

刷新

品牌名称：

查询

<div></div>	品牌ID	品牌名称	品牌首字母	操作
<div></div>	1	联想	L	<div>修改</div>
<div></div>	2	华为	H	<div>修改</div>
<div></div>	3	三星	S	<div>修改</div>
<div><div></div></div>	4	小米	X	<div>修改</div>
<div><div></div></div>	5	OPPO	O	<div>修改</div>
<div></div>	6	360	S	<div>修改</div>
<div></div>	7	中兴	Z	<div>修改</div>
<div></div>	8	魅族	M	<div>修改</div>
<div></div>	9	苹果	P	<div>修改</div>
<div></div>	10	VIVO	V	<div>修改</div>

<

1

2

>

第 1 页 每页 10 条 /共11条

6.2 后端代码

6.2.1 服务接口层

在 pinyougou-sellergoods-interface 的 BrandService.java 接口定义方法

```
/**
 * 批量删除
 * @param ids
 */
public void delete(Long [] ids);
```

6.2.2 服务实现层

在 pinyougou-sellergoods-service 的 BrandServiceImpl.java 实现该方法

```
/**
 * 批量删除
 */
@Override
```



```
public void delete(Long[] ids) {  
  
    for(Long id:ids){  
  
        brandMapper.deleteByPrimaryKey(id);  
  
    }  
  
}
```

6.2.3 控制层

在 pinyougou-manager-web 的 BrandController.java 中增加方法

```
/**  
  
 * 批量删除  
  
 * @param ids  
  
 * @return  
  
 */  
  
@RequestMapping("/delete")  
  
public Result delete(Long [] ids){  
  
    try {  
  
        brandService.delete(ids);  
  
        return new Result(true, "删除成功");  
  
    } catch (Exception e) {  
  
        e.printStackTrace();  
  
        return new Result(false, "删除失败");  
  
    }  
  
}
```



6.3 前端代码

6.3.1 JS

主要思路：我们需要定义一个用于存储选中 ID 的数组，当我们点击复选框后判断是选择还是取消选择，如果是选择就加到数组中，如果是取消选择就从数组中移除。在点击删除按钮时需要用到这个存储了 ID 的数组。

这里我们补充一下 JS 的关于数组操作的知识

(1) 数组的 `push` 方法：向数组中添加元素

(2) 数组的 `splice` 方法：从数组的指定位置移除指定个数的元素，参数 1 为位置，参数 2 为移除的个数

(3) 复选框的 `checked` 属性：用于判断是否被选中

```
$scope.selectIds=[];//选中的 ID 集合

//更新复选

$scope.updateSelection = function($event, id) {

    if($event.target.checked){//如果是被选中,则增加到数组

        $scope.selectIds.push( id);

    }else{

        var idx = $scope.selectIds.indexOf(id);

        $scope.selectIds.splice(idx, 1);//删除

    }

}

//批量删除 ,注意此处不能使用 delete 关键字定义方法名

$scope.dele=function(){

    //获取选中的复选框

    $http.get('../brand/delete.do?ids='+$scope.selectIds).success(
```



```
function(response){  
  
    if(response.success){  
  
        $scope.reloadList();//刷新列表  
  
    }  
  
}  
  
);  
}
```

6.3.2 HTML

(1) 修改列表的复选框

```
<input type="checkbox" ng-click="updateSelection($event,entity.id)">
```

(2) 修改删除按钮

```
<button type="button" class="btn btn-default" title="删除" ng-click="dele()"><i  
class="fa fa-trash-o"></i> 删除</button>
```

7.品牌条件查询

7.1 需求分析

实现品牌条件查询功能，输入品牌名称、首字母后查询，并分页。

7.2 后端代码

7.2.1 服务接口层

在 pinyougou-sellergoods-interface 工程的 BrandService.java 方法增加方法定义

```
/**
```



```
* 分页

* @param pageNum 当前页 码

* @param pageSize 每页记录数

* @return

*/

public PageResult findPage(TbBrand brand, int pageNum,int pageSize);
```

7.2.2 服务实现层

在 pinyougou-sellergoods-service 工程 BrandServiceImpl.java 实现该方法

```
@Override

public PageResult findPage(TbBrand brand, int pageNum, int pageSize) {

    PageHelper.startPage(pageNum, pageSize);

    TbBrandExample example=new TbBrandExample();

    Criteria criteria = example.createCriteria();

    if(brand!=null){

        if(brand.getName()!=null && brand.getName().length()>0){

            criteria.andNameLike("%"+brand.getName()+"%");

        }

        if(brand.getFirstChar()!=null && brand.getFirstChar().length()>0){

            criteria.andFirstCharEqualTo(brand.getFirstChar());

        }

    }

    Page<TbBrand> page= (Page<TbBrand>)brandMapper.selectByExample(example);
```



```
        return new PageResult(page.getTotal(), page.getResult());  
    }  
}
```

7.2.3 控制层

在 pinyougou-manager-web 的 BrandController.java 增加方法

```
/**  
 * 查询+分页  
 * @param brand  
 * @param page  
 * @param rows  
 * @return  
 */  
  
@RequestMapping("/search")  
  
public PageResult search(@RequestBody TbBrand brand, int page, int rows ){  
    return brandService.findPage(brand, page, rows);  
}  
}
```

7.3 前端代码

修改 pinyougou-manager-web 的

```
$scope.searchEntity={}; //定义搜索对象  
  
//条件查询  
  
$scope.search=function(page,rows){  
    $http.post('../brand/search.do?page='+page+"&rows="+rows,  
    $scope.searchEntity).success(  
        function(data){  
            $scope.searchEntity=data;  
        }  
    )  
}
```




```
function(response){  
  
    $scope.paginationConf.totalItems=response.total;//总记录数  
  
    $scope.list=response.rows;//给列表变量赋值  
  
}  
  
);  
}
```

修改 reloadList 方法

```
//刷新列表  
  
$scope.reloadList=function(){  
  
    $scope.search( $scope.paginationConf.currentPage,  
    $scope.paginationConf.itemsPerPage);  
  
}
```