# 京淘后台搭建

## 课前回顾

### 动态代理

1. JDK动态代理

JDK动态代理时JDK原生提供的.

要求:要求被代理者必须实现接口.

Spring创建代理对象的原则:

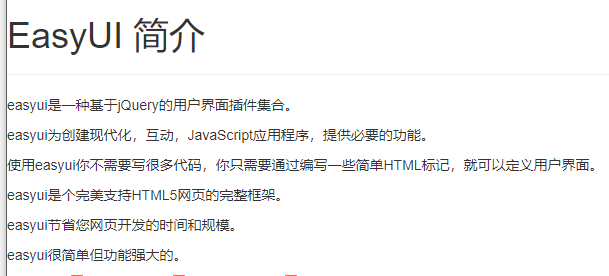
默认条件下使用JDK动态代理,如果目标类没有实现接口.则默认采用cglib.

1. CGLIB动态代理

CGLIB代理需要导入第三方jar包.CGLib可以为任意的类创建代理对象.同时代理对象是目标对象的子类.

## EasyUI介绍

### EasyUI官网



### EasyUI入门案例

1. 引入页面js和css样式表

<!--引入jquery的js，EasyUI的执行需要依赖于jQuery -->

<script type=*"text/javascript"*

src=*"/js/jquery-easyui-1.4.1/jquery.min.js"*></script>

<!--easyUI的js主文件 -->

<script type=*"text/javascript"*

src=*"/js/jquery-easyui-1.4.1/jquery.easyui.min.js"*></script>

<!--国际化的js文件 -->

<script type=*"text/javascript"*

src=*"/js/jquery-easyui-1.4.1/locale/easyui-lang-zh\_CN.js"*></script>

<!--引入easyUI的样式 -->

<link rel=*"stylesheet"* type=*"text/css"*

href=*"/js/jquery-easyui-1.4.1/themes/icon.css"* />

<link rel=*"stylesheet"* type=*"text/css"*

href=*"/js/jquery-easyui-1.4.1/themes/default/easyui.css"* />

1. 拖拽

<div class="**easyui-draggable**">拖动DIV</div>

1. 进度条

<div id="p" class="easyui-progressbar" style="width:400px;"></div>

1. 按钮

<a id="btn1" href="#" class="easyui-linkbutton" data-options="iconCls:'icon-back'">返回</a>

1. 其他案例自已学习

## PowerDesinger

### PD介绍

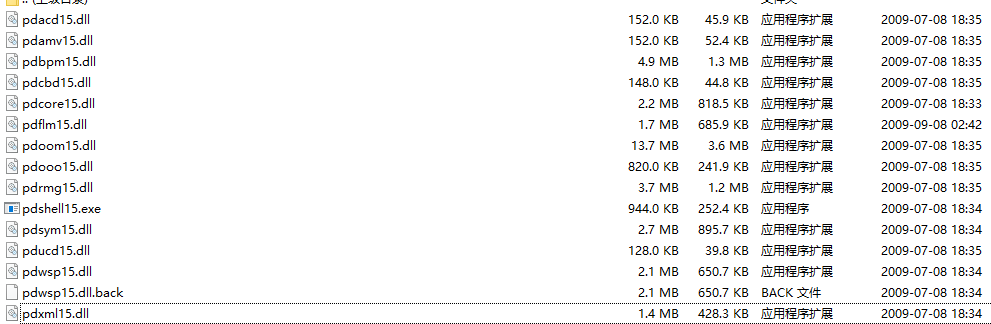
Power Designer 是[Sybase](https://baike.baidu.com/item/Sybase" \t "_blank)公司的CASE工具集，使用它可以方便地对[管理信息系统](https://baike.baidu.com/item/%E7%AE%A1%E7%90%86%E4%BF%A1%E6%81%AF%E7%B3%BB%E7%BB%9F)进行分析设计，他几乎包括了**数据库模型设计**的全过程。[利用](https://baike.baidu.com/item/%E5%88%A9%E7%94%A8)Power Designer可以制作数据[流程图](https://baike.baidu.com/item/%E6%B5%81%E7%A8%8B%E5%9B%BE" \t "_blank)、概念数据模型、物理数据模型，还可以为数据仓库制作结构模型，也能对团队设计模型进行控制。他可以与许多流行的[软件开发工具](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E5%B7%A5%E5%85%B7)，例如PowerBuilder、[Delphi](https://baike.baidu.com/item/Delphi" \t "_blank)、VB等相配合使开发时间缩短和使[系统设计](https://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F%E8%AE%BE%E8%AE%A1)更优化。

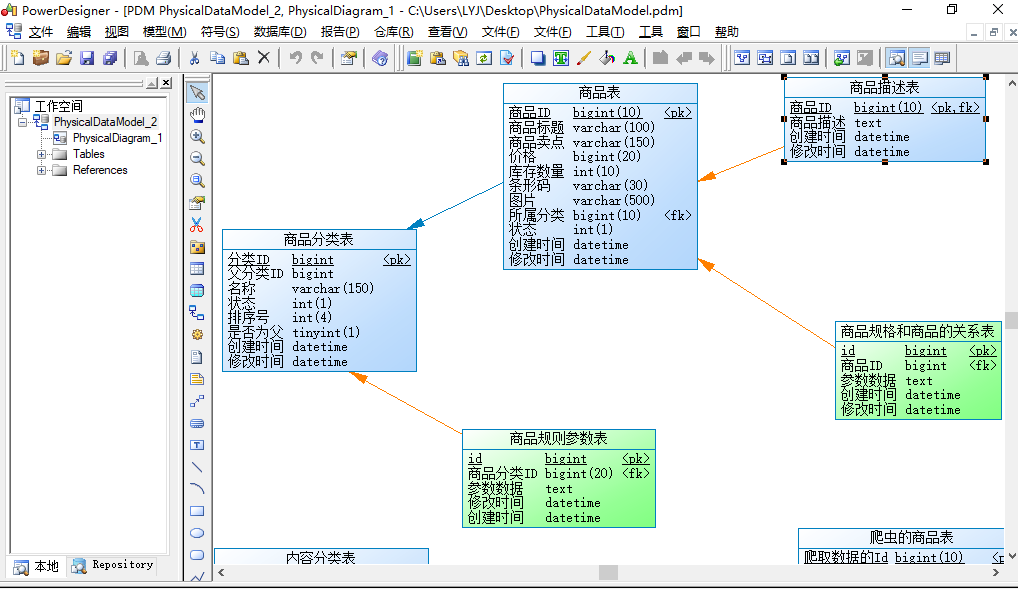
总结:PD是现在主流的数据建模工具.可以通过可视化的界面设计表结构.

可以根据不同的数据库类型自动的生成sql语句.同时提供了物理模型等一系列的建模工具模板方便用户使用.

### 汉化破解PD

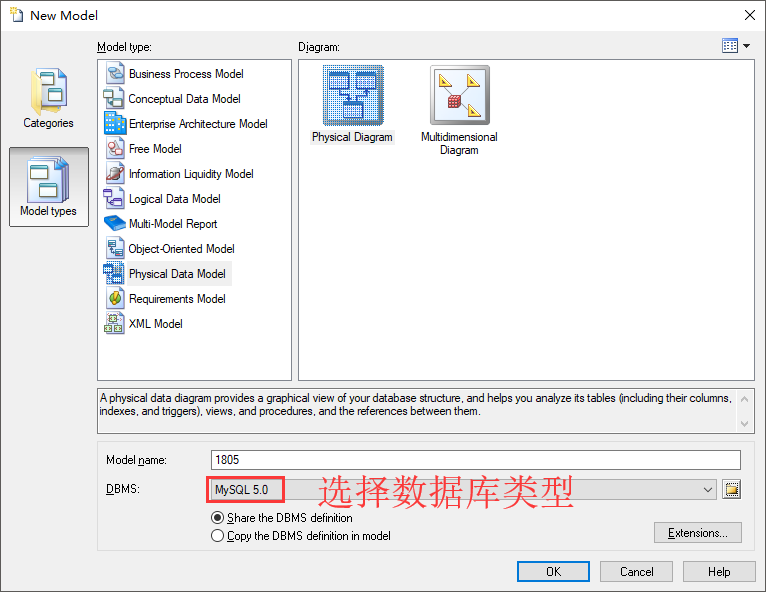
将汉化破解文件拷贝到PD安装的根目录后,之后重启PD即可





### PD使用

1.选择物理模型



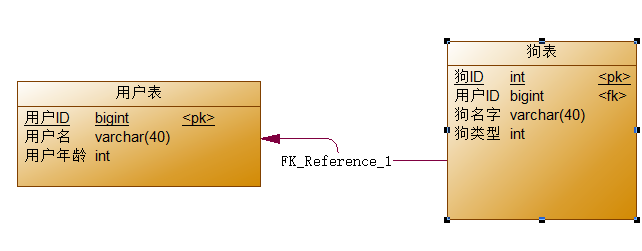
2.创建数据表



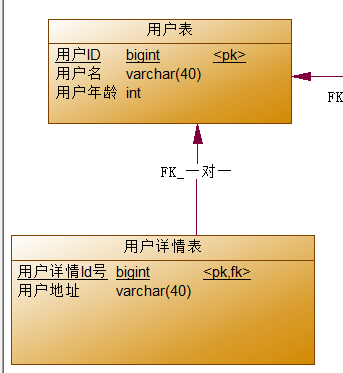
1. 定义关联关系
2. 一对一 夫妻 看证
3. 一对多 漂亮妹子的追求者
4. 多对多 动物世界/老师和学生 双向的一对多

策略:一般在模型中只会出现一对一和一对多,换线时一般由从表向主表划线.

说明:下图中是一对多



说明:下图中是一对一.如果一个字段既是主键也是外键则和主表的数据相同



总结:一般创建数据表时不会创建关联关系.目的提示表的执行的效率.全部的表关系都有程序员自己来维护.

### 基于RestFul实现页面跳转

需求:

url: /page/item-add

url: /page/item-list

如果用户有大量的用户请求,则后台Controller中需要编译对应的方法.这样做很繁琐.

优化:使用restFul风格实现数据动态传参

/\*\*

\* 通过restFul实现页面的通用跳转

\* url:

\* /page/item-add

\* /page/item-list

\* 思考:

\* 如果能够获取动态变化的数据,则可以实现页面的动态跳转

\* 实现:实现数据的动态获取

\* restFul格式要求:

\* 1.参数拼接在url中,并且以"/"进行分割

\* 2.参数的位置必须固定

\* 3.后台服务端接收参数使用{}包裹变量.之后使用

\* **@PathVariable实现数据的动态取值**

\* **@RequestMapper**(/addUser/{id}/{name}/{age})

\* localhost:8091/addUser/1/tom/18

\*

\* public String addUser(@PathVariable int id,String name,int age)

\* **@return**

\*/

@RequestMapping("/page/{module}")

**public** String module(@PathVariable String module){

**return** module;

}

## 商品列表展现

### 页面分析

<table class="easyui-datagrid" id="itemList" title="商品列表"

data-options="singleSelect:false,collapsible:true,pagination:true,**url:'/item/query'**,method:'get',pageSize:20,toolbar:toolbar">

<thead>

<tr>

<th data-options="field:'ck',checkbox:true"></th>

<th data-options="field:'id',width:60">商品ID</th>

<th data-options="field:'title',width:200">商品标题</th>

<th data-options="field:'cid',width:100,align:'center',formatter:KindEditorUtil.findItemName">叶子类目</th>

<th data-options="field:'sellPoint',width:100">卖点</th>

<th data-options="field:'price',width:70,align:'right',formatter:KindEditorUtil.formatPrice">价格</th>

<th data-options="field:'num',width:70,align:'right'">库存数量</th>

<th data-options="field:'barcode',width:100">条形码</th>

<th data-options="field:'status',width:60,align:'center',formatter:KindEditorUtil.formatItemStatus">状态</th>

<th data-options="field:'created',width:130,align:'center',formatter:KindEditorUtil.formatDateTime">创建日期</th>

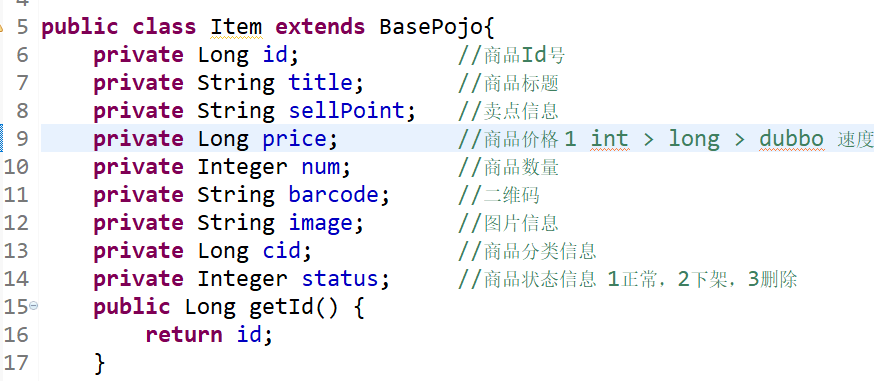
<th data-options="field:'updated',width:130,align:'center',formatter:KindEditorUtil.formatDateTime">更新日期</th>

</tr>

</thead>

</table>

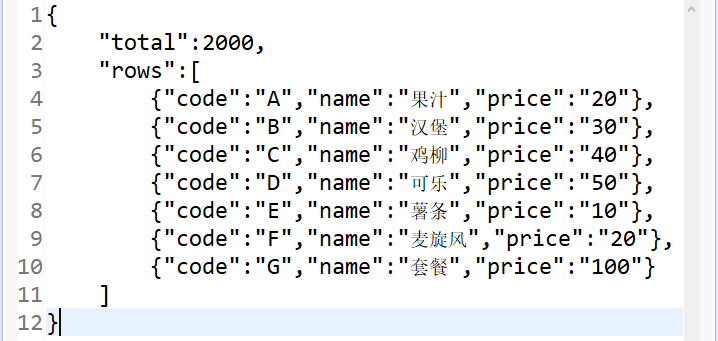
### 编辑POJO对象



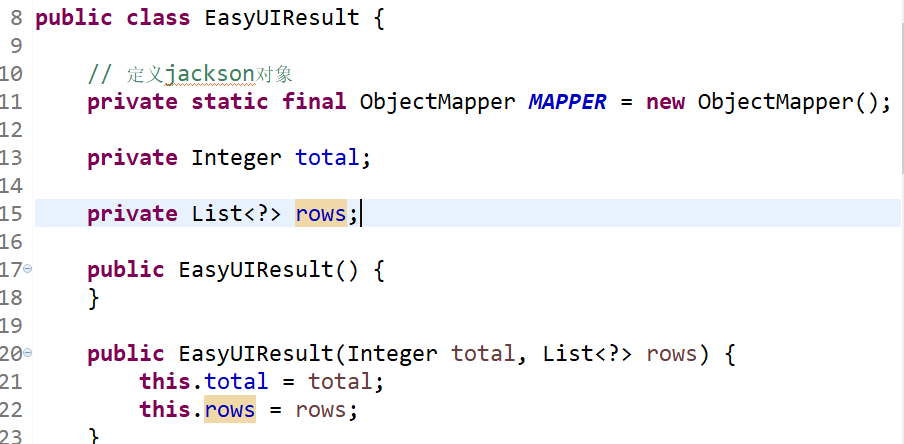
### 返回值类型分析

说明:根据url实现请求的拦截,之后将特定格式的数据进行返回.

经过页面分析:需要准备特定格式的vo对象total记录总数 rows行数等属性.



封装VO对象



### 编辑Controller

@Controller

@RequestMapping("/item")

**public** **class** ItemController {

@Autowired

**private** ItemService itemService;

//实现商品分页查询 http://localhost:8091/item/query?page=1&rows=50

@RequestMapping("/query")

@ResponseBody

**public** EasyUIResult findItemByPage(**int** page,**int** rows){

**return** itemService.findItemByPage(page,rows);

}

}

### 编辑Service

@Override

**public** EasyUIResult findItemByPage(**int** page, **int** rows) {

//查询商品的记录总数

**int** total = itemMapper.findCount();

/\*\*

\* limit 起始位置,查询条数

\* 第一页 select \* from tb\_user limit 0,20

\* 第二页 select \* from tb\_user limit 20,20

\* 第三页 select \* from tb\_user limit 40,20

\* 第N页 select \* from tb\_user limit (page-1)\*rows,rows

\*/

**int** start = (page - 1) \* rows;

List<Item> itemList = itemMapper.findItemByPage(start,rows);

**return** **new** EasyUIResult(total, itemList);

}

### 编辑Mapper

/\*\*

\* 说明:Mybatis中不允许多值传参.可以将多值封装为单值

\* 封装策略:

\* 1.将数据封装为POJO对象

\* 2.将数据封装为Map集合 map<key,value>

\* **@Param**("start")int start 底层实现时封装为map集合

\* 3.将数据封装为Array数组/List集合

\* **@param** start

\* **@param** rows

\* **@return**

\*/

List<Item> findItemByPage(@Param("start")**int** start,@Param("rows")**int** rows);

2.编辑Mapper映射文件

<!--根据分页实现商品信息查询

ctrl + shift + y 小写

ctrl + shift + x 大写

因为在Linux中严格区分大小写,所以要严格表名

#号和$符区别

#号有预编译的效果,会为参数添加一对""号,能用#号绝不用$

$只有以字段名称为参数时使用$

SELECT \* FROM tb\_item WHERE "cid" > 200 查询没有结果

-->

<select id=*"findItemByPage"* resultType=*"Item"*>

select \* from tb\_item order by created desc limit ${start},#{rows}

</select>

### EasyUI中数据格式转化

例子:格式化价格

formatPrice : **function**(val,row){

**return** (val/100).toFixed(2);

},

例子2:格式化时间

formatDateTime : **function**(val,row){

**var** now = **new** Date(val);

**return** now.format("yyyy-MM-dd hh:mm:ss");

},

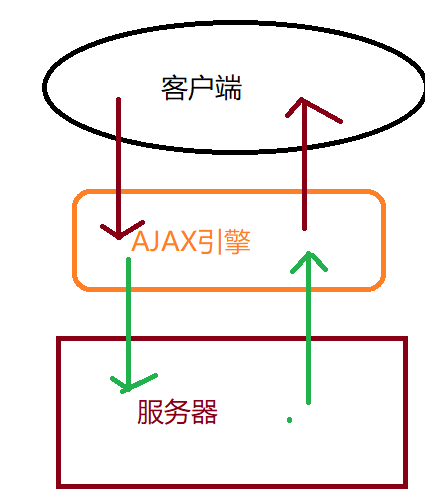
## 实现商品分类目录回显

说明:根据之前的案例实现商品分类目录的回显功能.

思路:根据value值发起ajax请求.将560转化为商品分类名称.

### AJAX数据异步原因

说明:因为ajax中使用ajax引擎.帮助用户发起请求,实现请求的异步.



### 页面AJAX请求

//格式化名称

findItemName : **function**(val,row){

**var** name;

$.ajax({

type:"post",

url:"/item/cat/queryItemName",

data:{itemId:val},

cache:**true**, //缓存

async:**false**, //表示同步 默认的是异步的true

dataType:"text",

success:**function**(data){

name = data;

}

});

**return** name;

},

### 编辑Controller

//根据商品分类id查询商品分类名称

@RequestMapping(value="/cat/queryItemName",produces="text/html;charset=utf-8")

@ResponseBody

**public** String findItemCatNameById(Long itemId){

**return** itemService.findItemCatNameById(itemId);

}

//问题1:使用@ResponseBody时处理实体对象时一般不乱码.如果处理String类型可能就乱码

/\*\*

\* 原因:因为使用@ResponseBody内部有多重的解析的机制.

\* 当解析为string类型时默认以ISO-8859-1的格式进行解析.

\*

\* public static final Charset DEFAULT\_CHARSET = Charset.forName("ISO-8859-1");

\* 当解析对象时,默认以UTF-8格式进行解析.

\* public static final Charset DEFAULT\_CHARSET = Charset.forName("UTF-8");

\*/

### 编辑Service

@Override

**public** String findItemCatNameById(Long itemId) {

**return** itemMapper.findItemCatNameById(itemId);

}

### 编辑Mapper接口

@Select("select name from tb\_item\_cat where id = #{itemId}")

String findItemCatNameById(Long itemId);

### 页面效果

