课程目标

1. 实现开发者用户的登入和注销
2. 实现app信息显示页的，搜索条件展示
3. 实现开发者用

分析:

通过用户名和密码去查询数据库，存在用户则登入成功，否则登入失败.

SELECT \* FROM dev\_user WHERE devCode='test001' AND devpassword='123456'

1. 编写dao层

略：selectByExample

1. 编写业务层

接口:

|  |
| --- |
| **public interface** DevUserService {  */\*\*  \* 实现登入功能  \** ***@param username*** *用户名  \** ***@param password*** *密码  \** ***@return*** *当前用户对象  \*/* **public** DevUser login(String username,String password); } |

实现:

|  |
| --- |
| @Service **public class** DevUserServiceImpl **implements** DevUserService {  @Autowired  **private** DevUserMapper **devUserMapper**;  @Override  **public** DevUser login(String username, String password) {  DevUserExample devUserExample=**new** DevUserExample();  DevUserExample.Criteria criteria =devUserExample.createCriteria();  *//添加查询条件* criteria.andDevcodeEqualTo(username);  criteria.andDevpasswordEqualTo(password);  List<DevUser> list=**devUserMapper**.selectByExample(devUserExample);  **if**(list!=**null**&&list.size()==1){  **return** list.get(0);  }**else**{  **return null**;  }  } } |

1. 编写控制器

|  |
| --- |
| @RequestMapping(**"/login"**) **public** String login( String username, String password, HttpSession session){  *//调用业务* DevUser devUser=**devUserService**.login(username,password);  **if**(devUser==**null**){  *//登入失败* **return "redirect:jsp/devlogin.jsp"**;  }**else**{  *//登入成功,使用session保存用户的信息* session.setAttribute("devUser",devUser);  session.setMaxInactiveInterval(600); *//10分钟有效* **return "developer/main"**;  } } |

1. 登入的表单:

略

1. 在主页面中显示登入人的姓名

欢迎你：**${**devUser.devname **}**

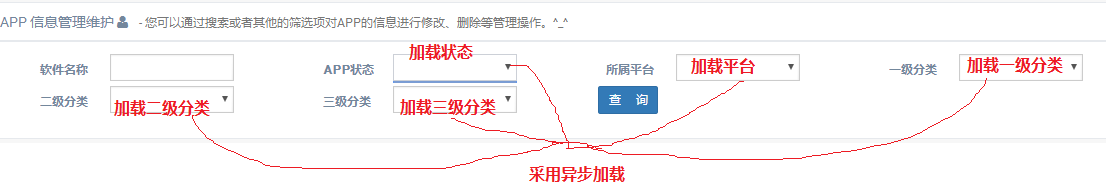
1. **实现退出功能**

**注意:登入的退出功能就是将登入时保存的用户信息删除即可(清除session)**

|  |
| --- |
| *//登入退出* @RequestMapping(**"/logout"**) **public** String logout(HttpSession session){  *//清除session* session.removeAttribute(**"loginInfo"**);  **return "redirect:jsp/devlogin.jsp"**; } |

**能过退出超链接跳转即可**

1. 实现app页面的搜索条件显示



1. 使用异步加APP状态信息

分析:

SELECT valueid,valuename FROM `data\_dictionary` WHERE typename='APP状态'

* 1. 编写dao层查询所有APP状态信息

接口:通过typename类型名称获取相关的类型信息

|  |
| --- |
| *//通过类型名称获取信息* List<DataDictionary> getDataDictionaryInfo(String typename); |

方法:

|  |
| --- |
| *<!--获取类型信息-->* <**select id="getDataDictionaryInfo" parameterType="string" resultType="DataDictionary"**>  SELECT valueId,valueName FROM data\_dictionary WHERE typeName=#{typename} </**select**> |

* 1. 编写业务返回信息

接口:

|  |
| --- |
| **public interface** DictionaryService {  */\*\*  \* 通过类型名称加载相关字典  \** ***@param typename*** *类型名称  \** ***@return*** *返回某类型的相关字典信息  \*/* **public** List<DataDictionary> getDictionary(String typename); } } |

实现:

|  |
| --- |
| @Service **public class** DictionaryServiceImpl **implements** DictionaryService {   @Autowired  **private** DataDictionaryMapper **dataDictionaryMapper**;   @Override  **public** List<DataDictionary> getDictionary(String typename) {  **return dataDictionaryMapper**.getDataDictionaryInfo(typename);  } } |

* 1. 编写控制器返回app状态信息的数据

|  |
| --- |
| @Controller **public class** DictionaryController {  @Autowired(required = **false**)  **public** DictionaryService **dictionaryService**;  */\* 方式一:  /getState 返回状态数据  /getAp 返回平台数据  方式二:  /getState?类型 返回类型的数所  \*/  //返回异步数据* @RequestMapping(**"/getState"**)  @ResponseBody  **public** List<DataDictionary> getState(){  **return dictionaryService**.getDictionary(**"APP状态"**);  } } |

* 1. 编写前端页面的代码:appinfolist.jsp

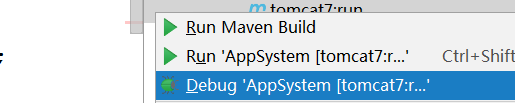
|  |
| --- |
| <**script language="javascript"**>  $(**function**(){ *//加载事件  //1.发送异步请求获取app状态进行展示* $.post(**"${**pageContext.request.contextPath **}/getAppStateData"**,**null**,**function**(data){  *//console.log(data);* **for**(**var** i=0;i<data.**length**;i++){  *//创建一个option* **var** node=$(**"<option value='"**+data[i].valueid+**"'>"**+data[i].valuename+**"</option>"**);  *//将节点添加到下拉列表* $(**"#queryStatus"**).append(node);  }  },**"json"**);    }); </**script**> |

**扩展:使用调试的步骤:**

1. **设置断点:**

**代码运行到断点处就会停止..**

1. **以debug模式运行工程(支持调试)**



1. 进行调试

