EasyEE-ssh 架构手册

EasyEE JaveEE background is a basis for the development framework. Background of project development for the enterprise provides the infrastructure that provides common components and user-based, role, rights management system privileges program (support page displays access control), front end uses EasyUI framework.

EasyEE-SSH

Based on Struts2,Hibernate4, Spring4 of EasyEE infrastructure development background. Include component:

- `SSH`: JAVA framework and related technical background and rights management module
- `EasyUI`: Based on the front-end framework jQuery
- `EasyUIEx`: EasyUI of extensions（API）
- `EasyFilter`: Jave Web content filtering requests replacement component, illegal keyword filtering （API）

Framework Manual

1. Project structure
2. Rapid Development Manual
3. Other configuration instructions

Project structure

```
EasySSH （项目）
   + src/main/java (源码目录)
       + cn.easyproject.easyssh (包前缀)
           + base (项目基础功能公共组件包)
               + action
                   + BaseAction.java（基础Action，所有Action类继承BaseAction）
               + dao
                   + CommonDAO.java (通用DAO接口)
                   + impl
                       + CommonDAOHibernateImpl.java (通用DAO实现类)
               + interceptor
                   + UserInterctptor.java (用户访问权限核心拦截器)
               + service
                   + BaseService.java (基础Service，所有Service类都基础BaseService)
               + tag
                   + ShowActionTag.java (权限管理，页面显示权限自定义标签类)
               + util
                   + ... (PageBean、加密、日期处理等日常开发和项目所需的工具类)
           + sys (权限管理系统实现包)
               + action, entity, service, util, criteria...
           + moudle （自定义模块开发包，演示了一个员工管理的Demo)
               + action, entity, service, criteria...
   + src/main/resource (源码资源目录)
       + applicationContext_bean_scan.xml (Spring Bean注解扫描)
       + applicationContext.xml (Spring 核心配置文件——数据源，事务管理)
       + ApplicationResources.properties (国际化资源文件，保存信息)
       + db.properties (数据库连接参数配置)
       + easyFilter.properties (EasyFilter Web请求内容过滤替换组件配置文件)
       + ehcache.xml (ehcache 二级缓存配置)
       + log4j.properties (Log4J日志配置文件)
       + struts.xml (Struts2 核心配置文件，负责引入各个模块的配置文件)
       + struts_easyssh_default.xml (Struts2 默认父包配置)
       + struts_easyssh_dispatcher.xml (Struts2 跳转配置，访问WEB-INF下的JSP视图)
       + struts_easyssh_sys.xml (权限系统配置文件)
       + struts.properties (Struts2 properties配置)
```

```
+ src/test/java (测试源码目录)
+ src/test/resource (测试资源目录)
+ WebRoot (Web根目录)
    + easyssh (easyssh核心js文件)
        + json (easyssh所需的JSON文件——主题，图标)
        + easyssh.main.js (easyssh核心JS，全局Ajax请求响应处理)
        + jquery.cookie.js (cookie处理)
    + easyui (EasyUI 前端框架)
    + easyuiex (EasyUIEx 扩展插件)
    + echarts (EChars 前端图形报表组件)
    + error (JSP错误页面)
    + images (图片)
    + jsp (一般JSP页面)
        + echarts (ECharts demo)
        + error (系统错误页面)
            + notFound.jsp (404错误提示页面)
            + permissionDenied.jsp (权限不足提示页面)
            + serverError.jsp (400错误提示页面)
        + VerifyCode.jsp (验证码生成JSP)
    + script (项目开发页面相关JS文件，和WEB-INF/content下的页面一一对应)
        + main
            + sys (系统权限模块JS...)
            + main.js (系统主页面JS)
        + login.js (登录页面JS)
    + style (css样式表)
        + easyssh.main.css (easyssh 系统全局css)
    + WEB-INF
        + content (项目核心页面)
            + dialog (EasyUI Dialog 相关页面)
                + sys (权限系统模块相关Dialog页面)
                + module (自定义开发模块，员工管理Demo相关Dialog页面)
            + main (EasyUI 核心页面)
                + sys (权限系统模块相关页面)
                + module (自定义开发模块，员工管理Demo相关页面)
                + main.jsp (EasySSH登录后的主页面)
            + login.jsp (登录页面)
        + lib (系统jar包，包含SSH、druid连接池、easyFilter、开发常用组件等等)
        + easyssh-tags.tld (权限管理系统面显示权限控制自定义标签库)
        + web.xml (部署描述符文件)
     + toLogin.action (跳转到登录页面的action文件)
```

Rapid Development Manual

1. **创建新模块包结构**
    - 在项目基础前缀包下创建新模块的包，如:

      cn.easyproject.easyssh.`module`
    - 在新模块包下创建功能包，如:

      cn.easyproject.easyssh.module.`entity`
      cn.easyproject.easyssh.module.`service`
      cn.easyproject.easyssh.module.`action`
      cn.easyproject.easyssh.module.`criteria`
2. **编写POJO实体类代码，并在Spring注册**
    - 在`entity`下根据表创建POJO实体类和ORM映射（注解或XML）
        - 注解映射

```
@Entity
@Table(name = "Emp", catalog = "EASYSSH")
public class Emp implements java.io.Serializable {
   //...
```

```
}
```

- XML 映射

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
<class name="cn.easyproject.easyssh.module.entity.Emp" table="module_emp" catalog="easyssh">
    <!-- ... -->
</class>
</hibernate-mapping>
```

1. **编写Service代码**

   - 在`service`下编写业务接口

```
public interface DeptService {
    public void add(Dept dept);

    public void delete(Integer deptno);

    public void update(Dept dept);

    public Dept get(Integer deptno);

    public void findByPage(PageBean pageBean, DeptCriteria deptCriteria);

    public int findMaxPage(int rowPerPage);

    public List<Dept> findAll();
}
```

   - 在`service.impl`下编写业务接口对应的实现类

     - 业务实现类统一继承BaseService类
     - BaseService中注入了通用DAO，直接调用commonDAO的数据方法方法即可
     - 使用@Service声明Bean

```
/**
 * 业务实现类统一继承BaseService类
 * BaseService中注入了通用DAO，直接调用commonDAO的数据方法方法即可
 *
 * @author easyproject.cn
 * @version 1.0
 *
 */
@Service("deptService")
public class DeptServiceImpl extends BaseService implements DeptService {

    @Override
    public void add(Dept dept) {
        commonDAO.save(dept);
    }

    @Override
    public void delete(Integer deptno) {
        commonDAO.updateByHql("delete from Dept where deptno=?",deptno);
    }

    @Override
    public void update(Dept dept) {
        commonDAO.update(dept);
    }

    @Override
```

```
    public Dept get(Integer deptno) {
        return commonDAO.get(Dept.class, deptno);
    }

    @Override
    public void findByPage(PageBean pageBean, DeptCriteria deptCriteria) {
        pageBean.setEntityName("Dept dept");
        pageBean.setSelect("select dept");

        // 按条件分页查询
        commonDAO.findByPage(pageBean,deptCriteria);
    }

    @Override
    public int findMaxPage(int rowPerPage) {
        return (commonDAO.findMaxPage("select count(*) from Dept", rowPerPage)-1)/rowPerPage+1;
    }

    @SuppressWarnings("unchecked")
    @Override
    public List<Dept> findAll() {
        return commonDAO.find("from Dept");
    }

}
```

- 在`applicationContext.xml` 中使用`aop:config`添加新模块业务层的声明式事务支持（或使用注解式配置）

```
  <!-- AOP 将通知织入切点表达式指定的方法-->
  <aop:config>
      <!--  事务AOP -->
      <aop:advisor advice-ref="txAdvie"
          pointcut="execution(* cn.easyproject.easyssh.sys.service..*.*(..))" />
      <!-- 自定义模块事务配置.... -->
      <aop:advisor advice-ref="txAdvie"
          pointcut="execution(* cn.easyproject.easyssh.module.service..*.*(..))" />
  </aop:config>
```

1. **编写Action控制器**

   - 编写控制器代码

```
 /**
  * 所有Action处理类统一继承BaseAction
  *
  * BaseAction中定义了一下内容：
  * - request, application Servlet API
  * - 请求响应相关的JSON参数（EasyUI框架请求都是通过JSON响应）
  * - 初始化JSON响应数据的方法（setJsonMap，setJsonMsgMap，setJsonPaginationMap(PageBean, Object...)）
  * - EasyUI分页信息相关的属性
  * - result="json" 的 JSON 常量
  *
  */
public class DeptAction extends BaseAction {

    // 必须和@Service("deptService")注解声明的业务Bean名称相同
    // 必须有setter方法
    private DeptService deptService;

    private Dept dept;

    /**
     * 分页
     * @return
     */
    public String list(){
```

```
        PageBean pb = super.getPageBean(); // 获得分页对
        deptService.findByPage(pb,dept);
        // EasyUI框架响应结果都是JSON
        // JSON数据初始化，包含EasySSH Ajax响应信息和分页信息
        super.setJsonPaginationMap(pb);
        return JSON;
    }

    /**
     * CRUD
     * @return
     */
    public String save(){
        // 保存用户
        try {
            deptService.add(dept);

            // 处理成功 消息
            msg = getText("msg.saveSuccess");

            // 如果需要刷新，跳转到最后一页
//            super.page = deptService.findMaxPage(rows);
        } catch (Exception e) {
            e.printStackTrace();
            msg = getText("msg.saveFail");
            statusCode=StatusCode.ERROR; //默认为OK
        }

        /*
         * Ajax响应信息
         * statusCode：响应状态码;
         * msg: 响应消息;
         * callback: 执行回调函数,
         * locationUrl: 跳转页面
         */
        // EasyUI框架响应结果都是JSON
        // JSON数据初始化，包含EasySSH Ajax响应信息
//        super.setJsonMsgMap();
        // 添加数据后，使用rowData信息更新行的内容
        super.setJsonMsgMap("rowData", dept);

        // 返回JSON
        return JSON;
    }

    public String delete(){
        try {
            deptService.delete(dept.getDeptno());
        } catch (Exception e) {
            e.printStackTrace();
            statusCode=StatusCode.ERROR;
        }
        super.setJsonMsgMap();
        return JSON;
    }

    public String update(){
        try {
            deptService.update(dept);
            msg=getText("msg.updateSuccess");
            super.refreshPermissions(); //刷新权限
        } catch (Exception e) {
            e.printStackTrace();
            msg=getText("msg.updateFail");
            statusCode=StatusCode.ERROR;
        }
        setJsonMsgMap();
```

```
        return JSON;
    }

    public Dept getDept() {
        return dept;
    }

    public void setDept(Dept dept) {
        this.dept = dept;
    }

    public void setDeptService(DeptService deptService) {
        this.deptService = deptService;
    }

}
```

- 配置控制器

1. 在`src/main/resources`下创建开发模块相应的struts2配置XML文件 `struts_easyssh_module.xml`， `package`继承`easyssh-default`父包：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
   <!-- 自定义模块 - 员工管理 -->
   <!-- 统一继承easyssh-default父包 -->
   <package name="easyssh-module" extends="easyssh-default" namespace="/">
            <!-- EasyUI框架响应结果都是JSON -->
            <action name="dept_*" class="cn.easyproject.easyssh.module.action.DeptAction" method="{1}">
            </action>
            <action name="emp_*" class="cn.easyproject.easyssh.module.action.EmpAction" method="{1}">
            </action>
   </package>
</struts>
```

2. 在`struts.xml`中include创建的模块配置文件`struts_easyssh_module.xml`:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
        <!-- easyssh 默认主系统配置模块包 -->
        <include file="struts_easyssh_default.xml"></include>
        <!-- 页面转发跳转配置包 -->
        <include file="struts_easyssh_dispatcher.xml"></include>
        <!-- EasySSH 权限系统配置包 -->
        <include file="struts_easyssh_sys.xml"></include>

        <!-- 自定义模块开发配置文件 - 员工管理 -->
        <include file="struts_easyssh_module.xml"></include>
</struts>
```

2. **页面开发**

- 在主目录下创建存放新模块页面的目录，如：
  - `WebRoot/WEB-INF/content/main/module`（EasyUI 页面主目录）
  - `WebRoot/WEB-INF/content/dialog/module`(EasyUI Dialog页面主目录)
- 按照EasyUI框架指导和Demo开发页面
  - EasyEE使用`EasyUI Tabs`的`href`属性引入tab子页面, 不是content(iframe框架)属性, 所以各个模块页面是和当前页面合并在一起, 无需在子页面重复引入主页面的JS, CSS等.

> EasyUI Tabs两种动态动态加载方式之间的区别：
>
> **-使用content(iframe框架)引入页面：**
>
> ```
> content : '<iframe scrolling="auto" frameborder="0" src="' + url + '" style="width:100%;height:100%;"></iframe>';
> ```
>
> 作为独立窗口存在, 页面内容独立, 与当前页面互不干扰
> 需要独立引入需要的JS和CSS资源
> 弹出的内容是在内部窗口内
> **-使用href方法：**
>
> ```
> href : url,
> ```
>
> 内容片段加载, 引入的内容和当前页面合并在一起
> 不需要引入页面已经引入的JS和CSS资源
> 引用的页面不能有body, 否则加载的内容内部的JS文件文法执行
> 会显示html渲染解析的提示
> 要防止页面间DOM元素, JS的命名冲突(DOM命名要使用统一前缀唯一, JS使用命名空间)

- 为了防止页面间DOM元素命名冲突, 应该为每个页面的DOM元素使用统一的唯一前缀
- 为了防止页面间JavaScript变量命名冲突, 应该为每个页面定义唯一的操作命名空间, 所有函数注册进命名空间
- 默认开发推荐结构如下：

```
<%-- 1. 页面Datagrid初始化相关JS --%>

<script type="text/javascript">
// 为了防止命名冲突, 为每个页面定义唯一的操作命名空间
// 所有函数注册进命名空间
var sysRole = {};
$(function() {
    /*
     * datagrid数据格式化
     */
    // ...

    /*
     * 数据表格初始化
     */

    /*
     * 数据表格CRUD
     */

    /*
     * 搜索
     */

});
</script>

<%-- 2. 页面内容, 无需Body --%>
<!-- ... -->

<%-- grid右键菜单 --%>
<!-- ... -->

<%-- 3. 包含的Dialog页面等其他内容 --%>
<!-- ... -->
```

3. **访问和权限配置**

1. 在`struts_easyssh_dispatcher.xml`中配置访问WEB-INF下JSP视图的action请求

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN" "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
    <!-- easyssh 转发页面包, 通过action 配置跳转到 /WEB-INF/ 下的JSP页面 -->
    <package name="easyssh-dispatcher" extends="easyssh-default" namespace="/">

        <!-- ... -->

        <action name="toDept">
              <result>/WEB-INF/content/main/module/dept.jsp</result>
        </action>
        <action name="toEmp">
              <result>/WEB-INF/content/main/module/emp.jsp</result>
        </action>
    </package>
</struts>
```

2. 通过系统管理员, 添加新模块`菜单权限`(toDept.action, 点击菜单访问JSP视图的action请求)
3. 通过系统管理员, 添加新模块`操作权限`(增删改查的action请求, 显示控制的action名称)
4. 针对用户角色按需`分配菜单权限和操作权限`

Other configuration instructions

1. EasySSH Ajax响应信息

- EasySSH Ajax 响应消息结构:

```
{
  statusCode：响应状态码,
  msg: 响应消息,
  callback: 执行回调函数,
  locationUrl: 跳转页面,
  //... 其他数据
}
```

- BaseAction中输出JSON结果相关的方法：

```
// 自定义JSON输出根对象
setJsonRoot(Object);
// JSON数据初始化, 包含自定义JSON键值对
setJsonMap(Object...)
// JSON数据初始化, 包含自定义JSON键值对, 及EasySSH Ajax响应信息
setJsonMsgMap(Object...)
// JSON数据初始化, 包含自定义JSON键值对, 分页信息, 及EasySSH Ajax响应信息
setJsonPaginationMap(PageBean, Object...)
```

2. BaseAction

BaseAction中定义了以下主要内容：

- Servlet API(request, application)
- JSON result常量： `final String JSON = "json";`
- JSON参数(请求响应相关的EasyUI分页参数, EasySSH Ajax消息参数)
- JSON响应初始化工具方法(`setJsonRoot`, `setJsonMap`, `setJsonMsgMap`, `setJsonPaginationMap`)

```
// 自定义JSON输出根对象
setJsonRoot(Object);
// JSON数据初始化, 包含自定义JSON键值对
setJsonMap(Object...)
// JSON数据初始化, 包含自定义JSON键值对, 及EasySSH Ajax响应信息
setJsonMsgMap(Object...)
```

```
// JSON数据初始化, 包含自定义JSON键值对, 分页信息, 及EasySSH Ajax响应信息
setJsonPaginationMap(PageBean, Object...)
```

- 工具方法

3. CommonDAO

CommonDAO提供了通用的持久层操作类,封装了各种日常操作方法,包含了基于PageBean和EasyCriteria的分页及查询条件处理组件.

4. PageBean分页和查询条件处理

- EasySSH使用PageBean对分页信息进行封装,`findByPage(PageBean)`可以实现分页查询:

```
// Demo:
//        PageBean pb = new PageBean();
//        pb.setSelect(" select new Account(id,username) ");// select查询条件, 可选, 默认为空
//        pb.setEntityName("Account"); // 查询的实体名, 必须
//        pb.setPageNo(1); // 查询页数, 可选, 默认为1
//        pb.setRowsPerPage(4);// 每页条数, 可选, 默认为10
//        pb.setCondition(" and id>2");// 查询条件, 可选, 默认为空
//  pb.addCondition(" and name='A'"); //追加查询条件
//        pb.setSort("id");// 排序字段, 可选, 默认为空
//
//        commonDAO.findByPage(pb); //执行查询
//
//        System.out.println(pb.getPageNo());// 当前页
//        System.out.println(pb.getRowsPerPage());// 每页条数
//        System.out.println(pb.getData());// 分页查询到的数据集合
//        System.out.println(pb.getRowsCount());// 总条数
//        System.out.println(pb.getPageTotal());// 总页数
```

- EasySSH使用EasyCriteria类型的对象对查询条件进行封装,`findByPage(PageBean, EasyCriteria)`可以在查询同时传入条件:
  - **criteria查询条件类编写(主要实现getCondition()方法,返回条件字符串)**

```
/**
 * SysUser查询标准条件类
 *
 * @author easyproject.cn
 * @version 1.0
 *
 */
public class SysUserCriteria extends EasyCriteria implements java.io.Serializable {
    /*
     * 1. 条件属性
     */
    private String name;
    private String realName;
    private Integer status; // 用户状态：0启用；1禁用；2删除

     /*
     * 2. 构造方法
     */
    public SysUserCriteria() {
    }

    public SysUserCriteria(String name, String realName, Integer status) {
        super();
        this.name = name;
        this.realName = realName;
        this.status = status;
    }

     /*
     * 3. 条件生成抽象方法实现
```

```
     */
    public String getCondition() {
        values.clear(); //清除条件数据
        StringBuffer condition = new StringBuffer();
        if (StringUtils.isNotNullAndEmpty(this.getName())) {
            condition.append(" and name like ?");
            values.add("%" + this.getName() + "%");
        }
        if (StringUtils.isNotNullAndEmpty(this.getRealName())) {
            condition.append(" and realName like ?");
            values.add("%" + this.getRealName() + "%");
        }
        if (StringUtils.isNotNullAndEmpty(this.getStatus())) {
            condition.append(" and status=?");
            values.add(this.getStatus());
        }
        return condition.toString();
    }

     /*
     * 4. Setters & Getters...
     */
}
```

- Service使用

```
@Override
public void findByPage(PageBean pb,SysUserCriteria sysUserCriteria) {
    pb.setEntityName("SysUser s");
    pb.setSelect("select s");

 // 按条件分页查询
    commonDAO.findByPage(pb, sysUserCriteria);
}
```

5. 权限配置

- **访问权限配置**
  1. 添加菜单权限和操作权限
  2. 为角色分配菜单权限和操作权限
  3. 为用户分配角色
- **显示控制权限配置(可参考用户管理模块实现)**
  1. 在操作权限中配置显示权限动作
  2. 在JSP页面引入`easyssh-tags`标签库, 将需要显示控制的内容定义在`showAction`标签内, `action属性`指定必须具有的显示权限动作名称

```
<%@ taglib uri="/easyssh-tags" prefix="e" %>

 <e:showAction action="sysUserDelBtn">
    <div onclick="sysUser.toDelete()" data-options="iconCls:'icon-remove'">删除</div>
 </e:showAction>
```

  3. 按需为用户分配显示权限

6. 附加组件：

- EasyFilter Jave Web请求内容过滤替换组件 (API)

  EasyFilter是一个Jave Web请求内容过滤替换组件, 支持使用properties或xml配置文件自定义过滤配置, 能够对用户请求中的以下信息进行过滤替换：

1. 特殊字符替换(如：<, >特殊标记, 脚本等)
2. 非法关键字替换(如：网络系统中国情不允许的特殊关键词)
3. SQL防注入过滤(如：%,*,or,delete,and等等SQL特殊关键字)

在`easyFilter.xml`中已经预定义了默认的替换配置.

- EasyCommons 通用开发组件(API)
  - EasyCommons-imageutils-1.4.jar

    提供图片压缩, 图片地址提取, 图片水印等工具类.
  - EasyCommons-objectutils-1.7.1.jar

    提供代替Java Properties对象的properties文件操作组件.
  - EasyCommons-propertiesutils-1.4.jar

    提供基于`字段表达式(FieldExpression)`对象属性抽取, 对象属性过滤, 对象属性置空等Obejct对象操作组件.

7. EasyUIEx

EasyUIEx API

EasyUIEx是针对使用jQuery EasyUI框架开发的扩展性插件, 主要对EasyUI框架的常用功能进行了封装和扩展, 能大大提高EasyUI的易用性, 简化操作的复杂性, 并提供附加功能.

在进行项目开发时使用jQuery EasyUI + EasyUIEx 架构能大大简化EasyUI框架使用的复杂性, 尤其在各种数据网格的CRUD方面, 做了高度封装.

在项目中, 优先查询EasyUIEx API来完成功能能达到事半功倍的作用.

End

Comments

If you have more comments, suggestions or ideas, please contact me.

Email：inthinkcolor@gmail.com

http://www.easyproject.cn