一个新闻发布系统 - 个人笔记

Version: 0.0.1

目录

序言 iv
1. anews是什么？ iv
2. 希望anews成为：成功整合流行工具，并且很好用，的架构范例。 iv
1. 技术选型 1
1.1. 选择哪些工具包做整合 1
1.2. 待选方案 1
1.3. 运行环境 2
1.4. 测试 3
1.5. 项目管理 3
2. 设计概述 5
2.1. 核心设计 5
2.1.1. 领域模型 5
2.1.2. manager层 5
2.1.3. controller层 5
2.1.4. 特定功能的封装 5
2.2. 权限模型 5
2.3. 表现层View 6
3. 我们要解决什么问题 7
3.1. 我们要解决什么问题 7
3.2. 数据字典 7
3.2.1. NamedEntityBean.java 7
3.2.2. NamedEntityDao.java 7
3.2.3. 应用场景 8
3.3. 树形结构 9
3.3.1. TreeEntityBean.java 9
3.3.2. TreeEntityDao.java 11
3.3.3. 应用场景 11
3.4. extjs与springmvc结合，实现JsonTree 11
3.5. extjs与springmvc结合，实现JsonGrid 11
3.6. 让extjs与acegi对接 11
4. 技术手册 13
4.1. spring 13
4.1.1. 在xml中使用spring-2.x的DTD。 13
4.1.2. default-lazy-init 13
4.1.3. default-autowire="byName" 13
4.1.4. import 13
4.1.5. CharacterEncodingFilter 13
4.1.6. IntrospectorCleanupListener 14
4.1.7. PropertyPlaceholderConfigurer 14
4.1.8. PropertyOverrideConfigurer 14
4.1.9. spring-2.x对aop和事务管理的简化配置 15
4.2. hibernate 16
4.2.1. 基于annotation的JPA式配置 16
4.2.2. 一对多 17

4.2.3. 多对多 17
4.2.4. 继承 17
4.3. acegi权限管理 17
4.3.1. 使用Filter拦截URL请求 17
4.3.2. 使用interpreter拦截METHOD调用 17
4.3.3. ACL如何实现？ 17
4.4. springmvc 18
4.5. extjs 19
4.5.1. Ext.lingo.JsonTree 19
4.5.2. Ext.lingo.JsonGrid 19
4.6. 测试与代码分析 19
4.6.1. spring-mock 19
4.6.2. cobertura 20
4.6.3. 代码分析 20
4.6.4. JDepend：管理代码依赖性 20
A. 修改日志 22

序言

写在所有之前

1. anews是什么？

先摘一段网上看到的评价

> Java最不好的地方就是：它其实没有一个Web框架，只有零敲碎打的一堆东西。
>
> Hibernate伟大吗？伟大。
>
> Spring伟大吗？伟大。
>
> Struts？Freemarker？Acegi？都很伟大。
>
> 但一堆伟大的东西凑合在一起，就谈不上伟大，而是痛苦。把这些东西凑在一起，就是配置的地狱~~
>
> 这就为什么ROR，或者.NET，或者别的什么开发效率更好的原因之一吧。

所以说java公司不容易啊，自己要把那么一堆“零敲碎打”的东西组合在一起，还要考虑包之间会不会冲突，怎么样才能简化配置。终于配好了，这套东西实际中好不好用还两说。

2. 希望anews成为：成功整合流行工具，并且很好用，的架构范例。

一个小程序当然不可能涵盖所有的行业，anews目前也仅仅是一个新闻发布系统。不过以后会变成什么样子，就不是我们该去关心的了。

我们现在应该关心的是： 选择哪些工具包， 关注哪些实际业务。

第 1 章 技术选型

1.1. 选择哪些工具包做整合

1. spring [http://www.springframework.org]

   我们需要它的pi依赖注入作为胶水，把其他工具包粘在一起

   还需要它的aop，进行事务配置，权限管理

   mvc使用springmvc [http://www.springframework.org]，因为extjs [http://extjs.com]使得mvc这一层变得不必要了。

2. hibernate [http://www.hibernate.org]

   作为ORM方案，使用hibernate3 [http://www.hibernate.org]对JPA的支持，使用annotation简化配置。

3. acegisecurity [http://acegisecurity.org]

   与spring [http://www.springframework.org]结合，控制权限

4. extjs [http://extjs.com]

   提供js控件，实现富客户端

   extjs [http://extjs.com]与服务器之间交互采用json格式传递数据，目前采用的json-lib-2.0 [http://json-lib.sourceforge.net]在多线程环境下会有同步问题，2.1-snapshot已经解决了这个问题，但正式版尚未发布。json-lib [http://json-lib.sourceforge.net]在处理循环引用的时候也有很大的问题，处理hibernate [http://www.hibernate.org]双向关联的时候100%出错。

   考虑自己写一个json实现，参考dwr [http://www.getahead.ltd.uk/dwr/]中生成s1={};s2={};s1.s2=s2;的方式。

5. dwr [http://www.getahead.ltd.uk/dwr/]

   使用dwr [http://www.getahead.ltd.uk/dwr/]和 commmons-fileupload [http://commons.apache.org]制作上传文件的进度条。

   考虑使用dwr-2 [http://www.getahead.ltd.uk/dwr/]的服务器推，制作在线聊天。

1.2. 待选方案

（以后可能用上，也可能用不上的工具包）：

1. Web Service

   CXF [http://incubator.apache.org/cxf/]是由Objectweb Celtix和Codehaus XFire合并成立的。与axis1 [http://ws.apache.org/axis/]~2 [http://ws.apache.org/axis2/]比较，更易于整

合到spring [http://www.springframework.org]中。如果以后需要用到Web Service的话，它一定是首选。

2. JMS

   activeMQ [http://activemq.apache.org/]，根据springside [http://www.springside.org.cn]的推荐，使用activeMQ [http://activemq.apache.org/]作为jms的实现方式。

3. 全文搜索

   compass [http://opensymphony.com/compass/] 同另外一个hibernate-search [http://hibernate.org/410.html]都是基于 lucence [http://lucene.apache.org/]，对数据库的全文搜索引擎。

   之前使用过compass [http://opensymphony.com/compass/]，在单元测试的时候会出现内存溢出的问题。

4. 定时调度

   quartz [http://opensymphony.com/quartz/]，被springside [http://www.springside.org.cn]称为唯一免费的选择。

5. 报表

   JasperReport [http://jasperforge.org/sf/projects/jasperreports]负责文本报表，JFreeChat [www.jfree.org/jfreechart/]图形报表。

6. 工作流

   jbpm [http://jboss.org]，目前仅用过的工作流引擎。osworkflow [http://opensymphony.com/osworkflow/]，号称最灵活的工作流引擎。两者都支持hibernate [http://www.hibernate.org]。

1.3. 运行环境

1. jdk5

   支持范型generic和注释annotation，可大大减少配置提高抽象程度。

   考虑用Retrotranslator [http://retrotranslator.sourceforge.net/]使项目可以在JDK1.4上运行

2. servlet容器：tomcat-5.5 [http://tomcat.apache.org]

   应用甚广的服务器，基本用到jsp的人都会有。

   maven2 [http://maven.apache.org]可以使用 jetty-6 [http://jetty.mortbay.com/]作为插件进行运行和测试，单独使用jetty的人不多。

3. 数据库

   采用hsqldb [http://hsqldb.org]作为嵌入式数据库，可与工程绑定发布，用户下载体验版后不

需要额外安装数据库，减少了配置出错的机会。

hibernate本身支持多数据库，包括mysql,sqlserver,oracle等，未来替换成企业数据库也不是问题。

4. 数据部署

DBDeploy [http://dbdeploy.com]让数据库脚本也可以实现版本控制。

1.4. 测试

1. 单元测试

junit [http://junit.org/]。经典单元测试工具，基本所有工具都支持。

js的单元测试工具 jsunit [http://www.jsunit.net]不知如何应用。

2. mock

easymock [www.easymock.org/]使用烦琐，但有的时候还必须依靠它制作mock进行测试。更多时候使用spring-mock [http://www.springframework.org]对servlet进行测试，spring-mock [http://www.springframework.org]还提供了几个常用的测试基类，可以缓存xml配置，支持事务。

3. 测试覆盖率

cobertura [http://cobertura.sourceforge.net/]，免费的测试覆盖率检测工具，1.9版本已经慢慢追上clover [http://www.atlassian.com/software/clover/]的效果了，从maven2 [http://maven.apache.org]的角度来看，配置比clover [http://www.atlassian.com/software/clover/]更简便，而且不需要像clover [http://www.atlassian.com/software/clover/]那样每月去申请授权。

4. 集成测试

selenium [http://openqa.org/selenium/]被众口称赞的集成测试工具，不过想灵活应用还需要一定的时间。

同类产品还有httpunit [httpunit.sourceforge.net/]和jwebunit [http://jwebunit.sourceforge.net/]。

使用maven2 [http://maven.apache.org]做集成测试时，可以搭配cargo [http://cargo.codehaus.org/]控制服务器的启动与停止。

5. 压力测试

jmeter [http://jakarta.apache.org/jmeter/index.html]听过但是没用过。

1.5. 项目管理

使用maven2 [http://maven.apache.org]进行项目管理，配置了jalopy

[http://jalopy.sourceforge.net/]对java代码进行美化，支持多种插件，进行自动打包，测试，代码分析，生成报表。

1. checkstyle [http://checkstyle.sourceforge.net]进行源代码格式分析。
2. pmd, cpd [http://pmd.sourceforge.net]检查重复代码。
3. findbugs [http://findbugs.sourceforge.net]检查程序漏洞。

第 2 章 设计概述

2.1. 核心设计

2.1.1. 领域模型

采用hibernate3支持的jpa方式实现ORM数据持久化。

这方面利用了jdk5的annotation注解和spring提供的AnnotationSessionFactoryBean，实现零配置文件。

2.1.2. manager层

no dao, no interface，将传统的五层结构缩减为两层，子类只需继承HibernateEntityDao就自动获得CURD功能。通过支持jdk5的generic范型，子类中的方法无需进行类型转换便可获得需要的类型，进一步简化操作。

到需要分层的时候，可以从manager内部拆分，提取对应的接口。

2.1.3. controller层

子类只需继承BaseController就能自动获得CURD功能。

利用spring-2.0新增的request范围，使每次请求生成一个controller实例进行处理。controller不再是被多个线程共同享有，这样可以简化方法定义，同时便于提供多个帮助方法，代码更简洁。

2.1.4. 特定功能的封装

为实现特定功能，提供一系列基类与帮助类。

1. NamedManager用于处理数据字典，提供createOrGet()方法，实现功能为先在数据库查找同名记录，如果存在则返回这条记录，如果记录不存在，则将名称保存进数据库，并返回保存记录对应的对象。

2. TreeManager用户处理自关联树形结构，如菜单，组织结构，提供getTops()方法，获得顶级节点的集合。

   TreeHelper提供检测parent与child之间是否存在循环关联的checkDeadLock()方法。

   进一步到达表示层，使用extjs封装的树形控件，少量代码即可实现树形的常用功能。

2.2. 权限模型

采用acegisecurity，实现基于RBAC（Role Based Access Control基于角色的访问控制）的权限模型管理。

acegisecurity本身自带多种验证方式，这里仿效springside-2.0中的security模块，对其数据模型进行了扩展，通过User用户 - Role角色 - Resource资源三者之间的多对多关系实现权限管理，附加Role角色与Menu菜单之间的关联，日常调度时，只需要为用户选择角色，就可以获得对应的访问权限。

URL资源的访问使用filter过滤器拦截，METHOD资源的访问使用aop拦截器拦截。

ACL（Access Control List访问控制列表）尚未实现。

2.3. 表现层View

网站后台的页面使用extjs渲染，用到了Layout布局管理，Grid表格，Tree树形，Dialog对话框等控件。extjs华丽效果的代价就是加载速度变慢，压缩后400多k的js文件还是需要好好考虑如何处理的。

extjs依然是目前见过的最完美的控件库，而且允许开发者在LGPL写一下免费使用。

extjs灵活的封装，无需修改即可以与不同后台对接，支持包括JSON,XML等多种数据格式，也让开发者有了更多的选择。目前我们采用的是json格式传输数据，不过json-lib-2.0在多线程并发时有缺陷，亟待解决。

第 3 章 我们要解决什么问题

3.1. 我们要解决什么问题

将同类项目中的类似功能抽象出来，提供特定的解决方案，模板，以及工具类。

让使用者可以通过：理解设计思想，参考模板实现，调用工具类三方面，更快的解决同类问题。

3.2. 数据字典

数据字典是系统中一种常用的数据结构，主要特征有：

1. 对用户来说，只有“名称”这一个字段是有意义的
2. 数据字典的名称不能重复

3.2.1. NamedEntityBean.java

```
package anni.core.domain;

import javax.persistence.Column;
import javax.persistence.Id;

public class NamedEntityBean {
    private Long id = null;
    private String name = null;

    @Id
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }

    @Column(unique = true, nullable = false)
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

pojo中定义id与name，注意在getName()使用的注释@Column(unique=true,nullable=false)，名称不能重复，不能为空。在插入的时候还应该判断不能为空字符串，也不能全是空格。

3.2.2. NamedEntityDao.java

dao中定义createOrGet方法，与其对应的是在前台显示的时候，要使用可以选择也可以输入的autocomplete输入框，与google的自动补完搜索框类似，既可以选择已有的数据字典，也可以输入一个新记录。

```
/**
 * 如果数据库中不存在指定name的记录，就将此条记录插入数据库，并返回对应的实体类.
 * 如果数据库中已经存在指定name的记录，就返回对应的实体类
 *
 * @param name 数据字典的name
 * @return T 返回对应的实体类
 *           当name为null或空字符串时，返回null
 *           当构造pojo实例时出现异常，也返回null
 */
public T createOrGet(String name) {
    // 输入的名称不应该为空.
    if ((name == null) || "".equals(name.trim())) {
        return null;
    }

    String beanName = name.trim();
    List<T> list = findBy("name", beanName); // 这句是不是可以改成findByUnique? 哪个效率更高?

    // 如果找到记录，应该是只有一项目，就返回这条记录的实体类
    if (list.size() > 0) {
        return list.get(0);
    } else {
        try {
            // 如果没找到记录，需要把这个数据字典保存进数据库
            // 需要pojo拥有空的构造方法
            T namedEntityBean = (T) getEntityClass().newInstance();
            namedEntityBean.setName(beanName);
            save(namedEntityBean);

            return namedEntityBean;
        } catch (InstantiationException ex) {
            System.err.println(ex);
        } catch (IllegalAccessException ex) {
            System.err.println(ex);
        }

        return null;
    }
}
```

3.2.3. 应用场景

产品录入界面，产品规格为数据字典，录入产品的时候，可以选择已有的规格，createOrGet()方法返回的是数据库中已有的记录。

当出现之前从未录入的规格，不需要打开数据字典管理页面，添加新规格，只需要输入新规格名称，createOrGet()方法自动将此规格名称保存到数据库，并返回对应实例，进行关联。

anni.anews.domain.NewsTag就是这种数据字典。

3.3. 树形结构

组织结构，系统菜单，产品分类，都是以树形显示的的数据结构，主要特征有：

1. 最少包括：id，名称，父节点id三个字段，如果需要排序，还要有排序字段。
2. 设计数据库表结构时，父节点id是自表关联的外键。
3. 可能有其他字段属性，不同级别下的名称可重复。

3.3.1. TreeEntityBean.java

```
package anni.core.domain.tree;

import java.util.HashSet;
import java.util.Set;

import javax.persistence.Id;
import javax.persistence.Column;
import javax.persistence.CascadeType;
import javax.persistence.FetchType;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.MappedSuperclass;
import javax.persistence.OneToMany;
import javax.persistence.OrderBy;
import javax.persistence.Transient;


/**
 * id是Long的Tree类实现.
 *
 * @author Lingo
 * @since 2007-06-06
 * @param <T> 本身子表关联，T代表的就是本身的类型
 */
public class LongTreeEntityBean<T extends LongTreeEntityBean> {
    private Long id = null;
    private String name = null;
    private Integer theSort = null;
    private T parent = null;
    private Set<T> children = new HashSet<T>(0);

    @Id
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }

    @Column
    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    @Column
    public Integer getTheSort() {
        return theSort;
    }
    public void setTheSort(Integer theSort) {
        this.theSort = theSort;
    }

    // 多对一关联
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "PARENT_ID")
    public T getParent() {
        return parent;
    }
    public void setParent(T parent) {
        this.parent = parent;
    }

    // 一对多
    // 查找children的时候，根据theSort正序排列，theSort相同时，根据id倒序排列
    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "parent")
    @OrderBy("theSort asc,id desc")
    public Set<T> getChildren() {
        return children;
    }
    public void setChildren(Set<T> childrenIn) {
        children = childrenIn;
    }

    /**
     * @return 是否为根节点.
     */
    @Transient
    public boolean isRoot() {
        return parent == null;
    }

    /**
     * @return 是否为叶子节点.
     */
    @Transient
    public boolean isLeaf() {
        return (children == null) || (children.size() == 0);
    }

    /**
     * 不允许将自表外键关系设置成环状.
     * 就是说，当前bean的子节点中，如果包含entityBean，就不能把entityBean设置为当前bean的上级节点
     *
     * @param entityBean TreeEntityBean
     * @return boolean 是否形成环状
     */
    @Transient
    public boolean checkDeadLock(T entityBean) {
        return TreeHelper.checkDeadLock(this, entityBean);
```

```
    }
}
```

包含了id主键，name名称，theSort排序编号，parent上级节点，children下级节点集合的pojo。并且提供了三个工具方法，isRoot()判断节点是否为根节点，isLeaf()判断节点是否为叶子节点，checkDeadLock(entityBean)判断pojo本身与作为参数传入的entityBean是否形成环形引用，避免把节点的parent关联到自己的子节点上。

3.3.2. TreeEntityDao.java

提供工具类，获得所有根节点的列表。

```
    public List<T> loadTops() {
        return find("from " + getEntityClass().getName()
            + " where parent is null");
    }
```

3.3.3. 应用场景

树形结构应用比较广泛，各种菜单，分类，部门结构都可以使用树形表示

新闻发布中的新闻目录，权限管理中的部门，菜单都使用了树形结构。

3.4. extjs与springmvc结合，实现JsonTree

TODO:

3.5. extjs与springmvc结合，实现JsonGrid

TODO:

3.6. 让extjs与acegi对接

对接的目的是，保证前台绚丽的弹出登录窗口，登录成功后华丽的窗口消失效果，登录失败时也要漂亮的闪现错误提示，不要多次跳转。

首先为了让acegi修改最少，使用ajax将帐号和密码提交到Filter中。这时候分四种情况考虑：

1. 首次登录成功，跳转到/sandbox/welcome.htm。返回已登录用户的真实姓名。

   接收json后进行的操作：打开左侧菜单，根据登录用户的信息，显示对应菜单。显示当前登录用户的真实姓名。

2. 首次登录失败，返回json格式的出错信息，可能为：用户不存在，密码错误，无法重复登录，验证码错误。

   这时候需要根据返回的错误信息，弹出alert对话框，或者suggest框。

3. 访问失败后，再次登录，验证后拥有此权限。

   打开左侧菜单，设置真实姓名，为iframe填充返回的页面信息。

4. 访问失败后，再次登录，验证后依然没有权限。

   打开左侧菜单，设置真实姓名，为iframe填充返回的错误信息。（这时候，其实是按照正常的URL去请求，不过因为没有权限，所以半路被拦截，变成了显示无权限的消息。）

这样，每次无论成功失败。都要求返回的是json，改造AuthentikcationProcessFilter，将所有跳转命令都转向LoginController，在LoginController中构造json返回。

因为只将登录的过程修改为返回json，所以覆盖successfulAuthentication()方法，将最后的sendRedirect()修改为转向LoginController.loginSuccess方法，同时把targetUrl需要跳转的URL附带上，在logingSuccess()里生成json，附带success:true登录成功标志，response:用户真实姓名，callback:iframe需要显示的URL。在js里处理后，完成整个对接过程。

js方面，展开对话框，处理登录的代码分别在widgets/sandbox/index.js与widgets/lingo/form/LoginDialog.js中。不过真正发起检验登录的入口在needLogin.ftl中，在其中调用window.top.index.isLogin()，当注销的时候，也会跳转到needLogin.htm来触发登录事件。

第 4 章 技术手册

4.1. spring

胶水技术，依靠DI黏合不同工具，让他们在一起运行。

4.1.1. 在xml中使用spring-2.x的DTD。

```
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "http://www.springframework.org/dtd/spring-beans-2.0.dtd">
```

相比schema真是方便了很多，这样可以在必须使用特定schema的时候，再让他重装上阵。

spring-2.x中去掉了singleton属性，使用scope属性做替代。如果还想使用singleton属性，必须配置成spring-1.x格式的DTD。

> 提示
> spring中已不再推荐使用singleton属性，因为单例在多jvm，远程调用，集群的情况下难以掌控，还是为实例指明生存的scope比较好。

4.1.2. default-lazy-init

懒惰加载，系统启动的时候并不加载xml中定义的bean，而是等到实际调用的时候才去加载，这样可以缩短系统初始化时间，在测试系统部分功能的情况下有极大的好处。

注意，springmvc，xfire，quartz等的配置文件不能声明为懒惰加载，否则会出问题。

4.1.3. default-autowire="byName"

按名称自动绑定。设置了这个，只要定义bean的时候名称与需要绑定的属性名相同，在实例化对象的时候，spring就会将这些实例自动绑定，不需要再去声明绑定哪些property。减少xml代码量，使得结构更清晰。

在使用compass的时候要注意，不能使用按名称自动绑定，会自动为compass绑定dataSource导致错误。

4.1.4. import

可以使用import导入多个配置文件

```
<import resource="classpath:spring/controller/admin-controller.xml"/>
```

这样可以把xml分成多个模块管理，如mail.xml里只配置邮件相关的配置。

4.1.5. CharacterEncodingFilter

spring提供的编码过滤器，好处一是不用自己动手写了，好处二是保证每次请求只过滤一次。配置如下：（web.xml）

```
<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

4.1.6. IntrospectorCleanupListener

spring提供的监听器，避免Struts，Quartz的内存泄露导致ClassLoader不能加载。配置如下：（web.xml）

```
<listener>
    <listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>
</listener>
```

4.1.7. PropertyPlaceholderConfigurer

读取properties中的变量，在xml中可以通过${变量名}的方式调用。配置如下：

```
<bean id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
        <list>
            <value>classpath*:conf/jdbc.properties</value>
            <value>classpath*:conf/hibernate.properties</value>
            <value>classpath*:conf/mail.properties</value>
        </list>
    </property>
</bean>
<bean id="foo" class="anni.Foo">
    <property name="test" value="${test}"/>
</bean>
```

4.1.8. PropertyOverrideConfigurer

与PropertyPlaceholderConfigurer不同，PropertyOverrideConfigurer会在ApplicationContext初始

化后，根据properties中的定义，修改对应属性的值。配置如下：

```
    <bean id="testPropertyConfigurer" class="org.springframework.beans.factory.config.PropertyOverrideConfigurer">
        <property name="locations" value="classpath*:test/override.properties"/>
        <property name="ignoreInvalidKeys" value="true"/>
    </bean>
```

用来在测试环境下覆盖已有的配置，比如在override.properties中有foo.test=111，那么id="foo"的bean的test属性就会被修改为111。

4.1.9. spring-2.x对aop和事务管理的简化配置

1. 首先要使用schema

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.0.xsd
           http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.0.xsd">
```

2. 然后要声明标记（不知道是不是必要的）

```
    <!-- 支持 @Transactional 标记 -->
    <tx:annotation-driven/>

    <!-- 支持 @AspectJ 标记-->
    <aop:aspectj-autoproxy/>
```

3. 配置aop

```
    <aop:config proxy-target-class="true">
        <aop:advisor pointcut="execution(* anni.anews.manager..*Manager.*(..))" advice-ref="txAdvice"/>
        <aop:advisor pointcut="execution(* anni.anews.dao..*Dao.*(..))" advice-ref="txAdvice"/>
        <aop:advisor pointcut="execution(* anni.core.dao..*Dao.*(..))" advice-ref="txAdvice"/>
    </aop:config>
```

4. 配置txAdvice处理事务

```
    <tx:advice id="txAdvice">
        <tx:attributes>
```

```
            <tx:method name="get*" read-only="true"/>
            <tx:method name="find*" read-only="true"/>
            <tx:method name="pagedQuery*" read-only="true"/>
            <tx:method name="load*" read-only="true"/>
            <tx:method name="*"/>
        </tx:attributes>
    </tx:advice>
```

提示

name-pattern千万不要写成*..*Manager ，这样子会把所有第三方类库的Manager比如Spring的PlatformTranstationManager 也加入aop，非常危险。所以最好还是加上项目的package前缀，如"anni.anews..*Manager"

因为有*，会修饰所有方法，有些hibernateTemplate的final的方法不能被cglib修改，会抛warning，无害。

4.2. hibernate

感觉是目前最成熟的ORM框架，家族中新添了hibernate-shards支持多数据库访问，hibernate-search提供全文检索，大有潜力可挖。

4.2.1. 基于annotation的JPA式配置

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratorType;
import javax.persistence.Id;

@Entity
public class Test {
    private Long id;
    private String name;

    @Id(generate = GeneratorType.AUTO)
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    @Column
    public String getName() {
        return name;
    }
    public void setName() {
        this.name = name;
    }
}
```

提示

如果pojo要定义getter方法，这个方法数据库的持久字段没有关联，必须使用@Transient标注，否则hibernate会把它当作一个持久字段，去数据库读取的时候会报错。

4.2.2. 一对多

见anni.anews.domain.NewsCategory

4.2.3. 多对多

见anni.anews.domain.News和anni.anews.domain.NewsTag

4.2.4. 继承

尚未用到。之前用过一次，一表多类，不知如何直接取得子类数据。

4.3. acegi权限管理

照搬springside-2.0的权限模型，实现User用户 - Role角色 - Resource资源三种组成的RBAC模型。

稍微注意一下，默认角色名称都要以ROLE_开头，否则不会被计入权限控制，如果需要修改，就在xml里配个什么前缀的。

4.3.1. 使用Filter拦截URL请求

使用在web.xml中配置的一系列filter实现对URL请求的拦截。

4.3.2. 使用interpreter拦截METHOD调用

采用aop方式，实现方法调用拦截。

4.3.3. ACL如何实现？

1. 根据springside的JIRA描述，实现基本的层次式ACL 管理，（Role A拥有部门A的审批，部门B的查看权限，请假单属于部门A，则Role A有该单的审批权限）。与插件式的规则引擎管理。
2. 来自： http://www.blogjava.net/RongHao/archive/2006/12/14/87634.html对acegi ACL扩展的构想

   扩展一：当你增加一条记录的同时向ACL表里插入权限信息，这个ss已经做到了，ss有一个借口标记那些domain需要ACL保护AclDomainAware。这里再扩展一个基类，目的很简单，当读出数据时附加上权限信息

```
public class BasicAclDomain implements AclDomainAware {
    private int mask; // 权限
```

```
    getMask
    setMask
}

public class Contact extends BasicAclDomain
```

扩展二：读取列表时进行数据过滤，原来的acegi在ACL的集合后处理会造成分页虎牙子。这里采用前拦截，在acl表里增加一个className字段，里面放上PO的类名，这样可以缩小数据查询范围。实际在读取集合时，先到acl表里完成分页，然后多的对contact的real id list，然后拦截实际DAO方法，动态改变SQL成select * from real_data where id in ({real id list})的形式。

分页实际是对acl表里相应记录的分页，比如说取第10条到20条，实际是取acl表里响应记录的第10条到20条来动态改变sql。

这个可以写一个专门用来拦截的类SecurityDAO方法，findByPageACL(query, page),ContactServiceImpl中getAllContacts方法强制调用该方法。

提示

这个方法在javaeye中看到了讨论，问题就在于：real id list过大时，数据库会将sql剪切。
所以数据量大的时候没法用，还有人说如果需要动态设置其他查询或排序条件，还会有问题，
具体是什么就没说，可能是解析出问题，或者效率问题

扩展三：后拦截。这里AFTER_ACL_COLLECTION_READ和AFTER_ACL_READ的目的就很简单了，因为他们不再进行数据过滤，他们只是把用户对每条记录的mask取最大权限就OK了，然后往PO级setMask。这样PO带了权限信息到页面上行就非常好处理了。比如button的显示等等。

扩展四：封装AclService，对单条记录的ACL权限管理。比如增加权限，修改权限，删除权限。这个acegi从1.0.3已经开始加入了。

还未实现的构想一：PO创建向ACL表里插入权限信息时，可以配置不同策略：比如通讯录创建一条新信息只能创建者可以看并管理，几你往请假表里插入一条新信息后，不仅你了，你的上司也可以同时看到。（这个还是比较easy的）。

还未实现的构想二：用户数据要实现数据库排序。需要在ACL_OBJECT_IDENTITY里增加几个额外字段，把PO相应的排序字段同步更新到ACL表中。什么？不好做？写配置文件啊！再对PO的update进行后拦截，想法就这样。实现？？

4.4. springmvc

PrototypeController基于springside-1的思想，对controller进行封装，采用spring-2.x提供的惯例配置的同时，将controller改造成每个请求生成一个实例的形式，这么做主要是为了省略方法中的HttpServletRequest和HttpServletResponset两个参数，将controller改造成jsp里那样，可以直接获得request, response, session, application, mv几个默认参数，无需每次手工创建。

ExtendController在这基础上提供了getLongParam(), getStrParam()几个帮助方法，方便快捷获得请求的参数。getLongParam("name", defaultValue)都可以设置默认值，在参数不存在的情况下自动返回默认值，不必担心NumberFormatException之类的错误，方法内已经做了自动处理，进一步简化操作。

BaseController是在ExtendController的基础上，基于generic范型封装的基类，子类继承它即可获得CURD操作。

JsonController也是在ExtendController基础上进行封装，默认为ModelAndView设置不渲染view，在controller中直接生成json字符串，写入到response.getWriter()中。

4.5. extjs

js控件库，可选择基于yui,prototype,jquery或者ext-base底层实现，在这些底层基础上构筑了丰富的组件库。采用oo风格封装，易于使用和扩展。

目前主要封装了Ext.lingo.JsonTree和Ext.lingo.JsonGrid，采用json格式的数据与后台交互，实现CURD等通用功能。

4.5.1. Ext.lingo.JsonTree

树形结构，适用于单表自关联结构。

1. 异步读取节点（不过管理分类的时候需要读取所有节点）
2. 双击节点编辑节点内容
3. 拖拽排序
4. 右键弹出菜单，进行详细配置

4.5.2. Ext.lingo.JsonGrid

实现CURD功能的表格。

1. 数据库端分页
2. checkbox全选，多选行
3. 按字段模糊搜索
4. 弹出对话框，进行新增或修改数据

4.6. 测试与代码分析

4.6.1. spring-mock

使用spring-mock提供的几个超类非常有好处，spring提供了可以一次读取xml并缓存的机制，让所有testCase都可以共享同一份applicationContext，大大加快了测试速度。

需要注意的是获得ctx缓存的key是根据getConfigLocation()获得的，如果在子类中覆盖了这个方法，返回不同的数值，会导致重新加载。同样的如果不覆盖方法返回不同值，那么使用的都是同一份ctx。

4.6.2. cobertura

因为1.8版本的一漏洞，每次生成覆盖率报表都是100%。幸好现在的1.9已经解决了这个问题，不过对应的maven2-plugin还没有发布，尝试2.2-snapshot插件使用cobertura-1.9，感觉功能与clover已经很接近了，而且配置也方便。

4.6.3. 代码分析

大体分成三方面

1. checkstyle检查源代码的编码风格，检查javadoc是否完整，缩进，空格是否一致。用来检查大家是否遵循同一种编码规范。
2. findbugs是检查编译后的class是否存在隐性bug，包括无用代码，不好的实践等等。
3. pmd与findbugs功能相似，与其一起工作的cpd可以检查代码中是否存在大量重复代码，如果重复代码过多，就说明代码应该重构了。提出重复代码部分，做成工具类，或者抽象出超类来。
4. jdepend用来检验代码依赖和复杂度，暂时不太会用。报表中可以显示当前代码都与哪些包关联。

4.6.4. JDepend：管理代码依赖性

JDepend遍历所有的Java代码目录，自动生成每个Package的依赖性度量。对于可扩展性、可重用性和可管理性，JDepend可自动度量一个设计在以上三个方面的质量。

JDepend为每个Package自动生成的依赖性度量指标，包括：

1. Number of Classes (Cc)

   被分析package的具体和抽象类（和接口）的数量，用于衡量package的可扩展性。

2. Afferent Couplings (Ca)

   依赖于被分析package的其他package的数量，用于衡量pacakge的职责。

3. Efferent Couplings (Ce)

   被分析package的类所依赖的其他package的数量，用于衡量package的独立性。

4. Abstractness (A)

   被分析package中的抽象类和接口与所在package所有类数量的比例，取值范围为0－1。

5. Instability (I)

   I＝Ce/（Ce＋Ca），用于衡量package的不稳定性，取值范围为0－1。I＝0表示最稳定，I＝1表示最不稳定。

6. Distance (D)

   被分析package和理想曲线A＋I＝1的垂直距离，用于衡量package在稳定性和抽象性之间的平衡。

理想的package要么完全是抽象类和稳定（x＝0，y＝1），要么完全是具体类和不稳定（x＝1，y＝0）。

取值范围为0－1，D＝0表示完全符合理想标准，D＝1表示package最大程度地偏离了理想标准。

为什么使用JDepend

1. 评价设计质量
2. 翻转依赖性
3. 支持并行开发和极限编程
4. 独立的发布模块
5. 识别package的循环依赖

附录 A. 修改日志

修订历史

修订 0.0.1 2007-09-29

1. 初稿完成。序言