Rose_Guide_Application2

Updated Sep 5, 2010 by qieqie.wang

#

第二支程序

- 目的
- 业务介绍
  - 地址规定
  - 登录规定
  - 功能
- URI设计
- 控制器设计
  - HomeController.java
  - book.BookController.java
  - book.RemarkController.java
  - LoginController.java
  - user.UserController.java
  - logs.LogsController.java
- 拦截器设计
  - AutoLogInterceptor.java
  - PassportInterceptor.java
  - LoginRequredInterceptor.java
  - LoginRequired.java
- 数据库表设计
  - user表
  - book表
  - remark表
  - oper_log表
- DAO设计
  - BookDAO.java
  - RemarkDAO.java
  - UserDAO.java
  - OperLogDAO.java
- 资源的提供
  - 数据库配置 WEB-INF/applicationContext-dataSource.xml
  - 执行器配置 WEB-INF/applicationContext-executor.xml
- 资源的使用
  - book.BookController.java
  - AutoLogInterceptor.java
- 视图组件的设计

目的

这个示例内容会比较丰富，通过这个示例的了解，我相信您将能够较为熟练地使用rose开发。

在实践前，我假设您已经把工程的基本建立了，具体过程可参考 第一支程序

业务介绍

这是一个图书库系统，但仅为演示之用，很显然的，它不是真的是一个符合现实的系统。

地址规定

http://localhost/lib 是它的访问地址，其中/lib是contextpath，因此简单地，你需要把完成后的webapp内容放在tomcat的webapps/lib目录下，形成webapps/lib/web.xml、webapps/lib/WEB-INF等文件或目录以及其他各种形式的目录和文件。

登录规定

进入图书库系统必须进行登录，但一旦登录之后不同用户之间的权限没有太大区别，但是所有的操作都有记录。这些操作记录是只读记录，且只有一个名为rose的用户才能看到。

用户的帐号由rose用户进行创建和注销。

用户登录到系统，要列出所有的书本清单，每页30本。

功能

任何用户可增加书、更改书的信息，但只有rose用户才能删书。

任何用户都可以为书编写备注、备注的个数不限制个数。任何人不能删除备注，包括自己的备注，但rose管理员可以删除。

URI设计

id	method	uri	desc
01	GET	/lib/	跳转到/lib/book；
02	GET	/lib/book	按页浏览书库的书，参数byBookId用于定位与分页；
03	POST	/lib/book	增加一本书，参数为name、author、price，都是必填项目；
04	GET	/lib/book/add	显示增加书的页面
05	GET	/lib/book/{bookId}	展示该书的详细信息，包括附属的所有评论，最新的备注在前；
			如果带有edit参数，则返回编辑页面，页面不包含评论
06	PUT	/lib/book/{bookId}	修改该书的信息
07	DELETE	/lib/book/{bookId}/remark	删除该书的所有备注
08	DELETE	/lib/book/{bookId}/remark/{remarkId}	删除某一个备注
09	GET	/lib/login	返回登录页面
10	POST	/lib/login	执行登录判断
11	GET	/lib/user	列出所有用户
12	POST	/lib/user	新增一个注册用户
13	GET	/lib/user/add	显示增加用户的页面
14	GET	/lib/user/{userId}	显示某个用户的详细信息；如果带有edit参数，则返回编辑页面
15	DELETE	/lib/user/{userId}	注销某个用户
16	GET	/lib/logs	按页列出所有用户操作日志

严重说明 :)

上面这个表格中使用了目前浏览器不能支持的PUT、DELETE两个方法，您有两种选择：

- 选择一：不使用PUT、DELETE：在这种选择下，建议您改为POST，但需要在URI增加一级/update、/delete
- 选择二：继续使用PUT、DELETE：在这种选择下，则请求需要以POST提交，并且在表单中增加一个hidden字段：name为_method，value为PUT或DELETE

本应用选择了第二种选择(并不是第一种就是不好的，这只是一个权衡，而且无论选择哪一种优缺点都很明显)。

控制器设计

HomeController.java

```
package com.company.lib.controllers;

// @Path是Rose提供的注解
@LoginRequired
@Path("")
public class HomeController {
    // 1)
    // 这个"1)"表示这个对应于《URI设计》表格中的序号
    @Get
    public String redirect() {
         return "r:/lib/book";
    }
}
```

book.BookController.java

```
package com.company.lib.controllers.book;

// LoginRequird是这个示例特有的注解，下面有对应的解释
@LoginRequired
@Path("")
public class BookController {

    // 2)
    // @Get 表示对空串的Get请求可有此方法处理
    @Get
    public String list(@Param("byBookId") long byBookId) {
        // 控制器方法返回以"@"开头，是为暂时策略;
        // 这样使得在没有页面的情况下不会404错误，当页面准备好的时候要去掉"@"
        // 下同
        return "@book-list";
    }

    // 3)
    @Post
    public String add(Book book) {
        // TODO: insert into database
        if (success) {
            return "r:/lib/book";
        } else {
            return "@book-add";
        }
    }

    // 4)
    // @Get("add") 表示对"add"的GET请求，可有此方法处理
    @Get("add")
    public String showAdd() {
        return "@book-add";
    }

    // 5)
    // 可以通过大括号表示一个变量，通过冒号说明他的正则规则
    // 正则通常是可以省略的(默认规则是除'\'的任何的多个字符)，就像后面还有的{remarkId}，虽然remarkId也应该只是数字
    @Get("{bookId:[0-9]+}")
    public String show(@Param("bookId") long bookId, @Param("edit") boolean isEdit) {
        // TODO: query from database
        if (isEdit) {
            return "@book-edit";
        }
        // TODO: query remarks from database
        return "@book-page";
    }

    // 6)
    // 这里使用目前浏览器无法支持的@Put，你懂的，详看《URI设计》表格中的《严重说明》:)
    @Put("{bookId:[0-9]+}")
    public String update(@Param("bookId") long bookId, Book book) {
        // TODO: update book
        return "r:/lib/book/" + bookId;
    }

}
```

book.RemarkController.java

```
package com.company.lib.controllers.book;

@LoginRequired
@Path("{bookId:[0-9]+}/remark")
public class RemarkController {

    // 7)
    @Delete
    public String clear(@Param("bookId") long bookId) {
        return "r:/lib/book/" + bookId;
    }

    // 8)
    @Delete("{remarkId}")
    public String delete(@Param("bookId") long bookId, @Param("remarkId") String remarkId) {
        return "r:/lib/book/" + bookId;
    }
}
```

LoginController.java

```
package com.company.lib.controllers;

@Path("login")
public class LoginController {

    // 9)
    @Get
    public String show() {
        return "@login";
    }

    // 10)
    @Post
    public String doLogin() {
        return "r:/lib/book";
    }
}
```

user.UserController.java

```
package com.company.lib.controllers.user;

@LoginRequired
@Path("")
public class UserController {

    // 11)
    @Get
    public String list() {
        return "@user-list";
    }

    // 12)
    @Post
    public String add(User user) {
        if (success) {
            return "r:/lib/user";
        } else {
            return "@user-add";
        }
    }

    // 13)
    @Get("add")
    public String showAdd() {
        return "@user-add";
    }

    // 14)
    @Get("{userId}")
    public String show(@Param("userId") String userId, @Param("isEdit") boolean isEdit) {
        if (isEdit) {
            return "@user-edit";
        }
        return "@user-page";
    }

    // 15)
    @Delete("{userId}")
    public String delete(@Param("userId") String userId) {
        return "r:/lib/user";
    }
}
```

logs.LogsController.java

```
package com.company.lib.controllers.logs;

@LoginRequired
@Path("")
public class LogsCotnroller {

    // 16)
    @Get
    public String list(@Param("offset") int offset) {
        return "@logs-list";
    }
}
```

拦截器设计

AutoLogInterceptor.java

```
// 拦截器放在controllers包下，称为局部拦截器，局部拦截器只作用于所在目录以及子目录的控制器
package com.company.lib.controllers;

// 通过扩展 ControllerInterceptorAdapter 实现一个拦截器类；
public AutoLogInterceptor extends ControllerInterceptorAdapter {

    // afterCompletion 在整个页面渲染完毕或者因异常导致流程被中断后执行
    // 无论请求正常或异常地被处理，每一个拦截器的afterCompletion都会被执行
    @Override
    public void afterCompletion(Invocation inv, Throwable ex) throws Exception  {
        String uid = inv.getResourceId(); // e.g. /lib/book/{bookId}
        String uri = inv.getRequestPath().getUri(); // e.g. /lib/book/123456
        boolean success = (ex == null);
        String remarks = success ? null : ex.getMessage();
        // TODO: save it to database
    }
}
```

PassportInterceptor.java

```
package com.company.lib.controllers;

// 设置oncePerRequest为true，表示如果当前的请求如果是被forward、include转来的，并且之前已经执行了该拦截器，则当前不再过该拦截器，在大
@Interceptor(oncePerRequest = true)
public PassportInterceptor extends ControllerInterceptorAdapter {

    public PassportInterceptor() {
        // 设置优先级，优先级越高的拦截器，before方法越先执行
        // PassportInterceptor要比很多拦截器都要先执行，其中包括LoginRequredInterceptor
        setPriority(1000);
    }

    // before方法在调用控制器方法前执行，相反的after则是控制器执行后才执行
    @Override
    protected Object before(Invocation inv) throws Exception  {
        User loginUser = ...;
        if (loginUser != null) {
            inv.getRequest().setAttribute("loginUser", loginUser);
        }
        return true;
    }
}
```

LoginRequredInterceptor.java

```
package com.company.lib.controllers;

public LoginRequredInterceptor extends ControllerInterceptorAdapter {

    public LoginRequredInterceptor() {
        // 这个优先级要小于PassportInterceptor，必须的
        setPriority(900);
    }

    // 覆盖这个方法返回一个注解类，使得只有注解了该annotation的方法才会被起作用(注解在控制器类或方法上均有效)
    // 还有一个相反功能的方法：getDenyAnnotationClass，表示注解了某个annotatioin后，拦截器不要拦截他
    @Override
    protected Class<? extends Annotation> getRequiredAnnotationClass() {
        return LoginRequired.class;
    }

    @Override
    protected Object before(Invocation inv) throws Exception {
        User loginUser = inv.getRequest().getAttribute("loginUser");
        // 如果当前没有登录就返回"r:/lib/login"表示重定向到http://host:port/lib/login页面
        if (loginUser == null) {
            // 没有返回true或null，表示要中断整个处理流程，即不再继续调用其他拦截器以及最终的控制器
            return "r:/lib/login";
        }
        // 返回true或null，表示继续整个流程
        return true;
    }
}
```

LoginRequired.java

```
package com.company.lib.controllers;

// 这是一个annotation，所谓annotation就是一个“标签”，他的职责是“表明”
// 至于表明之后该怎么样？则由其他代码来处理
// 对于LoginRequired具体的处理代码是LoginRequiredInterceptor.java
@Inherited
@Target( { ElementType.TYPE, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface LoginRequired {

}
```

数据库表设计

user表

```
+-------------+--------+--------------+------+-----+-------------------+-----------------------------+
| Field       | Desc   | Type         | Null | Key | Default           | Extra                       |
+-------------+--------+--------------+------+-----+-------------------+-----------------------------+
| id          | 主健   | bigint(20)   | NO   | PRI | NULL              | auto_increment              |
| login_name  | 登录名 | varchar(255) | NO   |     | NULL              |                             |
| password    | 密码   | varchar(255) | NO   |     | NULL              |                             |
| name        | 姓名   | varchar(255) | NO   |     | NULL              |                             |
| create_time |创建时间| timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-------------+--------+--------------+------+-----+-------------------+-----------------------------+
```

```
create table user (
    id bigint not null primary key auto_increment,
    login_name varchar(255) not null,
    password varchar(255) not null,
    name varchar(255) not null,
    create_time timestamp not null
) character set utf8;
```

book表

```
+-------------+--------+--------------+------+-----+-------------------+-----------------------------+
| Field       | Desc   | Type         | Null | Key | Default           | Extra                       |
+-------------+--------+--------------+------+-----+-------------------+-----------------------------+
| id          | 主键   | bigint(20)   | NO   | PRI | NULL              | auto_increment              |
| name        | 书名   | varchar(255) | NO   |     | NULL              |                             |
| price       | 价格   | float        | NO   |     | NULL              |                             |
| author      | 作者   | varchar(255) | NO   |     | NULL              |                             |
| create_time |创建时间| timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-------------+--------+--------------+------+-----+-------------------+-----------------------------+
```

```
create table book(
    id bigint not null primary key auto_increment,
    name varchar(255) not null,
    price varchar(255) not null,
    author varchar(255) not null,
    create_time timestamp not null
) character set utf8;
```

remark表

```
+-------------+--------+---------------+------+-----+-------------------+-----------------------------+
| Field       | Desc   | Type          | Null | Key | Default           | Extra                       |
+-------------+--------+---------------+------+-----+-------------------+-----------------------------+
| id          | 主键   | bigint(20)    | NO   | PRI | NULL              | auto_increment              |
| user_name   | 操作者 | varchar(255)  | NO   |     | NULL              |                             |
| book_id     | 书ID   | bigint(20)    | YES  |     | NULL              |                             |
| essay       | 内容   | varchar(2000) | NO   |     | NULL              |                             |
| create_time |创建时间| timestamp     | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+-------------+--------+---------------+------+-----+-------------------+-----------------------------+
```

```
create table remark(
    id bigint not null primary key auto_increment,
    user_name varchar(255) not null,
    book_id  bigint,
    essay varchar(2000) not null,
    create_time timestamp not null
) character set utf8;
```

oper_log表

```
+------------------+--------+--------------+------+-----+-------------------+-----------------------------+
| Field            | Desc   | Type         | Null | Key | Default           | Extra                       |
+------------------+--------+--------------+------+-----+-------------------+-----------------------------+
| id               | 主健   | bigint(20)   | NO   | PRI | NULL              | auto_increment              |
| user_name        | 操作者 | varchar(255) | NO   |     | NULL              |                             |
| resource_pattern |资源模式| varchar(255) | NO   |     | NULL              |                             |
| resource_id      |资源地址| varchar(255) | NO   |     | NULL              |                             |
| success          |成功标识| tinyint(1)   | NO   |     | NULL              |                             |
| remarks          |额外说明| varchar(255) | NO   |     | NULL              |                             |
| create_time      |创建时间| timestamp    | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+------------------+--------+--------------+------+-----+-------------------+-----------------------------+
```

```
create table oper_log(
    id bigint not null primary key auto_increment,
    user_name varchar(255) not null,
    resource_pattern varchar(255) not null,
    resource_id varchar(255) not null,
    success tinyint(1)  not null,
    remarks varchar(255) not null,
    create_time timestamp not null
) character set utf8;
```

DAO设计

BookDAO.java

```
// dao类必须在xxx.dao或子包下
package com.company.lib.dao;

// 必须是接口，并且以大写DAO结尾
// 必须标注@DAO，DAO中有一个catalog属性，对于大部分人来说，这个都是没用的
@DAO
public interface BookDAO {

    @SQL("select id, name, price, author from book where id = :1")
    public Book get(long bookId);

    @SQL("select id, name, price, author from book limit :1 order by id desc ")
    public List<Book>find(int limit);

    @SQL("select id, name, price, author from book where id < :1 limit :2 order by id desc ")
    public List<Book> find(long bookId, int limit);

    @SQL("update book set name=:1.name, price=:1.price, author=:1.author where id=:1.id")
    public void update(Book book);

    @SQL("insert into book (name, price, author) values (:1.name, :1.price, :1.author)")
    public Identity save(Book book)

}
```

RemarkDAO.java

```
package com.company.lib.dao;


@DAO
public interface RemarkDAO {

    // 标注一个@SQL，写入你的sql语句
    // 不能写select * from remark，这样的后果可能会因为数据库增加了一个字段，但Remark没有相应字段的属性，Jade将抛出异常
    // 参数以冒号开始，:1表示第一个参数
    @SQL("select id, user_name, book_id, essay, create_time from remark where book_id=:1")
    public List<Remark> findByBook(long bookId);

    @SQL("delete from remark where book_id=:1")
    public void deleteByBook(long bookId);

    // 返回int表示变更的条数，就这个示例而言，应该就是返回1
    @SQL("delete from remark where id=:1")
    public int delete(long remarkId);

    // 返回Idenity用于获取自增生成的那个id
    // :1.userName表示第一个参数的userName属性
    @SQL("insert into remark (user_name, book_id, essay) values (:1.userName, :1.bookId, :1.essay")
    public Identity save(Remark remark);
}
```

UserDAO.java

```
package com.company.lib.dao;

@DAO
public interface UserDAO {

    @SQL("select id, name, login_name from user where login_name=:1")
    public User getByLoginName(String loginName);

    @SQL("select id, name, password, login_name, create_time from user where id=:1")
    public User get(long userId);

    @SQL("select id, name, login_name, create_time from user")
    public List<User> find();

    @SQL("delete from user where id=:1")
    public void delete(long userId);

}
```

OperLogDAO.java

```
package com.company.lib.dao;

@DAO
public interface OperLogDAO {

    @SQL("select id, user_name, resource_pattern, resource_id, success, remarks, create_time from oper_log")
    public List<OperLog> find();

    @SQL("insert into oper_log (user_name, resource_pattern, resource_id, success, remarks) values (:1.userName, :1.r
    public void save(OperLog operLog);

}
```

更详细的，可转看Jade DAO编写规范

资源的提供

数据库配置 WEB-INF/applicationContext-dataSource.xml

```
<?xml version="1.0" encoding="utf-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-be
<beans>
    <!-- 这里使用Spring自带的DriverManagerDataSource，实际开发产品应该使用具有连接持管理的DataSource等 -->
    <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/app_lib?useUnicode=true&amp;characterEncoding=utf8"/>
        <property name="username" value="root"/>
        <property name="password" value=""/>
    </bean>

</beans>
```

执行器配置 WEB-INF/applicationContext-executor.xml

由AutoLogInterceptor调用，用于具体执行将操作日志写入到数据

```
<?xml version="1.0" encoding="utf-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-be
<beans>
    <bean class="org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor" destroy="shutdown"/>
</beans>
```

资源的使用

此处将以BookController以及AutoLogInterceptor作为示例，演示资源使用相关的两个问题：

- 如何声明并注入
  - 通过@Autowired注入；
  - DAO接口不用写实现类即可使用
- 如何调用
  - 和调用普通类方法一样

book.BookController.java

```
package com.company.lib.controllers.book;

@LoginRequired
@Path("")
public class BookController {

    // 推荐使用bookDAO作为字段名，但这不是必须的，如果要以其它名称作为名字也不需要另外的配置
    // 如果使用多个DAO，则需要写多个@Autowired在每个DAO声明前
    @Autowired
    private BookDAO bookDAO;

    // 2)
    @Get
    public String list(Invocation inv, @Param("byBookId") long byBookId) {
        int limit = 30;
        List<Book> books = (byBookId <= 0) ? bookDAO.find(limit) : bookDAO.find(byBookId, limit);
        inv.addModel("books", books);
        return "@book-list";
    }

    // 3)
    @Post
    public String add(Book book) {
        bookDAO.save(book);
        return "r:/lib/book";
    }

    // 4)
    @Get("add")
    public String showAdd() {
        return "@book-add";
    }

    // 5)
    @Get("{bookId}")
    public String show(Invocation inv, @Param("bookId") long bookId, @Param("edit") boolean isEdit) {
        Book book = bookDAO.get(bookId);
        inv.addModel("book", book);
        if (isEdit) {
            return "@book-edit";
        }
        return "@book-page";
    }

    // 6)
    @Put("{bookId}")
    public String update(@Param("bookId") long bookId, Book book) {
        book.setId(bookId);
        bookDAO.update(book);
        return "r:/lib/book/" + bookId;
    }

}
```

AutoLogInterceptor.java

```
package com.company.lib.controllers;

public AutoLogInterceptor extends ControllerInterceptorAdapter {

    // 这里的executorService，在applicationContext-executor.xml中并没有id="xxx", 但这没有问题，Spring会给该Bean一个默认的id（类
    @Autowired
    private ExecutorService executorService;

    public void afterCompletion(Invocation inv, Throwable ex) throws Exception  {
        final OperLog operLog = new OperLog();
        operLog.setResourcePattern(inv.getResourceId());
        operLog.setResourceId(inv.getRequestPath().getUri());
        operLog.setSuccess(ex == null);
        operLog.setRemarks(ex == null ? null : ex.getMessage());

        User loginUser = (User) inv.getRequest().getAttribute("loginUser");
        if (loginUser != null) {
```

```
            operLog.setUserName(loginUser.getName());
        }

        // 封装为一个任务
        Runnable task = new Runnable() {
            public void run() {
                operLogDAO.save(operLog);
            }
        };

        // 将插入到数据库的操作提交executorService做异步更新
        // 在实际场景中，这种方式要注意webapp shutdown的时候，还未执行的Task的处理问题
        executorService.submit(task);
    }
}
```

视图组件的设计

- 你需要在webapps/lib下创建views目录,也就是views目录要和WEB-INF同级别，这个不能错了，否则会404;
- controllers下的控制器对应的视图仓库是views，controllers.book、controllers.user、controllers.logs对应的视图仓库是views/book、views/user、views/logs，即与他们向对于controlles的路径一样;
- 您可以使用jsp、vm，就像 第一支程序 中陈述那样，无论食用jsp还是vm，控制器返回都不需要带后缀，rose会判断实际views目录中存在的是jsp或vm等(极端地，如果2个文件都存在，则以字母顺序优先，即jsp优先于vm)(rose会做缓存的，而非总是通过转动磁盘来判断)
- 具体的jsp和vm如何写，是否需要在本页面中作示范？[您可以在评论中发表意见或直接发邮件给我qieqie.wang # gmail.com]
- 因为我们使用了PUT、DELETE，注意相关的请求，在页面上需要写一个hidden字段，参考 《URI设计》表格下面的 **严重说明 :)**

Comment by lonre...@gmail.com, Jul 9, 2010

太好了，真是效率...

Comment by project member qieqie.wang, Jul 9, 2010

to lonrevip: 3q

Comment by skd...@gmail.com, Jul 10, 2010

两个XML文件，都少了：

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

这行。我启动时报错。

Comment by project member qieqie.wang, Jul 10, 2010

to skdlwf: 已加入 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

Comment by skd...@gmail.com, Jul 10, 2010

applicationContext-dataSource.xml 文件中：

```
<property name="user" value="root"/>
```

应为

```
<property name="username" value="root"/>
```

Comment by project member qieqie.wang, Jul 10, 2010

to skdlwf: 已改为name="username"

Comment by zhangzho...@gmail.com, Sep 4, 2010

```
1.BookDAO中
  public Book find(int limit);
  public Book find(long bookId, int limit);
返回类型应该为 List <Book>

2.另外可加上jsp和vm如何写吗？
```

Comment by project member qieqie.wang, Sep 5, 2010

to zhangzhongmac:

1、已改过来； 2、第二个问题我没有理解；

Comment by zhangzho...@gmail.com, Sep 7, 2010

具体的jsp和vm如何写，是否需要在本页面中作示范？[您可以在评论中发表意见或直接发邮件给我qieqie.wang # gmail.com] 可不可以辛苦下，呵呵，具体页面怎么写作个示范，欣赏一下。

Comment by project member qieqie.wang, Sep 7, 2010

to zhangzhongmac:

OK. 我本周末做一个下载，这样什么都有了。

写jsp/vm按照以下步骤完成即可： 1、在{webapphome}下创建views目录(views目录将和WEB-INF同级别) 2、在views目录下创建jsp或vm文件，假设是index.jsp 3、那么在controllers中的控制器方法中只要return "index";不要带.jsp后缀即可使用该页面； 4、如果你的控制器不是controllers根下，而是controllers.xyz.abc类似的package下，则对应的视图目录将是views/xyz/abc，即views目录应和controllers目录是一样的结构，这样controllers.xyz.abc的控制器方法return "hello"，rose将从views/xyz/abc找以hello开头的第一个文件作为视图文件。

Comment by zhangzho...@gmail.com, Sep 8, 2010

to qieqie.wang:

Thanks very much in advance!

Comment by ninja....@gmail.com, Sep 23, 2010

只有controller层和dao层么？ service层怎么写？

Comment by xuy...@gmail.com, Nov 11, 2010

AutuLogInterceptor?里会报Throwable这个bean未定义

Comment by bjh...@gmail.com, Nov 24, 2010

<?xml version="1.0" encoding="utf-8"?> <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd" default-lazy-init="true"> <beans>

<!-- 这里使用Spring自带的DriverManagerDataSource?，实际开发产品应该使用具有连接持管理的DataSource?等 --> <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">

<property name="driverClassName" value="com.mysql.jdbc.Driver"/> <property name="url" value="jdbc:mysql://localhost:3306/app_lib?useUnicode=true&characterEncoding=utf8"/> <property name="username" value="root"/> <property name="password" value=""/>

</bean>

</beans>

不应该有第二个<beans>标签吧...

Comment by mr.mang...@gmail.com, Dec 18, 2010

这位大哥 可以直接写好一个DEMO 让我们下载跑跑看

Comment by haung...@gmail.com, Dec 21, 2010

执行器配置 WEB-INF/applicationContext-executor.xml

<bean class="org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor" destroy="shutdown"/>

中的destroy="shutdown"应该是destroy-method吧？

Comment by hnxi...@gmail.com, Jan 11, 2011

能不能提供一个第二支程序的demo下载呢，这样能比较直观的把它跑起来，再学习就容易些了

Comment by pkon...@gmail.com, Mar 27, 2011

连接数据库那一部分可以使用hibernate来实现么？如果可以的话，spring如何管理啊！

Comment by baal...@gmail.com, Apr 7, 2011

请问一下人人网的专家及各位网友，paoding-rose图片的上传下载，如何实现?

Comment by jiansheliu@gmail.com, Apr 26, 2011

sql中的limit应用放到最后. 登陆时的loginUser应该放在session中吧.

Comment by yutianl1...@gmail.com, May 16, 2011

看了半天，还是不太明白rose到底是用来干嘛的？能给我带来什么好处？

Comment by zhangsui...@gmail.com, Jul 31, 2011

我想问问，views能不能不放在{webapphome}下呢？这样用户可以浏览到。是否可以设置让views目录放在WEB-INF下？谢谢

Comment by zhuyoul...@gmail.com, Aug 28, 2011

lib项目中方法的返回值： return "@book-list"; 是什么意思？好像不是页面啊，在哪里定义呢

Comment by dongqian...@gmail.com, Sep 26, 2011

SQLInterpreter的用法是怎样的啊!

Comment by dongqian...@gmail.com, Sep 26, 2011

怎样将DAO生成的sql语句打印出来呢?

Comment by anne.mia...@gmail.com, Dec 7, 2011

表单提交到controller怎么处理呀？如何将页面的request映射到controller中的对象呀？

```
@Post
    public String add(Book book) {
        bookDAO.save(book); return "r:/lib/book";
```

}

求解？谁能提供下页面的例子呀？

Comment by Camry.Chen88, Mar 1, 2012

求楼主给例子

Comment by yuxin.ya...@gmail.com, May 4, 2012

求楼主给例子

Comment by liujianc...@gmail.com, Aug 10, 2012

楼主有完整的demo下载么，我想学了直接用到我们的项目开发中，如果能有demo真是万分感谢。

Comment by Think...@gmail.com, Sep 12, 2012

好多错误，希望楼主快点改正！

```
package cn.phoinix.controllers;

import java.util.concurrent.ExecutorService?;

import net.paoding.rose.web.ControllerInterceptorAdapter?; import net.paoding.rose.web.Invocation;

import org.springframework.beans.factory.annotation.Autowired;

import cn.phoinix.dao.OperLogDAO; import cn.phoinix.model.OperLog?; import cn.phoinix.model.User;

public class AutoLogInterceptor? extends ControllerInterceptorAdapter? {

    // 这里的executorService，在applicationContext-executor.xml中并没有id="xxx", 但这没有问题，Spring会给该Bean一个默认的id（类全名）
    @Autowired private ExecutorService? executorService;

    @Autowired

    private OperLogDAO operLogDAO;

    public void afterCompletion(Invocation inv, Throwable ex) throws Exception {

        final OperLog? operLog = new OperLog?(); operLog.setResourcePattern(inv.getRequestPath().getUri());//此处已改正
        System.out.println("setResourcePattern: "+inv.getRequestPath().getUri()); operLog.setResourceId(inv.getResourceId());//此处已改正
        System.out.println("setResourceId: "+inv.getResourceId()); operLog.setSuccess(ex == null); operLog.setRemarks(ex == null ? null :
        ex.getMessage());

    User loginUser = (User) inv.getRequest().getSession().getAttribute("user");//此处已改正 if (loginUser != null) {

        operLog.setUserName(loginUser.getNickname());

    }

    // 封装为一个任务 Runnable task = new Runnable() {

        public void run() {

            operLogDAO.save(operLog);

        }

    };

    // 将插入到数据库的操作提交executorService做异步更新 // 在实际场景中，这种方式要注意webapp shutdown的时候，还未执行的Task的处理
    问题 executorService.submit(task);

    }
```

```
}
```

Comment by mycxsky@gmail.com, Nov 6, 2012

java.lang.NullPointerException?

```
net.paoding.rose.jade.core.RowMapperFactoryImpl?.getRowMapper(RowMapperFactoryImpl?.java:92)
net.paoding.rose.jade.core.JdbcOperationFactoryImpl?.getJdbcOperation(JdbcOperationFactoryImpl?.java:70)
net.paoding.rose.jade.core.JadeDaoInvocationHandler?.invoke(JadeDaoInvocationHandler?.java:85) $Proxy17.findUser(Unknown Source)
com.yunjiao.elearning.controllers.IndexController?.index(IndexController?.java:20)
```

请教，问什么controller调用DAO会抛出错误？

```
@Path("{sid:\\d+}") public class IndexController? {

    @Autowired private UserDAO userDAO;

    @Get public String index(@Param("sid") String sourceId) {

        System.out.println(sourceId + "============");

    System.out.println("==============------" + userDAO.findUser()); return "@hello";

    }

}
```

Comment by szr6991...@gmail.com, Nov 21, 2012

```
    @Post

        public String add(Book book) {

            // TODO: insert into database if (success) {

                return "r:/lib/book";

            } else {

                return "@book-add";

            }

        }
```

book中的属性create_time ,通过页面提交不能设置到book中，是什么原因呢？ 日期格式为create_time=2012-12-13

Comment by xsg2...@gmail.com, Jan 21, 2013

你好 我想知道 paoding rose 如何写单元测试，能否使用spring 的单元测试类 进行单元测试

Comment by chrnc0...@gmail.com, May 22, 2013

和grails 相差的也太远了

Comment by nod...@gmail.com, Jun 4, 2013

这边的版本已经不维护了哟，都转移到 github 去了。

Comment by zzti...@gmail.com, Jun 6, 2013

有个字是错的 “无论食用jsp还是vm”

► Sign in to add a comment