### [用maven整合struts+spring+hibernate之一 用maven创建基础的web应用](http://ll-feng.iteye.com/blog/220675)

前言   
前期参与了一个项目,采用了maven+appfuse来融合相关主流技术框架.个人感觉maven是个好东东，很实用，但appfuse自动化成度太高，融合的东西也太多。做大型项目还可以，中小项目就显行太臃肿了。故萌生了给它瘦身的想法。也就是基本框架采用maven+ssh(struts、spring、Hibernate).在这个骨架上，再来按需添加一些技术框架。经过一翻折腾，终于弄出来一个基本的开发框架。现在将框架的配置流程整理出来。   
  
因为是基本骨架，我只需要它能跑起来，技术细节就略了。   
  
用maven作项目管理工具并集成struts2、spring2、hibernate3   
对象持久化采用annotation的方式

### 创建web工程

### mvn archetype:create -DgroupId=net.apex.ssh -DartifactId=sshExt -DarchetypeArtifactId=maven-archetype-webapp

### 这里创建的web工程的目录结构还不完整，需要手动补足test及其下的java resources filters等及main下的java目录 其实这一步还有更简单的方式，就是直接创建struts2工程：

mvn archetype:create -DgroupId=net.apex.ssh -DartifactId=sshExt -DarchetypeGroupId=org.apache.struts -DarchetypeArtifactId=struts2-archetype-starter -DarchetypeVersion=2.0.9-SNAPSHOT -DremoteRepositories=http://people.apache.org/repo/m2-snapshot-repository

这样就基本自动完成了一个基于struts2+spring的web框架。如果自己再加上Hibernate的配置就算一个基础的ssh框架了。不过这个"archetypeVersion"是会经常换的最新的应是2.0.11.2-SNAPSHOT了   
  
二、将web工程加入到eclipse的工作空间

mvn -Declipse.workspace=d:\projects eclipse:add-maven-repo

如果在以前创建过类似的maven工程，这一步大概就不需要了。   
  
三、将web发布到tomcat   
tocmat的conf\catalian\localhost下创建一个xml文件，如sshExt.xml,内容:

<Context path="/sshExt" docBase="D:\lvlf\projects\sshExt\target\sshExt" debug="0" privileged="true" reloadable="false" workDir="D:\lvlf\projects\sshExt\target\sshExt-1.0-SNAPSHOT\tmp"></Context>

四、测试工程   
到这里一个空空的用maven构建的web工程就算告一段落了。第一步要走踏实了才能进行下一步，所以先测试一下。   
执行：   
mvn package   
这个过程第一次会下载一些依赖包。由于我们前期已配置了tomcat，所以命令完成后，就可以启动tomcat，如果没有错误在浏览器里输入：http://localhost:8080/sshExt/index.jsp   
好，OK了

### [用maven整合struts+spring+hibernate之二 数据库生成和测试数据](http://ll-feng.iteye.com/blog/221252)

这一步要实现的目标是在执行mvn package时，maven为我们自动创建数据表，并将测试数据添加进去。是不是很实用？你要觉得不所谓，也要以跳过这步。   
  
一、加入ssh支持   
就是增加struts、spring、hibernate等的依赖包啦。。不细说了。后面详细说明。   
  
二、配置插件   
这一块是最复杂的，先来了解两个插件：   
  
1、hibernate3-maven-plugin插件可实现自动生成数据库schema   
hibernate3:hbm2cfgxml: Generates hibernate.cfg.xml   
hibernate3:hbm2ddl: Generates database schema.   
hibernate3:hbm2doc: Generates HTML documentation for the database schema.   
hibernate3:hbm2hbmxml: Generates a set of hbm.xml files   
hibernate3:hbm2java: Generates Java classes from set of \*.hbm.xml files   
hibernate3:schema-export: Creates SQL DDL file and generates the database schema from set of \*.hbm.xml files   
hibernate3:schema-update: Updates the database schema based on the set of \*.hbm.xml files   
  
2、dbunit-maven-plugin,可以实现数据库中数据的导入导出   
dbunit:operation: Execute a database operation using an external dataset file.   
dbunit:export: Export database tables into a dataset file.   
dbunit:compare: Compare a dataset with database.   
  
了解上面两个插件后，就可以用到它们了。怎么用，就是在pom.xml加入以下代码：

**<build>**

**<finalName>sshExt</finalName>**

**<!-- hibernate3-maven-plugin负责生成数据库 -->**

**<plugins>**

**<plugin>**

**<groupId>org.codehaus.mojo</groupId>**

**<artifactId>hibernate3-maven-plugin</artifactId>**

**<version>2.1</version>**

**<dependencies>**

**<dependency>**

**<groupId>mysql</groupId>**

**<artifactId>mysql-connector-java</artifactId>**

**<version>5.0.5</version>**

**</dependency>**

**</dependencies>**

**<configuration>**

**<components>**

**<component>**

**<name>hbm2ddl</name>**

**<implementation>annotationconfiguration</implementation>**

**</component>**

**</components>**

**<componentProperties>**

**<drop>true</drop>**

**<jdk5>true</jdk5>**

**<propertyfile>target/classes/jdbc.properties</propertyfile>**

**</componentProperties>**

**<skip>${maven.test.skip}</skip><!--很重要，可以控制是否要在package时重新建表-->**

**</configuration>**

**<executions>**

**<execution>**

**<phase>process-test-resources</phase>**

**<goals>**

**<goal>hbm2ddl</goal>**

**</goals>**

**</execution>**

**</executions>**

**</plugin>**

**<!-- dbunit-maven-plugin负责数据库的导入导出 -->**

**<plugin>**

**<groupId>org.codehaus.mojo</groupId>**

**<artifactId>dbunit-maven-plugin</artifactId>**

**<version>1.0-beta-1</version>**

**<executions>**

**<execution>**

**<id>test-compile</id>**

**<phase>test-compile</phase>**

**<goals>**

**<goal>operation</goal>**

**</goals>**

**</execution>**

**<execution>**

**<id>test</id>**

**<phase>test</phase>**

**<goals>**

**<goal>operation</goal>**

**</goals>**

**</execution>**

**</executions>**

**<dependencies>**

**<dependency>**

**<groupId>${jdbc.groupId}</groupId>**

**<artifactId>${jdbc.artifactId}</artifactId>**

**<version>${jdbc.version}</version>**

**</dependency>**

**</dependencies>**

**<configuration>**

**<dataTypeFactoryName>${dbunit.dataTypeFactoryName}</dataTypeFactoryName>**

**<driver>${jdbc.driverClassName}</driver>**

**<username>${jdbc.username}</username>**

**<password>${jdbc.password}</password>**

**<url>${jdbc.url}</url>**

**<src>src/test/resources/sample-data.xml</src>**

**<type>${dbunit.operation.type}</type>**

**<skip>${maven.test.skip}</skip><!--很重要，可以控制是否要在package时插入测试数据-->**

**</configuration>**

**</plugin>**

**</plugins>**

**</build>**

三、配置数据库连接   
除了pom.xml文件。还需一个jdbc.properties文件。内容如下：

**hibernate.dialect=org.hibernate.dialect.MySQLInnoDBDialect**

**hibernate.connection.username=root**

**hibernate.connection.password=**

**hibernate.connection.url=jdbc:mysql://localhost/test?createDatabaseIfNotExist=true&amp;useUnicode=true&amp;characterEncoding=utf-8**

**hibernate.connection.driver\_class=com.mysql.jdbc.Driver**

四、配置实体，由于我们采用hibernate框架   
hibernate.cfg.xml：

**<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN"**

**"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">**

**<hibernate-configuration>**

**<session-factory>**

**<mapping class="net.apex.framework.model.User"/>**

**<!-- <mapping class="net.apex.framework.model.Role"/> -->**

**</session-factory>**

**</hibernate-configuration>**

五、配置持久化方案   
persistence.xml:

**<persistence xmlns="http://java.sun.com/xml/ns/persistence"**

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

**xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence\_1\_0.xsd"**

**version="1.0">**

**<persistence-unit name="ApplicationEntityManager" transaction-type="RESOURCE\_LOCAL">**

**<provider>org.hibernate.ejb.HibernatePersistence</provider>**

**<!--**

**<properties>**

**<property name="hibernate.hbm2ddl.auto" value="create-drop"/>**

**</properties>**

**-->**

**</persistence-unit>**

**</persistence>**

这个文件基本就这样。   
  
六、添加实体对象   
User.java:

**/\*\***

**\***

**\*/**

**package net.apex.framework.model;**

**import javax.persistence.Column;**

**import javax.persistence.Entity;**

**import javax.persistence.GeneratedValue;**

**import javax.persistence.Id;**

**/\*\***

**\* @author Administrator**

**\***

**\*/**

**@Entity**

**public class User {**

**private Long id;**

**private String name;**

**/\*\***

**\* @return the id**

**\*/**

**@Id**

**@GeneratedValue**

**public Long getId() {**

**return id;**

**}**

**/\*\***

**\* @param id the id to set**

**\*/**

**public void setId(Long id) {**

**this.id = id;**

**}**

**/\*\***

**\* @return the name**

**\*/**

**@Column(name="name")**

**public String getName() {**

**return name;**

**}**

**/\*\***

**\* @param name the name to set**

**\*/**

**public void setName(String name) {**

**this.name = name;**

**}**

**}**

这个实体对象采用annotaion来做持久化映射   
  
七、添加测试数据   
sample-data.xml:

**<?xml version="1.0" encoding="UTF-8"?>**

**<dataset>**

**<table name="user">**

**<column>id</column>**

**<column>name</column>**

**<row>**

**<value description="id">-1</value>**

**<value description="name">user</value>**

**</row>**

**<row>**

**<value description="id">-2</value>**

**<value description="name">admin</value>**

**</row>**

**</table>**

**</dataset>**

以上配置基本实现了。在这个maven工程启动时，会自动创建数据库，并将sample-data.xml中的测试数据插入到数据库中。那么如果数据库通过其它来源已插入了一些不错的数据，怎么将它导入到sample-data.xml中呢。有办法：

**mvn dbunit:export -Ddest=sample-data.xml**

上面的几个文件的目录是这样的：   
src/main/resources/hibernate.cfg.xml   
src/main/resources/jdbc.properties   
src/main/resources/META-INF/persistence.xml   
src/test/resources/sample-data.xml   
pom.xml

[**用maven整合struts+spring+hibernate之三 加入框架支持**](http://ll-feng.iteye.com/blog/221167)

创建完web工程之后，一般我们就可以开始开发了。所以，这一节我们需要为这个web工程加入j2ee框架的支持。   
  
这里其实不太好分清楚先加哪后加哪，因为系统框架是需要整体运行的，相互的关系很紧密，比如我只加入struts2.那么在运行时会报错。因为struts2缺省是用spring来做容器的。（当然也可以修改，但这里不是不作讨论）。

一、依赖包

Java代码

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-core</artifactId>

<version>2.0.9</version>

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-codebehind-plugin</artifactId>

<version>2.0.9</version>

</dependency>

<dependency>

<groupId>org.apache.struts</groupId>

<artifactId>struts2-spring-plugin</artifactId>

<version>2.0.9</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring</artifactId>

<version>2.5</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-hibernate3</artifactId>

<version>2.0.6</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate</artifactId>

<version>3.2.5.ga</version>

</dependency>

<dependency>

<groupId>javax.persistence</groupId>

<artifactId>persistence-api</artifactId>

<version>1.0</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.0.5</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-annotations</artifactId>

<version>3.3.0.ga</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-commons-annotations</artifactId>

<version>3.3.0.ga</version>

</dependency>

以上不止加入struts支持，也加入了spring和hibernate及jpa的支持。其实这些只是主包，还有不少依赖包会自动加入到maven的仓库里去的。   
  
二、配置struts   
下面配置struts2,涉及到几个文件   
1、在web.xml加入以下内容：

<filter>

<filter-name>struts2</filter-name>

<filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>

</filter>

<filter-mapping>

<filter-name>struts2</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

2、增加struts.xml文件   
这个文件主要配置struts的action及拦截器等，可以有多个（命名为struts-\*.xml)   
位置在src\main\resources下，内容大致如下：

**<?xml version="1.0" encoding="UTF-8"?>**

**<!DOCTYPE struts PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"**

**"http://struts.apache.org/dtds/struts-2.0.dtd">**

**<struts>**

**<!-- Constants -->**

**<constant name="struts.devMode" value="false"/>**

**<constant name="struts.i18n.encoding" value="UTF-8"/>**

**<constant name="struts.action.extension" value="html"/>**

**<constant name="struts.objectFactory" value="spring"/>**

**<constant name="struts.custom.i18n.resources" value="ApplicationResources,errors"/>**

**<constant name="struts.multipart.maxSize" value="2097152"/>**

**<constant name="struts.ui.theme" value="css\_xhtml"/>**

**<constant name="struts.codebehind.pathPrefix" value="/WEB-INF/pages/"/>**

**<constant name="struts.enable.SlashesInActionNames" value="true"/>**

**<!-- Configuration for the default package. -->**

**<package name="default" extends="struts-default">**

**<interceptors>**

**<!-- Copied from struts-default.xml and changed validation exclude methods -->**

**<interceptor-stack name="defaultStack">**

**<interceptor-ref name="exception"/>**

**<interceptor-ref name="alias"/>**

**<interceptor-ref name="servlet-config"/>**

**<interceptor-ref name="prepare"/>**

**<interceptor-ref name="i18n"/>**

**<interceptor-ref name="chain"/>**

**<interceptor-ref name="debugging"/>**

**<interceptor-ref name="profiling"/>**

**<interceptor-ref name="scoped-model-driven"/>**

**<interceptor-ref name="model-driven"/>**

**<interceptor-ref name="fileUpload"/>**

**<interceptor-ref name="checkbox"/>**

**<interceptor-ref name="static-params"/>**

**<interceptor-ref name="params">**

**<param name="excludeParams">dojo\..\*</param>**

**</interceptor-ref>**

**<interceptor-ref name="conversionError"/>**

**<interceptor-ref name="validation">**

**<param name="excludeMethods">cancel,execute,delete,edit,list</param>**

**</interceptor-ref>**

**<interceptor-ref name="workflow">**

**<param name="excludeMethods">input,back,cancel,browse</param>**

**</interceptor-ref>**

**</interceptor-stack>**

**<interceptor-stack name="fileUploadStack">**

**<interceptor-ref name="fileUpload"/>**

**<interceptor-ref name="defaultStack"/>**

**</interceptor-stack>**

**</interceptors>**

**<global-results>**

**<result name="mainMenu" type="redirect-action">mainMenu</result>**

**<result name="dataAccessFailure">/WEB-INF/pages/dataAccessFailure.jsp</result>**

**</global-results>**

**<global-exception-mappings>**

**<exception-mapping exception="org.springframework.dao.DataAccessException" result="dataAccessFailure"/>**

**</global-exception-mappings>**

**<action name="addUser" class="userAction" method="save">**

**<result name="success">/WEB-INF/pages/user/list.jsp</result>**

**<result name="input">/WEB-INF/pages/user/form.jsp</result>**

**</action>**

**<action name="toAddUser">**

**<result>/WEB-INF/pages/user/form.jsp</result>**

**</action>**

**<!-- Add additional actions here -->**

**</package>**

**</struts>**

3、添加国际化支持   
也就是加几个资源文件主要是：   
ApplicationResources.properties   
ApplicationResources\_zh\_CN.properties   
这个就不我说了,文件位置在src\main\resources下   
  
4、添加action   
以用户为例UserAction.java:

**package net.apex.framework.web.action;**

**import net.apex.framework.model.User;**

**import com.opensymphony.xwork2.ActionSupport;**

**/\*\***

**\* @author Administrator**

**\***

**\*/**

**public class UserAction extends ActionSupport {**

**private User user;**

**public User getUser() {**

**return user;**

**}**

**public void setUser(User user) {**

**this.user = user;**

**}**

**public String save(){**

**System.out.println(user.getName());**

**return SUCCESS;**

**}**

**}**

struts配置完毕   
  
三、配置spring   
1、web.xml   
加入以下代码：

**<!--配置spring上下文(可选)-->**

**<context-param>**

**<param-name>contextConfigLocation</param-name>**

**<param-value>/WEB-INF/applicationContext\*.xml</param-value>**

**</context-param>**

**<!--加入spring支持-->**

**<listener>**

**<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>**

**</listener>**

2、添加ApplicationContext.xml   
这个也可以有多个，内容大致如下：

**<?xml version="1.0" encoding="UTF-8"?>**

**<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-beans-2.0.xsd**

**http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-2.0.xsd**

**http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-2.0.xsd">**

**<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">**

**<property name="driverClassName" value="com.mysql.jdbc.Driver" />**

**<property name="url" value="jdbc:mysql://localhost/test" />**

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

**<property name="password" value="1234" />**

**</bean>**

**<bean id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">**

**<property name="dataSource" ref="dataSource"/>**

**<property name="configLocation" value="classpath:hibernate.cfg.xml"/>**

**<property name="hibernateProperties">**

**<props>**

**<prop key="hibernate.dialect">${hibernate.dialect}</prop>**

**<prop key="hibernate.cache.use\_second\_level\_cache">false</prop>**

**<prop key="hibernate.cache.provider\_class">org.hibernate.cache.EhCacheProvider</prop>**

**<prop key="show\_sql">true</prop>**

**</props>**

**<!-- Turn batching off for better error messages under PostgreSQL -->**

**<!-- hibernate.jdbc.batch\_size=0 -->**

**</property>**

**</bean>**

**<bean id="userAction" scope="prototype" class="net.apex.framework.web.action.UserAction"></bean>**

**</beans>**

这样struts、spring和hibernate框架都加入到我们的工程里了。   
在工程里添加一个jsp页面让它调用UserAction就可以跑起来了。。。

### [用maven整合struts+spring+hibernate之四 添加事务管理](http://ll-feng.iteye.com/blog/238193)

前面几步已经基本搭起了一个可以开发的框架了。但是还不完整。比如事务管理就没有加上。这里补上。   
由于我们在持久层用hibernate来实现。所有的CRUD都是hibernate的API.所以事务就可以委托给hibernate来做。那么我们需要在spring的配置文件里加上以下代码：

**<!--事务管理bean-->**

**<bean id="transactionManager" class="org.springframework.orm.hibernate3.HibernateTransactionManager">**

**<property name="sessionFactory">**

**<ref bean="sessionFactory"/>**

**</property>**

**</bean>**

**<!-- 配置事务通知 -->**

**<tx:advice id="txAdvice" transaction-manager="transactionManager">**

**<tx:attributes>**

**<tx:method name="save\*" propagation="REQUIRED" read-only="false"/>**

**<tx:method name="get\*" propagation="REQUIRED" read-only="true"/>**

**<tx:method name="\*" read-only="true"/>**

**</tx:attributes>**

**</tx:advice>**

**<!--添加事务-->**

**<aop:config>**

**<!--切入点-->**

**<aop:pointcut id="txPointCut" expression="execution(\* net.apex.ssh.common.service.\*.\*(..))"/>**

**<!--通知器-->**

**<aop:advisor advice-ref="txAdvice" pointcut-ref="txPointCut"/>**

**</aop:config>**

这样，应用的事务就添加完成了。说明一下，spring2的声明式事务管理可以用注解来做，也可以用基于schema的xml配置来做。本例即后一种方式。   
  
那么这样就完了吗？测试一下就知道了。   
可以在service实现类里抛出异常。看看能否能回滚。比如：

**public void saveUser(User user) {**

**userDao.save(user);**

**throw new RuntimeException("测试事务！............");**

**}**

我在调用userDao.saveUser(user)对user对象进行持久化之后，马上手动抛出异常。如果事务配置起作用了，就会进行回滚，也就是之前保存了数据并不会提交，否则数据就会保存到数据库。如果一切正常，应该就配置完成了。   
  
这里要注意，spring的事务管理并不是所有的异常都会导致回滚。实际上默认是运行时异常及它的子类异常才会被回滚。不过这些是可以配置的。可以参考spring的参考文档，对以上配置加以修改。、

作者博客地址：

### http://ll-feng.iteye.com