### 数据库的BIGINT类型对应的就是JAVA里的LONG型数据

### svnkit使用addFile来实现rename\move\copy时的错误

当仓库为FSFS格式本地仓库时，使用addFile来实现svnCopy操作时存在问题，代码如下：

//doCopyFile

**private** **boolean** copyEntry(ISVNEditor editor,String srcParentPath, String srcEntryName, String dstParentPath,String dstEntryName,**boolean** isMove)

{

**try** {

editor.openRoot(-1);

editor.openDir(dstParentPath, -1);

//Copy the file

String dstEntryPath = dstParentPath + dstEntryName;

String srcEntryPath = srcParentPath + srcEntryName;

editor.addFile(dstEntryPath, srcEntryPath, -1);

editor.closeFile(dstEntryPath, "");

//close the parent Dir

editor.closeDir();

**if**(isMove)

{

editor.deleteEntry(srcEntryPath, -1);

}

/\*

\* Closes the root directory.

\*/

editor.closeDir();

} **catch** (SVNException e) {

System.*out*.println("copyFile(): Schedule to copyEntry Failed!");

e.printStackTrace();

**return** **false**;

}

**return** **true**;

}

E160000: Got source path but no source revision for '/rdata/AutoDeploy55'

从源码中可以看出，是检查copyFromRevision时报的错误，但如果我传入latestRevision时，则报checkSum错误，按道理如果是仓库内部不复制的话，不需要穿入sendDelta，因此这个问题比较麻烦，据说Highlevel Interface也是基于LowLevelApI实现的，所以可能是我的使用方法不正确，因此我需要使用HLInterface来try的copy接口，如果能正常工作的话就参考HLAPI的实现方法

**public** **void** addFile(String path, String copyFromPath, **long** copyFromRevision) **throws** SVNException {

DirBaton parentBaton = (DirBaton) myDirsStack.peek();

String fullPath = SVNPathUtil.*getAbsolutePath*(SVNPathUtil.*append*(myBasePath, path));

**if** (copyFromPath != **null** && FSRepository.*isInvalidRevision*(copyFromRevision)) {

SVNErrorMessage err = SVNErrorMessage.*create*(SVNErrorCode.*FS\_GENERAL*, "Got source path but no source revision for ''{0}''", fullPath);

SVNErrorManager.*error*(err, SVNLogType.*FSFS*);

} **else** **if** (copyFromPath != **null**) {

SVNNodeKind kind = myTxnRoot.checkNodeKind(fullPath);

**if** (kind != SVNNodeKind.*NONE* && !parentBaton.isCopied()) {

SVNErrorManager.*error*(FSErrors.*errorOutOfDate*(fullPath, kind), SVNLogType.*FSFS*);

}

copyFromPath = myRepository.getRepositoryPath(copyFromPath);

FSRevisionRoot copyRoot = myFSFS.createRevisionRoot(copyFromRevision);

myCommitter.makeCopy(copyRoot, copyFromPath, fullPath, **true**);

} **else** {

myCommitter.makeFile(fullPath);

}

}

### svnkit接口使用限制

调用svnkit底层addFile或addDir接口时需要保证您要操作的节点的父节点处于open状态，即你必须先调用openDir接口打开其父节点

openRoot(-1);

openDir(parentPath,-1);

addFile(fileName,null,-1);

....

closeFile

closeDir()

closeDir()

### 相等符号 == 的使用问题

String和Integer类型变量相等判断，需要使用equals()，==是指针判断，是否相同要看运气，两个变量指向同一地址时才会相等

### Input file onchang不触发问题

Onchange只有在input发生变化时才会触发，因此下次如果是选择了相同文件会触发不了，需要先置空input

1. <a href=””> 和 window.location.href=””

使用相对路径时，是指当前页面所在的路径，可以../表示上层目录

在使用window.location.href时出现一个问题，就是链接总是不正确，而且还带着表单的参数，经确认应该是使用了form的原因，改为div就不会出现该问题了，原因估计是代码了会自动提交表单的原因。

### 后台是否能够控制跳转到MVC解析器配置目录以外的html文件

MVC解析器配置：

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="prefix" value="/WEB-INF/jsp/" />

<property name="suffix" value=".jsp" /><!--可为空,方便实现自已的依据扩展名来选择视图解释类的逻辑 -->

</bean>

根据上面的配置分析，如果controller函数 return “test”，则解析器会加上前缀和后缀后，找到 WebRoot下的/WEB-INF/jsp/test.jsp文件，并进行解析后返回给前台。

因此，如果要实现跳转到MVC配置目录以外的html文件是做不到的，到可以将prefix配置在WebRoot根目录下，suffix设置为空，这样可以跳转到任何文件。

### insert一条记录如何获取新增的记录的ID

JDBC里约定Insert、Update、Delete的返回值是影响的行数，因此如果要知道新增的记录有以下两个方法：

1. 查询表中最后增加的记录ID

SELECT   IDENT\_CURRENT('TableName')

存在异步的时间差问题，如果在insert和查询中间，存在其他insert操作，会导致查询到的记录信息不正确

1. 根据插入的记录的信息（具有唯一性）查询出对应的记录，从记录中获取ID

要求信息具有唯一性

1. 使用如下的insert语句

<insert id=*"add"* parameterType=*"com.DocSystem.entity.Doc"*

useGeneratedKeys=*"true"* keyColumn=*"id"* keyProperty=*"id"*>

sql语句会将id值设置到传入的参数的对应字段里去

这个方法比较简单，一次性就搞定了

### 数据库字段不能使用保留字符

ADD IN 等在sql语句中的特殊字符，如果用作数据库的字段名称的话，会出现莫名其妙的语法错误（syntax error）

### 数据库查询错误ID is ambiguous

表示字段ID不明确（模棱两可），通常在把两张表left join一起时会发生这种情况，因此需要指定是哪张表的字段，

错误（因为repos\_auth的表里也有同样的ID字段）： select ID, NAME, REPOS\_ID, IS\_ADMIN, ACCESS, EDIT\_EN, ADD\_EN, DELETE\_EN from user LEFT JOIN repos\_auth on repos\_auth.user\_id = user.id WHERE repos\_auth.REPOS\_ID = #{reposId,jdbcType=INTEGER}

正确： select user.ID, NAME, REPOS\_ID, IS\_ADMIN, ACCESS, EDIT\_EN, ADD\_EN, DELETE\_EN from user LEFT JOIN repos\_auth on repos\_auth.user\_id = user.id WHERE repos\_auth.REPOS\_ID = #{reposId,jdbcType=INTEGER}

### 数据库中有数据，但查询出的结果中却只有一部分

错误：select user.ID, NAME, REPOS\_ID, IS\_ADMIN, ACCESS, EDIT\_EN, ADD\_EN, DELETE\_EN

因为使用的是resultType，缺少对带下划线字段的自动映射，因此需要手动加映射。

正确姿势： select user.ID, NAME, REPOS\_ID as reposId, IS\_ADMIN as isAdmin, ACCESS, EDIT\_EN as editEn, ADD\_EN as addEn, DELETE\_EN as deleteEn

### 数据表合并查询案例

查询仓库的所有授权用户：

select user.ID, NAME, REPOS\_ID, IS\_ADMIN, ACCESS, EDIT\_EN, ADD\_EN, DELETE\_EN from user LEFT JOIN repos\_auth on repos\_auth.user\_id = user.id WHERE repos\_auth.REPOS\_ID = #{reposId,jdbcType=INTEGER}

查询仓库的所有用户（有授权信息的带授权信息，没有的留空）：

select user.ID, NAME, REPOS\_ID, IS\_ADMIN, ACCESS, EDIT\_EN, ADD\_EN, DELETE\_EN from user LEFT JOIN repos\_auth on repos\_auth.user\_id = user.id AND repos\_auth.REPOS\_ID = #{reposId,jdbcType=INTEGER}

### 通常出现Address already in use: JVM\_Bind错误的可能性分析

I．就是当前端口已经有别的程序在占用着，所以要么把占用这个端口的程序关闭，要么重新换一个端口  
II．端口号被占用,如果你有装oracle的话,有可能是oracle使用了8080端口，oracle安装后并且如果启动了OracleHttp服务会占用8080端口  
III．我认为很可能是多启动了几次TOMCAT,在ECLIPSE下重复启动TOMCAT就会出现这个问题,你去调查一下看看是否是这个原因.  
IV．如果不是windows操作系统,那么80端口已经被占用.如果是windows操作系统.请检查是否装有IIS.

V.启动了多个Tomcat。

我就是因为启动了两个Tomcat，所以才会报这样的错误，通常情况下， 多次启动Tomcat或者非正常关闭Myeclipse，但是占用端口的进程没有关闭，也会出现这样的错误。解决方法是关闭javaw.exe进程。

### [Mybatis的Dao向mapper传多个参数（三种解决方案）](http://blog.csdn.net/bangrenzhuce/article/details/52574879)

（1）第一种：

Dao层接口：

Public User selectUser(String name,String area);

Mapper.xml：

<select id="selectUser" resultMap="BaseResultMap" parameterType="java.lang.String">

select \* from user\_user\_t where user\_name = #{0} and user\_area=#{1}

</select>

其中，#{0}代表接收的是dao层中的第一个参数，#{1}代表dao层中第二参数，更多参数一致往后加即可。

（2）第二种：采用Map传多参数

Dao层接口：

Public User selectUser(Map paramMap);  、

Mapper.xml：

<select id=" selectUser" resultMap="BaseResultMap">

select \* from user\_user\_t where user\_name = #{userName，jdbcType=VARCHAR} and user\_area=#{userArea,jdbcType=VARCHAR}

</select>

但调用Dao层接口时稍微麻烦一些（你需要把参数放入到Map里）：

Private User xxxSelectUser(){

Map paramMap=new hashMap();

paramMap.put(“userName”,”对应具体的参数值”);

paramMap.put(“userArea”,”对应具体的参数值”);

User user=xxx. selectUser(paramMap);}  

（3）第三种：

Dao层接口：

Public User selectUser(@param(“userName”)Stringname,@param(“userArea”)String area);

Mapper.xml：

<select id=" selectUser" resultMap="BaseResultMap">

select \* from user\_user\_t where user\_name = #{userName，jdbcType=VARCHAR} and user\_area=#{userArea,jdbcType=VARCHAR}

</select>

这种方法能让开发者看到dao层方法就知道该传什么样的参数，比较直观。

### Maven的使用方法