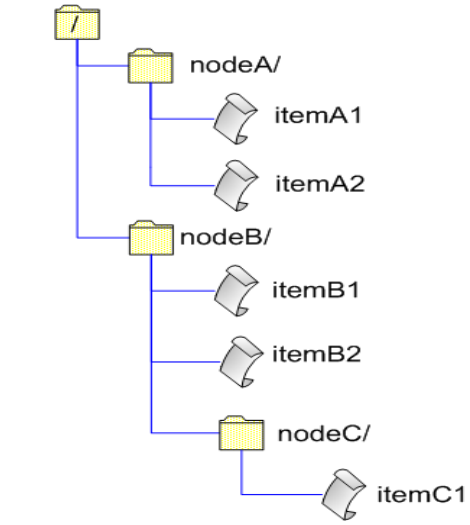
随笔-72 文章-0 评论-2

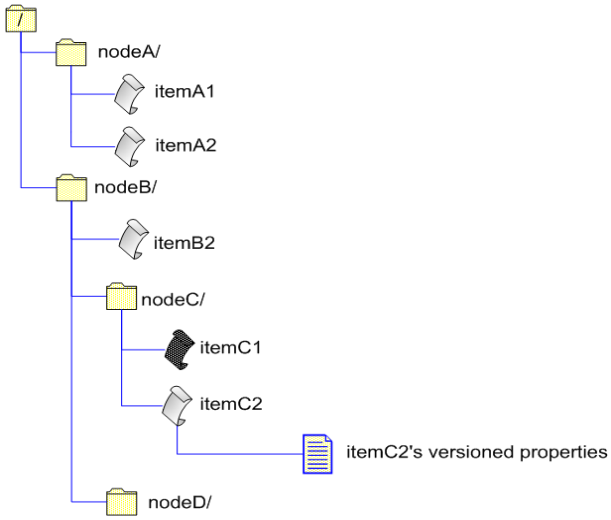
SVNKit学习——使用低级别的API（ISVNEditor接口）直接操作Repository的目录和文件（五）

本文是参考官方文档的实现，官方wiki：
https://wiki.svnkit.com/Committing_To_A_Repository

本文核心使用的是ISVNEditor这个接口直接对Repository进行各种AM操作~

以下两张示例图分别代表我们操作前、操作后仓库的结构：





具体实现：

```
package com.demo;

import com.google.gson.Gson;
import org.tmatesoft.svn.core.*;
import
```

java(20)
javascript(1)
jboss(1)
jdbc
jndi
linux(6)
maven(7)
memcached(4)
mysql(3)
redis(2)
solr(1)
springMVC(6)
Swing(1)
tomcat(1)
版本控制、管理工具(11)
计算机基础(2)
硬件、装机(1)

随笔档案

2017年4月 (6)
2017年3月 (7)
2017年2月 (47)
2017年1月 (4)
2016年12月 (7)
2016年11月 (1)

最新评论

1. Re:SVNKit学习——svn二次开发背景和闲谈（一）
使用svnkit如何解决merge或者commit时候的冲突问题,我是直接合并服务器端的trunk和branch
--rking_java
2. Re:SVNKit学习——使用低级别的API（ISVNEditor接口）直接操作Repository的目录和文件（五）
受益匪浅。最近我们也正在用svnkit做二次开发，刚做了版本变更的通知功能。现在正在找API，能查出目录的用户列表，但一直没找到，请问楼主有啥建议不？在windows服务端VisualSvnServ.....
--tinyhead

阅读排行榜

1. SVN学习——简单入门之创建仓库、导入、检出（一）(5835)
2. 关于程序中使用servlet-api.jar和jsp-api.jar与服务器lib包jar包冲突的问题(2855)

```
org.tmatesoft.svn.core.auth.ISVNAuthenticationManager;
import
org.tmatesoft.svn.core.internal.io.dav.DAVRepositoryFacto
ry;
import org.tmatesoft.svn.core.io.ISVNEditor;
import org.tmatesoft.svn.core.io.SVNRepository;
import org.tmatesoft.svn.core.io.SVNRepositoryFactory;
import org.tmatesoft.svn.core.io.diff.SVNDeltaGenerator;
import org.tmatesoft.svn.core.wc.SVNWCUtil;
import java.io.ByteArrayInputStream;
import java.io.UnsupportedEncodingException;

/**
 * 提交到仓库
 * 这块看官方demno的意思如果不用权限认证，会使用a session user
name作为提交的author，但是我试了会报错401，author required~
 * 本例是基于初始仓库图A转换为目标仓库图B的过程，我们需要执行的操作
有：
 * 1.删除nodeB/itemB1
 * 2.编辑nodeC/itemC1
 * 3.新增nodeC/itemC2，并设置itemC2的文件属性
 * 4.新增nodeB子节点nodeD
 */
public class CommitToRepository {
    public static void main(String[] args) throws
Exception{
        //1.根据访问协议初始化工厂
        DAVRepositoryFactory.setup();;
        //2.初始化仓库，由于我们所有的操作都是基于nodeB节点以下
的，所以我们将nodeB作为本次操作的root节点
        String url = "https://wlyfree-
PC:8443/svn/svnkitRepository2/trunk/nodeB";
        SVNRepository svnRepository =
SVNRepositoryFactory.create(SVNURL.parseURIEncoded(url));
        //3.初始化权限
        String username = "wly";
        String password = "wly";
        char[] pwd = password.toCharArray();
        ISVNAuthenticationManager
isvnAuthenticationManager =
SVNWCUtil.createDefaultAuthenticationManager(username,pwd
);

svnRepository.setAuthenticationManager(isvnAuthentication
Manager);
        //====================================DEMO
START=========================================
        ISVNEditor editor = null;
        long revisionNo = -1; //指定版本号为最新版本
        //4.1.删除nodeB/itemB1
        try{
            //获取编辑器
            editor =
svnRepository.getCommitEditor("delete
file",null,true,null);
            String itemB1Path = "itemB1";//要删除的文件路径
```

3. mysql的sql_mode介绍和修改 (2445)
4. SpringMVC学习（二）——基于xml配置的springMVC项目（maven+spring4）(1473)
5. SpringMVC学习（六）——@InitBinder注解(1253)

评论排行榜

1. SVNKit学习——svn二次开发背景和闲谈（一）(1)
2. SVNKit学习——使用低级别的API（ISVNEditor接口）直接操作Repository的目录和文件（五）(1)

推荐排行榜

1. SVNKit学习——svn二次开发背景和闲谈（一）(1)
2. Swing入门学习(1)
3. SVNKit学习——使用High-Level API管理Working Copy示例(六)(1)
4. SVNKit学习——使用低级别的API（ISVNEditor接口）直接操作Repository的目录和文件（五）(1)
5. git学习——简介、使用（一）(1)

```
            SVNCommitInfo svnCommitInfo =
deleteFile(editor,revisionNo);//执行删除并返回执行结果
            System.out.println("执行删除操作的返回结果：" +
svnCommitInfo);
        }catch (SVNException e){
            //发生异常需要终止操作
            editor.abortEdit();
            e.printStackTrace();;
        }
        //4.2.编辑nodeC/itemC1
        try{
            //获取编辑器
            editor =
svnRepository.getCommitEditor("modify
file",null,true,null);
            SVNCommitInfo svnCommitInfo =
modifyFile(editor,revisionNo);
            System.out.println("执行编辑操作的返回结果：" +
svnCommitInfo);
        }catch(SVNException e){
            //发生异常需要终止操作
            editor.abortEdit();
            e.printStackTrace();;
        }
        //4.3.新增nodeC/itemC2，并设置itemC2的文件属性
        try{
            editor = svnRepository.getCommitEditor("add
file",null,true,null);
            SVNCommitInfo svnCommitInfo =
addFile(editor,revisionNo);
            System.out.println("执行新增文件操作的返回结果："
+ svnCommitInfo);
            //校验nodeC/itemC2的属性是否成功设置进去
            SVNProperties s = new SVNProperties();

svnRepository.getFile("nodeC/itemC2",-1,s,null);
            Gson gson = new Gson();
            System.err.println(gson.toJson(s));
        }catch (SVNException e){
            editor.abortEdit();
            e.printStackTrace();
        }

        //4.4.新增nodeB子节点nodeD
        try{
           editor = svnRepository.getCommitEditor("add
dir",null,true,null);
            SVNCommitInfo svnCommitInfo =
addDir(editor,revisionNo);
            System.out.println("执行新增目录操作的返回结果："
+ svnCommitInfo);
        }catch (SVNException e){
            editor.abortEdit();
            e.printStackTrace();
        }
    }
```

```
    /**
     *  删除文件
     *  @param editor 编辑器
     *  @param revisionNo 修订版版本号
     *  @return  SVNCommitInfo 提交结果信息
     *  @throws SVNException
     */
    private static SVNCommitInfo deleteFile(ISVNEditor
editor,long revisionNo) throws SVNException{
        // 进入Root节点，即nodeB
        editor.openRoot(revisionNo);
        //4.3.删除文件
        editor.deleteEntry("itemB1",revisionNo);
        //操作完成要关闭编辑器，并返回操作结果
        return editor.closeEdit();
    }

    /**
     * 编辑文件
     *  @param editor 编辑器
     *  @param revisionNo 修订版版本号
     *  @return  SVNCommitInfo 提交结果信息
     *  @throws SVNException
     */
    private static SVNCommitInfo modifyFile(ISVNEditor
editor,long revisionNo) throws SVNException{
        // 进入Root节点，即nodeB
        editor.openRoot(revisionNo);
        //.进入nodeC节点
        editor.openDir("nodeC",revisionNo);
        // 编辑nodeC/itemC1的内容
        String itemC1Path = "nodeC/itemC1";//路径都是相对于
root的
        editor.openFile(itemC1Path,revisionNo);
        //确保客户端这个文件的内容和服务端的是一样的，如果不一致的
话是不允许提交的。底层实现使用MD5
        String baseChecksum = null;
        editor.applyTextDelta(itemC1Path,baseChecksum);
        //提交文件变更的数据,windows默认是100kb大小
        byte[] oldData = new byte[]{};
        byte[] newData = null;
        try {
            newData = "我来测试一下编辑2".getBytes("utf-
8");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        ByteArrayInputStream baseData = new
ByteArrayInputStream(oldData);
        ByteArrayInputStream workingData = new
ByteArrayInputStream(newData);
        SVNDeltaGenerator svnDeltaGenerator = new
SVNDeltaGenerator();//100KB-windows generator
        String checksum =
svnDeltaGenerator.sendDelta(itemC1Path,baseData,0,working
```

```
Data,editor,true);
        // 关闭文件
        editor.closeFile(itemC1Path,checksum);
        // 关闭目录nodeC
        editor.closeDir();
        // 关闭根目录nodeB
        editor.closeDir();
        // 关闭编辑器，并返回执行结果
        return editor.closeEdit();
    }

    /**
     * 新增文件
     * @param editor
     * @param revisionNo
     * @return
     * @throws SVNException
     */
    private static SVNCommitInfo addFile(ISVNEditor
editor,long revisionNo) throws SVNException{
        // 进入Root节点，即nodeB
        editor.openRoot(revisionNo);
        //.进入nodeC节点
        editor.openDir("nodeC",revisionNo);
        // 新增itemC2文件
        editor.addFile("nodeC/itemC2",null,revisionNo);
        //确保客户端这个文件的内容和服务端的是一样的，如果不一致的
话是不允许提交的。底层实现使用MD5
        String itemC2Path = "nodeC/itemC2";
        String baseChecksum = null;
        editor.applyTextDelta(itemC2Path,baseChecksum);
        //提交文件变更的数据,windows默认是100kb大小
        byte[] oldData = new byte[]{};//旧数据
        byte[] newData = null;//新数据
        try {
            newData = "我来测试一下 -
addFile".getBytes("utf-8");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        ByteArrayInputStream baseData = new
ByteArrayInputStream(oldData);
        ByteArrayInputStream workingData = new
ByteArrayInputStream(newData);
        SVNDeltaGenerator svnDeltaGenerator = new
SVNDeltaGenerator();//100KB-windows generator
        String checksum =
svnDeltaGenerator.sendDelta(itemC2Path,baseData,0,working
Data,editor,true);
        //设置文件的属性，key是字符串，值被包装成
SVNProperyValue了

editor.changeFileProperty("nodeC/itemC2","properName1",SV
NPropertyValue.create("properValue1"));

editor.changeFileProperty("nodeC/itemC2","properName2",SV
```

```
NPropertyValue.create("properValue2"));
        System.out.println("checksum:" + checksum );
        //关闭文件
        editor.closeFile("nodeC/itemC2",checksum);
        //关闭目录nodeC
        editor.closeDir();
        //关闭root
        editor.closeDir();
        return editor.closeEdit();
    }

    /**
     * 新增目录
     * @param editor 编辑器
     * @param revisionNo 修订版本号
     * @return  SVNCommitInfo 提交结果信息
     * @throws SVNException
     */
    private static SVNCommitInfo addDir(ISVNEditor
editor,long revisionNo) throws SVNException{
        // 进入Root节点，即nodeB
        editor.openRoot(revisionNo);
        //新增目录
        editor.addDir("nodeD",null,revisionNo);
        editor.closeDir();//nodeD
        editor.closeDir();//nodeB
        return editor.closeEdit();
    }
}
```

运行效果：

```
执行删除操作的返回结果：r51 by 'wly' at Wed Dec 07 13:48:15
CST 2016
执行编辑操作的返回结果：r52 by 'wly' at Wed Dec 07 13:48:15
CST 2016
checksum:a107fb58070bfbaf11513c8750f87466
执行新增文件操作的返回结果：r53 by 'wly' at Wed Dec 07
13:48:15 CST 2016
{"myProperties":{"svn:entry:uuid":{"myValue":"e5dd1e38-
0390-574a-b68d-e269ce50c382"},"svn:entry:revision":
{"myValue":"53"},"properName1":{"myData":
[112,114,111,112,101,114,86,97,108,117,101,49]},"svn:entr
y:committed-date":{"myValue":"2016-12-
07T05:48:15.464756Z"},"properName2":{"myData":
[112,114,111,112,101,114,86,97,108,117,101,50]},"svn:wc:r
a_dav:version-url":
{"myValue":"/svn/svnkitRepository2/!svn/ver/53/trunk/node
B/nodeC/itemC2"},"svn:entry:checksum":
{"myValue":"a107fb58070bfbaf11513c8750f87466"},"svn:entry
:committed-rev":{"myValue":"53"},"svn:entry:last-author":
```

```
{"myValue":"wly"}}}
```

执行新增目录操作的返回结果：`r54 by 'wly' at Wed Dec 07 13:48:15 CST 2016`

总结：

其实走读一遍代码就知道，无论进行什么操作都是有一定规律性的。

无论是操作目录还是文件，大的框架可以大体总结为以下几步：

```
//1.根据访问协议初始化工厂
//2.初始化仓库，由于我们所有的操作都是基于nodeB节点以下的，所以我们将nodeB作为本次操作的root节点
//3.初始化权限
//4.获取编辑器对象
//5.进入目录/文件
//6.执行操作
//7.关闭目录/文件
//8.关闭编辑器
```

实际工作中，感觉这种方式不是特别灵活，不一定适用于普通的应用场景，相对来讲，High-Level API更倾向于用户和SVN的交互。

分类: 版本控制、管理工具

标签: svnkit, java, svn, svn二次开发, 版本控制, 版本管理

鱼我所欲也
关注 - 1
粉丝 - 8
+加关注

« 上一篇：SVNKit学习——基于Repository的操作之print repository tree、file content、repository history(四)
» 下一篇：SVNKit学习——使用High-Level API管理Working Copy示例(六)

posted @ 2016-12-07 14:04 鱼我所欲也 阅读(1079) 评论(1) 编辑 收藏

评论列表

#1楼 2017-02-09 23:50 tinyhead

受益匪浅。最近我们也正在用svnkit做二次开发，刚做了版本变更的通知功能。现在正在找API，能查出目录的用户列表，但一直没找到，请问楼主有啥建议不？
在windows服务端VisualSvnServer里面对 目录 点右键--选中properties--security 中就有显示用户列表。

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部