使用svnkit api，纯java操作svn，实现svn提交，更新等操作（修正版）

作者：hardwin (http://my.csdn.net/hardwin)

此篇是在上一篇基础上修改了bug。

```
import java.io.File;

import org.apache.log4j.Logger;
import org.tmatesoft.svn.core.SVNCommitInfo;
import org.tmatesoft.svn.core.SVNDepth;
import org.tmatesoft.svn.core.SVNException;
import org.tmatesoft.svn.core.SVNNodeKind;
import org.tmatesoft.svn.core.SVNURL;
import org.tmatesoft.svn.core.auth.ISVNAuthenticationManager;
import org.tmatesoft.svn.core.internal.io.dav.DAVRepositoryFactory;
import org.tmatesoft.svn.core.internal.io.fs.FSRepositoryFactory;
import org.tmatesoft.svn.core.internal.io.svn.SVNRepositoryFactoryImpl;
import org.tmatesoft.svn.core.internal.wc.DefaultSVNOptions;
import org.tmatesoft.svn.core.io.SVNRepository;
import org.tmatesoft.svn.core.io.SVNRepositoryFactory;
import org.tmatesoft.svn.core.wc.SVNClientManager;
import org.tmatesoft.svn.core.wc.SVNRevision;
import org.tmatesoft.svn.core.wc.SVNStatus;
import org.tmatesoft.svn.core.wc.SVNUpdateClient;
import org.tmatesoft.svn.core.wc.SVNWCUtil;

/**
 * SVNKit Utility
 * @author lena yang
 *
 */
public class SVNUtil {

        private static Logger logger = Logger.getLogger(SVNUtil.class);

        /**
         * 通过不同的协议初始化版本库
         */
        public static void setupLibrary() {
                DAVRepositoryFactory.setup();
                SVNRepositoryFactoryImpl.setup();
                FSRepositoryFactory.setup();
        }

        /**
         * 验证登录svn
         */
        public static SVNClientManager authSvn(String svnRoot, String username,
                        String password) {
                // 初始化版本库
                setupLibrary();

                // 创建库连接
                SVNRepository repository = null;
                try {
                        repository = SVNRepositoryFactory.create(SVNURL
                                        .parseURIEncoded(svnRoot));
                } catch (SVNException e) {
                        logger.error(e.getErrorMessage(), e);
                        return null;
                }

                // 身份验证
                ISVNAuthenticationManager authManager = SVNWCUtil

                .createDefaultAuthenticationManager(username, password);

                // 创建身份验证管理器
                repository.setAuthenticationManager(authManager);

                DefaultSVNOptions options = SVNWCUtil.createDefaultOptions(true);
                SVNClientManager clientManager = SVNClientManager.newInstance(options,
                                authManager);
                return clientManager;
        }

        /**
         * Make directory in svn repository
         * @param clientManager
         * @param url
         *                      eg: http://svn.ambow.com/wlpt/bsp/trunk
         * @param commitMessage
         * @return
         * @throws SVNException
         */
        public static SVNCommitInfo makeDirectory(SVNClientManager clientManager,
                        SVNURL url, String commitMessage) {
                try {
                        return clientManager.getCommitClient().doMkDir(
                                        new SVNURL[] { url }, commitMessage);
                } catch (SVNException e) {
                        logger.error(e.getErrorMessage(), e);
                }
                return null;
        }

        /**
         * Imports an unversioned directory into a repository location denoted by a
         *      destination URL
         * @param clientManager
         * @param localPath
         *                      a local unversioned directory or singal file that will be importe
d into a
         *                      repository;
         * @param dstURL
         *                      a repository location where the local unversioned directory/file
 will be
     *                  imported into
         * @param commitMessage
         * @param isRecursive 递归
         * @return
         */
        public static SVNCommitInfo importDirectory(SVNClientManager clientManager,
                        File localPath, SVNURL dstURL, String commitMessage,
                        boolean isRecursive) {
```

```
		try {
			return clientManager.getCommitClient().doImport(localPath, dstURL,
					commitMessage, null, true, true,
					SVNDepth.fromRecurse(isRecursive));
		} catch (SVNException e) {
			logger.error(e.getErrorMessage(), e);
		}
		return null;
	}

	/**
	 * Puts directories and files under version control
	 * @param clientManager
	 *			SVNClientManager
	 * @param wcPath
	 *			work copy path
	 */
	public static void addEntry(SVNClientManager clientManager, File wcPath) {
		try {
			clientManager.getWCClient().doAdd(new File[] { wcPath }, true,
					false, false, SVNDepth.INFINITY, false, false,
					true);
		} catch (SVNException e) {
			logger.error(e.getErrorMessage(), e);
		}
	}

	/**
	 * Collects status information on a single Working Copy item
	 * @param clientManager
	 * @param wcPath
	 *			local item's path
	 * @param remote
	 *			true to check up the status of the item in the repository,
	 *			that will tell if the local item is out-of-date (like '-u' option
in the SVN client's
	 *			'svn status' command), otherwise false
	 * @return
	 * @throws SVNException
	 */
	public static SVNStatus showStatus(SVNClientManager clientManager,
			File wcPath, boolean remote) {
		SVNStatus status = null;
		try {
			status = clientManager.getStatusClient().doStatus(wcPath, remote);
		} catch (SVNException e) {
			logger.error(e.getErrorMessage(), e);
		}
		return status;
	}

	/**
	 * Commit work copy's change to svn
	 * @param clientManager
	 * @param wcPath
```

```
         *                       working copy paths which changes are to be committed
         * @param keepLocks
         *                       whether to unlock or not files in the repository
         * @param commitMessage
         *                       commit log message
         * @return
         * @throws SVNException
         */
        public static SVNCommitInfo commit(SVNClientManager clientManager,
                        File wcPath, boolean keepLocks, String commitMessage) {
                try {
                        return clientManager.getCommitClient().doCommit(
                                        new File[] { wcPath }, keepLocks, commitMessage, null,
                                        null, false, false, SVNDepth.INFINITY);
                } catch (SVNException e) {
                        logger.error(e.getErrorMessage(), e);
                }
                return null;
        }

    /**
     * Updates a working copy (brings changes from the repository into the working copy).
     * @param clientManager
     * @param wcPath
     *                  working copy path
     * @param updateToRevision
     *                  revision to update to
     * @param depth
     *                  update的深度：目录、子目录、文件
     * @return
     * @throws SVNException
     */
        public static long update(SVNClientManager clientManager, File wcPath,
                        SVNRevision updateToRevision, SVNDepth depth) {
                SVNUpdateClient updateClient = clientManager.getUpdateClient();

                /*
                 * sets externals not to be ignored during the update
                 */
                updateClient.setIgnoreExternals(false);

                /*
                 * returns the number of the revision wcPath was updated to
                 */
                try {
                        return updateClient.doUpdate(wcPath, updateToRevision,depth, false, fals
e);
                } catch (SVNException e) {
                        logger.error(e.getErrorMessage(), e);
                }
                return 0;
        }

        /**
         * recursively checks out a working copy from url into wcDir
         * @param clientManager
         * @param url
         *                       a repository location from where a Working Copy will be checked o
ut
         * @param revision
         *                       the desired revision of the Working Copy to be checked out
         * @param destPath
         *                       the local path where the Working Copy will be placed
         * @param depth
         *                       checkout的深度，目录、子目录、文件
         * @return
         * @throws SVNException
         */
        public static long checkout(SVNClientManager clientManager, SVNURL url,
                        SVNRevision revision, File destPath, SVNDepth depth) {

                SVNUpdateClient updateClient = clientManager.getUpdateClient();
                /*
                 * sets externals not to be ignored during the checkout
                 */
                updateClient.setIgnoreExternals(false);
                /*
                 * returns the number of the revision at which the working copy is
                 */
                try {
                        return updateClient.doCheckout(url, destPath, revision, revision,depth, f
alse);
                } catch (SVNException e) {
                        logger.error(e.getErrorMessage(), e);
                }
                return 0;
        }

        /**
         * 确定path是否是一个工作空间
         * @param path
         * @return
         */
        public static boolean isWorkingCopy(File path){
                if(!path.exists()){
                        logger.warn("'" + path + "' not exist!");
                        return false;
                }
                try {
                        if(null == SVNWCUtil.getWorkingCopyRoot(path, false)){
                                return false;
                        }
                } catch (SVNException e) {
                        logger.error(e.getErrorMessage(), e);
                }
                return true;
        }

        /**
         * 确定一个URL在SVN上是否存在
```

```
	 * @param url
	 * @return
	 */
	public static boolean isURLExist(SVNURL url,String username,String password){
		try {
			SVNRepository svnRepository = SVNRepositoryFactory.create(url);
			ISVNAuthenticationManager authManager = SVNWCUtil.createDefaultAuthentica
tionManager(username, password);
			svnRepository.setAuthenticationManager(authManager);
			SVNNodeKind nodeKind = svnRepository.checkPath("", -1);
			return nodeKind == SVNNodeKind.NONE ? false : true;
		} catch (SVNException e) {
			e.printStackTrace();
		}
		return false;
	}

}
```

```
/**
 * 创建项目框架，SVN提交、更新
 * @author lena yang
 *
 */
@Service("svnProjectService")
public class SvnProjectService {

        private Logger logger = Logger.getLogger(SvnProjectService.class);

        // 项目的存放位置
        private String workspace = null;

        private ResourceBundle rb = ResourceBundle.getBundle("application");

        // SVN的用户名、密码
        private String username = null;
        private String password = null;

        private String templete = null;

        @Resource(name="xcodeService")
        private XcodeService xcodeService;


        private void init(){
                String webapp = System.getProperty("webapp.root");
                if(null!=webapp&&!webapp.endsWith("/") && !webapp.endsWith("\\")){
                        webapp = webapp + "/";
                }
                // 发布到web服务器以后，有可能WebContent没了
                if(new File(webapp + "WebContent").exists()){
                        webapp = webapp + "WebContent";
                }
                username = rb.getString("svn.username");
                password = rb.getString("svn.password");
                workspace = rb.getString("project.svn.path");
                templete = webapp + "templete";
        }

        public SvnProjectService(){
                super();
                init();
        }

        /**
         * 创建项目框架
         * @param project
         *                      Project
         * @return
         */
        public boolean createProjectFrame(Project project,List<String> tableNames) {
                if(project == null){
                        return false;
                }
                File src = new File(templete);  // 模板项目的位置
                File ws = new File(workspace);  // work copy
                if(!ws.exists()){
                        ws.mkdirs();
                }
                File dest = new File(workspace + "/" + project.getName());
                if(!dest.exists()){
                        dest.mkdirs();
                }

                checkWorkCopy(project); // 确定工作空间

                // 复制模板项目到工作空间
                try {
                        FileUtils.copyDirectory(src, dest);
                } catch (IOException e) {
                        logger.error(e.getMessage(), e);
                        return false;
                }

                //修改.project文件中的内容
                editProjectFile(project);

                // 生成框架代码
                xcodeService.createBaseFrameWithDatasource(project,tableNames);

                // 提交到SVN
                commitProjectToSvn(project);

                return true;
        }

        /**
         * 修改项目文件
         * @param project
         * @throws DocumentException
         * @throws IOException
         */
        @SuppressWarnings("unchecked")
        private void editProjectFile(Project project) {

                String pro = workspace + "/" + project.getName() + "/";

                String settings = pro + ".settings/";

                // 1. 修改.settings/org.eclipse.wst.common.component
                Document document = null;
                try {
                        document = XmlReaderUtil.getDocument(new File(settings
                                        + "org.eclipse.wst.common.component"));
                } catch (DocumentException e) {
                        e.printStackTrace();
                }

                Element root = document.getRootElement();
                root.element("wb-module").attribute("deploy-name")
                                .setValue(project.getName());

                if (root.element("wb-module").element("property").attribute("name")
                                .getValue().equals("java-output-path")) {

                        root.element("wb-module").element("property").attribute("value")
                                        .setValue("/" + project.getName() + "/build/classes");
                }

                Iterator<Element> itr = (Iterator<Element>) XmlReaderUtil.getElements(
                                document, "//wb-module//property").iterator();

                while (itr.hasNext()) {
                        Element element = itr.next();
                        if ("context-root".equals(element.attribute("name").getValue())) {
                                element.attribute("value").setValue("/" + project.getName());
                        }
                }

                // 将修改后的值写入
                XmlReaderUtil.writerXml(document, settings
                                + "org.eclipse.wst.common.component");

                // 2. 修改.project
                try {
                        document = XmlReaderUtil.getDocument(new File(pro + ".project"));
                        XmlReaderUtil.setElementText(document, "//projectDescription",
                                        "name", project.getName());
                        XmlReaderUtil.writerXml(document, pro + ".project");
                } catch (DocumentException e) {
                        e.printStackTrace();
                }
        }

        /**
         * 从SVN更新项目到work copy
         * @param project
         *                      Project
         * @return
         */
        public boolean updateProjectFromSvn(Project project) {
                if(null == project || null == rb.getString("svn.url")){
                        return false;
                }
                project.setSvnUrl(rb.getString("svn.url"));

        SVNClientManager clientManager = SVNUtil.authSvn(project.getSvnUrl(), username, passwor
d);
                if (null == clientManager) {
                        logger.error("SVN login error! >>> url:" + project.getSvnUrl()
                                        + " username:" + username + " password:" + password);
                        return false;
                }
        // 注册一个更新事件处理器
        clientManager.getCommitClient().setEventHandler(new UpdateEventHandler());

        SVNURL repositoryURL = null;
                try {
                        // eg: http://svn.ambow.com/wlpt/bsp
                        repositoryURL = SVNURL.parseURIEncoded(project.getSvnUrl()).appendPath("t
runk/"+project.getName(), false);
                } catch (SVNException e) {
                        logger.error(e.getMessage(),e);
                        return false;
                }

        File ws = new File(new File(workspace), project.getName());
        if(!SVNWCUtil.isVersionedDirectory(ws)){
                SVNUtil.checkout(clientManager, repositoryURL, SVNRevision.HEAD, new File(workspa
ce), SVNDepth.INFINITY);
        }else{
                SVNUtil.update(clientManager, ws, SVNRevision.HEAD, SVNDepth.INFINITY);
        }
                return true;
        }

        /**
         * 提交项目到SVN
         * @param project
         *                      Project
         * @return
         */
        public boolean commitProjectToSvn(Project project) {
        SVNClientManager clientManager = SVNUtil.authSvn(project.getSvnUrl(), username, passwor
d);

        clientManager.getCommitClient().setEventHandler(new CommitEventHandler());

        File wc_project = new File( workspace + "/" + project.getName());

        checkVersiondDirectory(clientManager,wc_project);

                SVNUtil.commit(clientManager, wc_project, false, "svnkit");

                return true;
        }

        /**
         * 递归检查不在版本控制的文件，并add到svn
         * @param clientManager
         * @param wc
         */
        private void checkVersiondDirectory(SVNClientManager clientManager,File wc){
                if(!SVNWCUtil.isVersionedDirectory(wc)){
                        SVNUtil.addEntry(clientManager, wc);
                }
                if(wc.isDirectory()){
                        for(File sub:wc.listFiles()){
                                if(sub.isDirectory() && sub.getName().equals(".svn")){
                                        continue;
                                }
                                checkVersiondDirectory(clientManager,sub);
                        }
                }
        }

        private void checkWorkCopy(Project project){
                project.setSvnUrl(rb.getString("svn.url"));
                SVNClientManager clientManager = SVNUtil.authSvn(project.getSvnUrl(), username, p
assword);

                SVNURL repositoryURL = null;    // trunk
                try {
                        // eg: http://svn.ambow.com/wlpt/bsp
                        repositoryURL = SVNURL.parseURIEncoded(project.getSvnUrl())
                                        .appendPath("trunk", false);
                } catch (SVNException e) {
                        logger.error(e.getMessage(),e);
                }

                File wc = new File(workspace);
                File wc_project = new File( workspace + "/" + project.getName());

                SVNURL projectURL = null;       // projectName
                try {
                        projectURL = repositoryURL.appendPath(project.getName(), false);
                } catch (SVNException e) {
                        logger.error(e.getMessage(),e);
                }

                if(!SVNUtil.isWorkingCopy(wc)){
                        if(!SVNUtil.isURLExist(projectURL,username,password)){
                                SVNUtil.checkout(clientManager, repositoryURL, SVNRevision.HEAD,
 wc, SVNDepth.EMPTY);
                        }else{
                                SVNUtil.checkout(clientManager, projectURL, SVNRevision.HEAD, wc_
project, SVNDepth.INFINITY);
                        }
                }else{
                        SVNUtil.update(clientManager, wc, SVNRevision.HEAD, SVNDepth.INFINITY);
                }
        }

}
```

文章主题是svnkit的api操作。所以涉及到其它类的，可以忽略。svnkit有两套api，high level和low level 。high level是操作工作拷贝即working copy的。

我这儿用的就是high level。

checkWorkCopy方法里面，先检查当前指定的路径是否是一个working copy。如果是，执行update操作；

如果不是的话，分两种情况：

1. svn上已经有以此project命名的项目，可能svn上有了，但working copy被删了什么的。就将该project checkout下来；

2. svn上没有该project。此时checkout一个空的东西。SVNDepth.EMPTY，只是让我们的路径成为一个working copy，路径下出现一个.svn目录，svn上面的文件不会被checkout下来。因为我们的svn上可能项目很多。如果把整个上级目录checkout下来会很慢也浪费硬盘空间。

用 svnkit 执行commit操作，概况来说，流程如下：

1. 保证本地path是一个working copy。 即checkWorkCopy方法

2. woking copy可能发生添加，更新，删除等变化操作

3. 检查working copy里面的文件是否under version control。即checkVersiondDirectory方法

4. 执行commit操作

查看原文>> (http://blog.csdn.net/hardwin/article/details/7963318)

0

看过本文的人也看了：

- Java 知识结构图 (http://lib.csdn.net/base/java/structure)
- Eclipse SVN 安装使用笔记 (http://lib.csdn.net/article/java/22954)
- svn的使用详细说明 (http://lib.csdn.net/article/java/12719)
- java操作svn【svnkit】实操 (http://lib.csdn.net/article/java/22813)
- MyEclipse10 安装SVN插件及SVN插件在... (http://lib.csdn.net/article/java/12717)
- Java配置maven+jenkins+git（svn）+tomcat... (http://lib.csdn.net/article/java/64033)

发表评论

输入评论内容

发表

16个评论

(http://my.csdn.net/u012922262)

u012922262 (http://my.csdn.net/u012922262)

JAVA企业级框架：https://item.taobao.com/item.htm?spm=686.1000925.0.0.PN9Zga&id=541298019664

2016-11-16 23:57:46 回复

(http://my.csdn.net/lonely_Quan)

lonely_Quan (http://my.csdn.net/lonely_Quan)

辛苦楼主了，如果还有保存源代码工程的话能发我一份吗？269872441@qq.com。好人一生平安

2016-07-26 14:28:44 回复

(http://my.csdn.net/hardwin)

hardwin (http://my.csdn.net/hardwin)

回复lonely_Quan： 找不到原来的项目了

2016-10-04 20:07:47 回复

(http://my.csdn.net/tmhzxc)

tmhzxc (http://my.csdn.net/tmhzxc)

非常有参考价值!不过有的地方不是很全!得自己研究一下,自己补上了剩余的代码!

2015-11-07 18:22:13 回复

(http://my.csdn.net/qq_16918809)

qq_16918809 (http://my.csdn.net/qq_16918809)

非常好呀唯一美中不足的就是代码太少，能不能把源代码贡献出来。感谢楼主

2014-09-29 17:40:15 回复

加载更多