Seamless round trip editing for Domino attachments

Summary

Users expect a seamless editing experience when using office type documents (word processing, spreadsheets, presentation, graphics, PDF) in web applications. Today this is typically achived using an ActiveX plug-in into Internet Explorer.

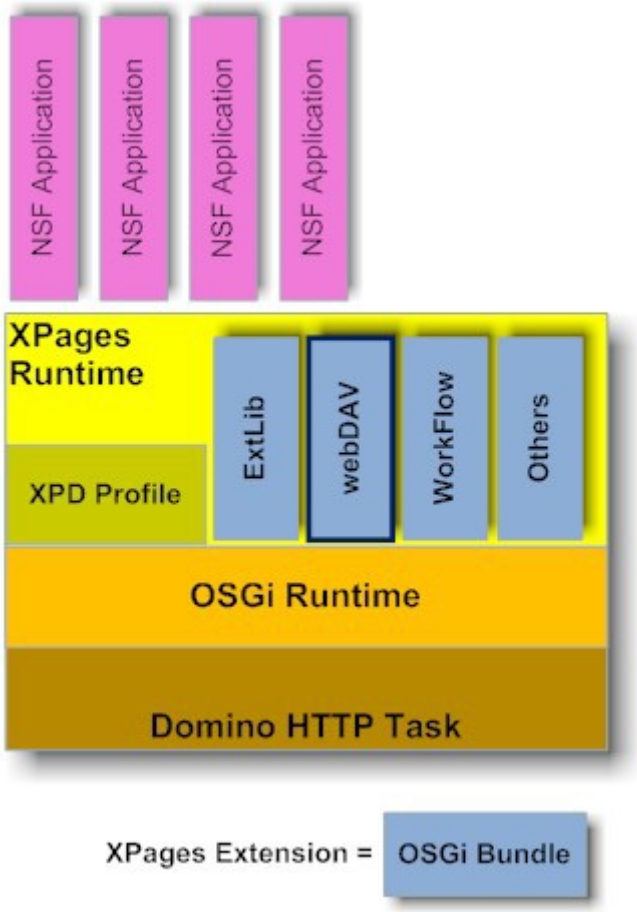
This approach is problematic in the future for a number of reasons:

- ActiveX has been subject to security concerns
- ActiveX is limited to Internet Explorer and Microsoft Windows. So the approach cannont work on computers using Apple Mac OS/X or Linux and, more importantly, can't work on mobile devices like iPad, iPhone or Android

The alternative approach is to make files available using the open standard HTTP extension webDAV. Using webDAV documents can be directly opened using the familiar File – Open dialog. For full integration into web applications the protocol webdav:// and webdavs:// are used. A protocol handler on the device (PC/Mobile) makes files accessible. All security settings (reader/author) of Domino are fully respected.

Architecture

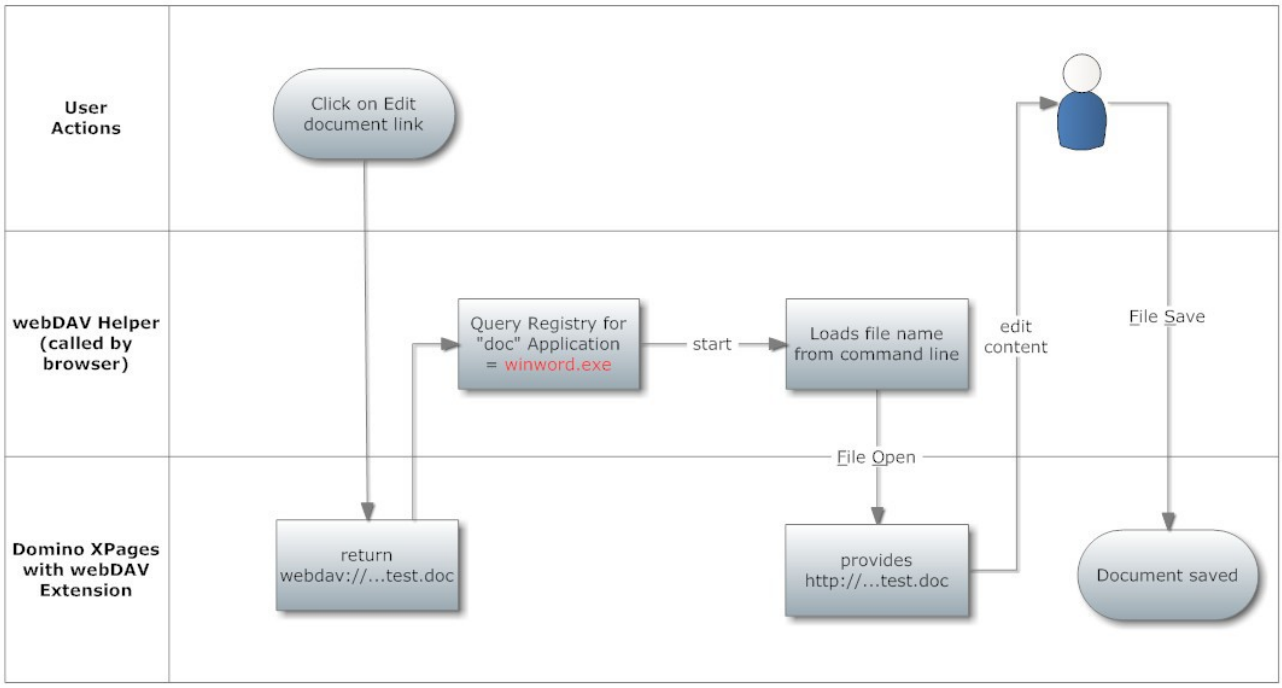
The webDAV extension has been implemented as Domino XPages extension (OSGi bundle) and requires a Domino 8.5.2 or later server. It follows the Extension API guidelines and co-exists with other extensions like the Extension library.



Configuration of available data sources is handled in the webdavcfg.nsf database that needs to be in the root directory of the server. Using the configuration database the amount of available documents can be precisely specified. Currently supported are: Files in a server directory and Attachments of documents in a view. More entries can be added later on.

It is the responsibility of the application developer to provide an URL that begins with webdav:// or webdavs:// It contains the server as the current domino server and the short-name given to the location in the configuration document. Using a shortname allows to later move resource locations without breaking existing links (just the entry in the configuration needs to be updated).

For the user the process is completely transparent. They click on a link and the respective application (MS Office, Photoshop, OpenOffice etc.) opens in Edit mode (if the user has edit rights):



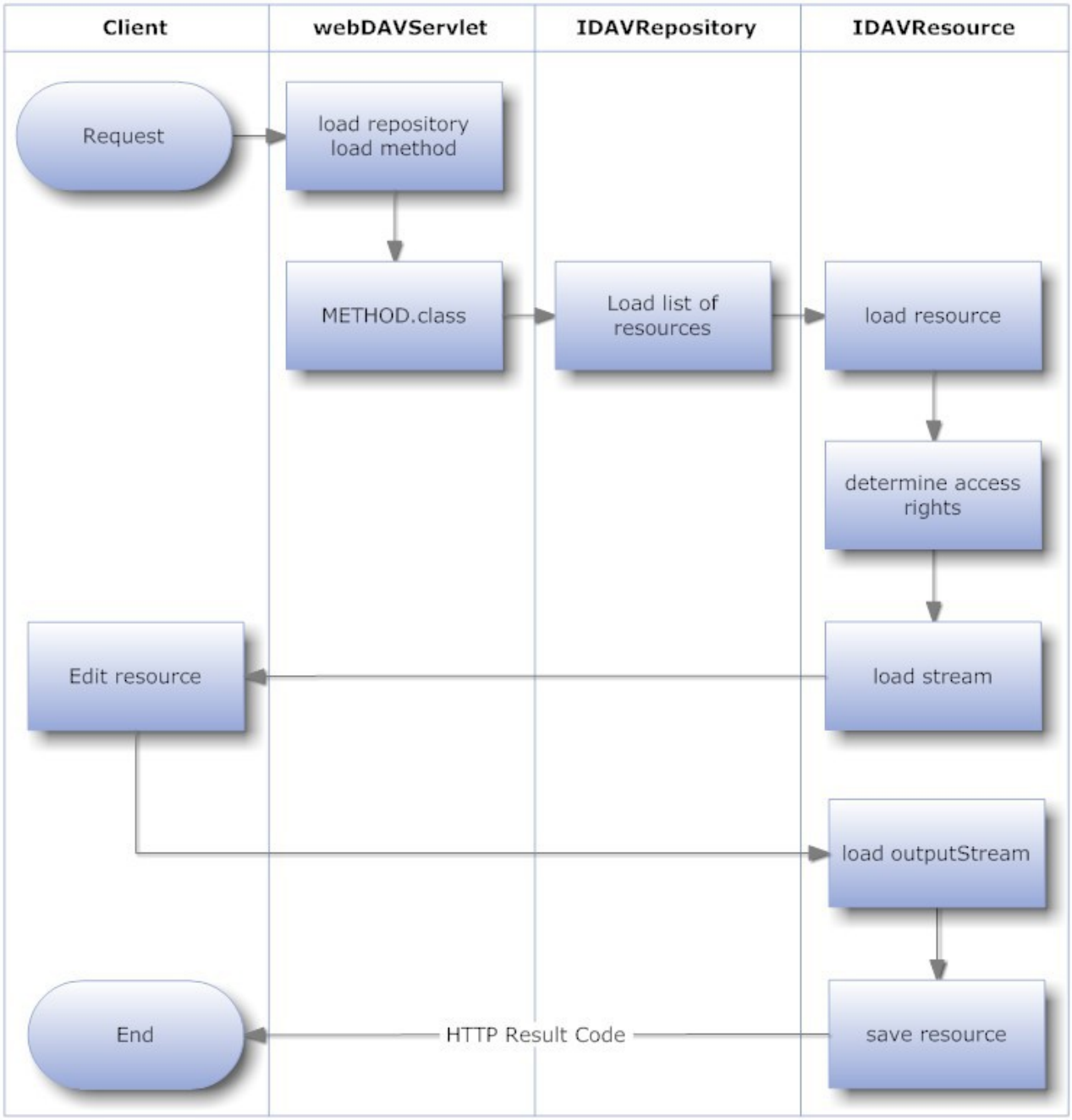
There are key advantages for this approach (besides the browser and platform independence);

1. The document can be edited and saved even if the user closes the browser window (or the browser crashes)
2. The document can be located anywhere. It doesn't need to be an attachment in the document displayed in the browser. Just the link is necessary
3. Concurrent access: since webDAV provides document locking the file opened in edit mode in the office application is locked and other users can't edit that. Works cross platform

API

The extension uses a class WebDAvServlet that handles all incoming and outgoing requests. The servlet delegates every HTTP or webDAV method to a specialized Java class. So additional methods (e.g. for calDAV) can be implemented easily without the need to change existing code.

The plug-in loads a list of repositories that implement the interface IDAVRepository. Each repository loads resources using classes that implement IDAVResource. IDAVResource has a method getStream() and getOutputStream() that provide the stream to read and write the content of the resource. There is no dependency on a File API, so a resource could be completely virtual (e.g. a list of available reports that is only generated once a report gets requested.



Availability

The webDAV for Domino plug-in is planned for a September release on OpenNTF including its source code.

Tasks left to to

There are a number of task that will make the round trip experience more complete. What we know is:

- Provide sample JavaScript for helper discovery
- Provide helpers for Mac OS/X, iOS, Linux and Android
- Provide install script support for Chrome, Safari (we do FF & IE for now)
- Provide a better sample database
- Provide additional content type plug-ins

Appendix – The Interfaces

```
/* ============================================================================ *
 * Copyright (C) 2006, 2007 TAO Consulting Pte <http://www.taoconsulting.sg/> *
 *           based on work of                                                   *
 * Copyright (C) 2004-2005 Pier Fumagalli <http://www.betaversion.org/~pier/> *
 *                           All rights reserved.                             *
 * ============================================================================ *
 *                                                                              *
 * Licensed under the  Apache License, Version 2.0  (the "License").  You may *
 * not use this file except in compliance with the License.  You may obtain a *
 * copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>.       *
 *                                                                              *
 * Unless  required  by applicable  law or  agreed  to  in writing,  software *
 * distributed under the License is distributed on an  "AS IS" BASIS, WITHOUT *
 * WARRANTIES OR  CONDITIONS OF ANY KIND, either express or implied.  See the *
 * License for the  specific language  governing permissions  and limitations *
 * under the License.                                                           *
 *                                                                              *
 * ============================================================================ */
package biz.taoconsulting.dominodav.interfaces;

import java.util.HashSet;
import java.util.Set;

import com.ibm.xsp.webdav.DAVCredentials;
import biz.taoconsulting.dominodav.exceptions.DAVNotFoundException;
import biz.taoconsulting.dominodav.repository.DAVRepositoryListing;

/**
 * @author stw
 *
 * Interface that defines all repository methods a lightweight repository needs
 * to support.
 *
 *
 */
public interface IDAVRepository {

	/**
	 * <p>
	 * A {@link String} of all acceptable characters in an URI.
	 * </p>
	 */
	String ACCEPTABLE = "ABCDEFGHIJLKMNOPQRSTUVWXYZ" + // ALPHA
				// (UPPER)
				"abcdefghijklmnopqrstuvwxyz" + // ALPHA (LOWER)
				"0123456789" + // DIGIT
				"_-!.~'()*" + // UNRESERVED
				",;:$&+=" + // PUNCT
				"?/[]@"; // RESERVED

	/**
	 *
```

```
         */
        int RESOURCE_TYPE_ALL = 2;

        /**
         *
         */
        int RESOURCE_TYPE_COLLECTION = 1;

        /**
         *
         */
        int RESOURCE_TYPE_FILE = 0;

        /**
         * @param listener adds a listener
         */
        void addListener(IDAVListener listener);

        /**
         * @param uri The location as seen in the browser
         * @return the collection
         */
        IDAVResource createNewCollection(String uri);

        /**
         * @param externalAddress The location as seen in the browser
         * @return the resource
         */
        IDAVResource createNewResource(String externalAddress);


        /**
         *
         * @return Space separated String with method names
         */
        String getAvailableMethods();

        /**
         * @return The RepositoryListing object
         */
        DAVRepositoryListing getRepositoryListing();

        /**
         * @param requestURL - The path in the URL minus protocol/server to request this
resource
         * @return Resourceobject
         * @throws DAVNotFoundException If Resource could not be localized
         */
        IDAVResource getResource(String requestURL) throws DAVNotFoundException;

        /**
         * @param requestURL - The path in the URL minus protocol/server to request this
resource
         * @param withoutChildren mark Resource as member (hide its children)
         * @return Resourceobject
         * @throws DAVNotFoundException If Resource could not be localized
         */
        IDAVResource getResource(String requestURL, boolean withoutChildren) throws
DAVNotFoundException;

        /**
         *
         * @return credential DAVcredentials
         */
        DAVCredentials getCredentials();

        /**
         *
```

```
         * @param method Method to check
         * @return boolean isSupportedMethod
         */
        boolean isSupportedMethod(String method);

        /**
         *
         * @param from original location
         * @param to new location
         * @return true/false
         */
        int moveResource(String from, String to);
        /**
         * @param listener removes a listener
         */
        void removeListener(IDAVListener listener);


        /**
         *
         * @param availableMethods  method names in Uppercase
         */
        void setAvailableMethods(HashSet<String> availableMethods);

        /**
         *
         * @param availableMethods String method names in Uppercase separated by comma or
space
         */
        void setAvailableMethods(String availableMethodsString);

        /**
         * @param cred The DAVCredential object
         * @return True if Login was successful, otherwise False
         */
        boolean setCredentials(DAVCredentials cred);

        /**
         * @param repositoryListing {...}
         */
        void setRepositoryListing(DAVRepositoryListing repositoryListing);


        /**
         * @param resc The resource which shall be written
         */
        void writeResource(IDAVResource resc);

        /**
         *
         * @return The current listeners configured for the resource
         */
        Set<IDAVListener> getListeners();

        /**
         *
         * @param listeners Listeners to get notified on changes
         */
        void setListeners(Set<IDAVListener> listeners);


        /**
         * @param tempDir the directory for temporary file operations
         */
        void setTempDir(String tempDir);

        /**
         * @return The current temporary directory
```

```
	 */
	String getTempDir();

	/**
	 * Convert the external URL into the respective internal URI. Can be a file name
	 * or a database path or a call to an external system. The mapping is per
repository
	 * @param externalURL
	 * @return the internal URL
	 */
	String getInternalAddressFromExternalUrl(String externalURL, String callee);

}

/*
 * ==================================================================== *
 * Copyright (C) 2006, 2007 TAO Consulting Pte <http://www.taoconsulting.sg/> *
 * All rights reserved. *
 * ==================================================================== *
 * * Licensed under the Apache License, Version 2.0 (the "License"). You may *
 * not use this file except in compliance with the License. You may obtain a *
 * copy of the License at <http://www.apache.org/licenses/LICENSE-2.0>. * *
 * Unless required by applicable law or agreed to in writing, software *
 * distributed under the License is distributed on an "AS IS" BASIS, WITHOUT *
 * WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the *
 * License for the specific language governing permissions and limitations *
 * under the License. * *
 * ====================================================================
 */

package biz.taoconsulting.dominodav.interfaces;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.Date;
import java.util.Vector;

import com.ibm.xsp.webdav.interfaces.IDAVXMLResponse;

//import biz.taoconsulting.dominodav.DAVProperties;

/**
 * IDAVResource represents an entry in a WebDAV Repository. It can be a
 * collection (a.k.a a directory) containing other Resources or it can be a
 * "file" which is something that returns a stream or can be written to as a
 * stream. A resource has 3 identifiers The href = access to the resource
 * relative to the repository as seen from the browser The path = access to the
 * resource as seen from it's internal mechanism like (the absolute) path in a
 * file system or url or query statement The uri = access to the resource from
 * the browser including servlet and repository
 *
 * @author stw
 */
public interface IDAVResource {

	/**
	 * addToKXml adds reponse parts to the XML output
	 *
	 * @param dxr
	 *            - Object that writes XML information for responses in webDAV
	 *            standard
	 * @throws IOException
	 *             for XML Errors
	 */
	public void addToDavXMLResponse(IDAVXMLResponse dxr) throws IOException;

	/**
```

```
     * @return true/false - Success of delete operation
     */
    public abstract boolean delete();

    /**
     * @return Length of content
     */
    public Long getContentLength();

    /**
     * @return Date CreationDate (not supported in Java)
     */
    public Date getCreationDate();

    /**
     * eTag is to identify a resource compared to other versions of it
     * @return String eTag
     */
    public String getETag();

    /**
     * @return String the file extension
     */
    public String getExtension();

    /**
     * @return Date LastModified Date
     */
    public Date getLastModified();

    /**
     * @return mime type of file resource
     */
    public String getMimeType();

    /**
     * @return Outputstream to update resource
     */
    public abstract OutputStream getOutputStream();

    /**
     * @return Owner of this resource
     */
    public IDAVRepository getOwner();

    /**
     * @return DAVProperties: all properties of the resource
     */
    // public DAVProperties getProperties();

    /**
     * @return repository - the owning repository
     */
    public IDAVRepository getRepository();

    /**
     * @return The internal type of resource Externally we only have files and
     *         directories internally it can be anything
     */
    public String getResourceType();

    /**
     * @return InputStream - Stream Object to read resource
     */
    public InputStream getStream();

    /**
     * @return boolean: is it a collection/directory
```

```
     */
    public boolean isCollection();

    /**
     * @return is it a member, so there won't be any sub elements in it
     */
    public boolean isMember();

    /**
     * @return Returns the readOnly.
     */
    public boolean isReadOnly();

    /**
     * @param isCollection
     *            declare it a collection with true
     */
    public void setCollection(boolean isCollection);

    /**
     * @param isMember
     *            Make it a member
     */
    public void setMember(boolean isMember);

    /**
     * @param members
     *            Vector with DAVResourceObjects
     */
    public void setMembers(Vector<IDAVResource> members);

    /**
     * @param the mime type of file resource
     */
    public void setMimeType(String newMimeType);

    /**
     * @param owner
     *            Owner of this resource
     */
    public void setOwner(IDAVRepository owner);

    /**
     * @param type
     *            the internal resource type (String)
     */
    public void setResourceType(String type);

}
```