Configuring Protege Install Folders (Non-NCI Users)

Configuring Protégé Install Folders (Non-NCI Users)

Contents of this Page

Overview

This page serves as an instructional guide as well as a brief history and and overview about Protege to all non-NCI users. Users are able to download a zipped copy of this public release Protege extension package, which contain the necessary installation folders. Within each folder, certain file(s) will need to be configured in order to properly run the application. Once configured in the user should be able to run Protege within their own server environment.

EVS Protege Extensions Introduction and History

Currently, version 1.3.0.188 is the third public release of the EVS Protege Extensions. There have been multiple revisions released, tested and used internally over the past year. The current version is considered stable and reliable and suitable for external release.

These extensions were created in response to a need for an open-source editing tool that could be customized to the needs of the EVS content editors. In addition, EVS wanted the ability to exchange and share editable content with external collaborators. The Protégé editing tool, developed by Stanford, was used as the basis and extended to meet these needs.

The extensions are built on the Protégé 3.4 Beta 508 codebase and utilize pellet 1.5. Some improvements to the Protégé software were required to meet the needs of EVS, but these have already been merged into the Protégé trunk. Both Protégé and Pellet are included within the download. The Explanation Server was developed in collaboration with Clark & Parsia LLC.

The Protégé and the extensions are built and expect to run on Java 1.5.

EVS Protégé Extensions

As metioned above, the EVS Protégé Extensions are broken into three separate packages. The Explanation server provides classification and explanation functionality. The Protégé server provides access to a central editing project for multiple users. The Protégé client is the end-user application for accessing the Protégé server and editing content.

Explanation Server

The Explanation Server is used to do classification of the ontology and provide explanations on demand to the Protégé Client GUI. It runs directly against the database, independently of the Protégé Server. The Protégé Server queries the Explanation Server for information when needed and controls requests for classification, so multiple clients cannot try and classify at the same time. The Explanation Server is designed to run in the bash shell on a Linux or Unix computer. Please ensure that the JRE is installed on the machine to run the explanation server.

Protégé Server

The Protégé server provides multi-user access to a single ontology stored in a database. It coordinates and controls user activities and resource allocation. It stores a history of user actions for use in tracking changes and resolving conflicts. The server also provides a centralized means of enforcing business rules and configurations upon client applications. EVS has extended the Protégé server to allow tracking of workflow and assigning of editing tasks. The Protégé server is designed to run in the bash shell on a Linux or Unix computer. Please ensure that the JRE is installed on the machine to run the explanation server.

Protégé Client

The Protégé client is a java-swing based GUI used for editing an ontology. EVS uses the client to connect to the Protégé server application, allowing multiple editors to share the same ontology. The Protégé client can also be used in standalone mode to edit a single ontology but some of the client-server specific extensions are then disabled. The client is used in standalone mode by managers to perform Prompt comparisons and terminology exports. The Protégé client is designed to run in the bash shell on a Linux or Unix computer or on a Windows computer. The JRE has been set for the Protégé client to run on a Windows machine, however, this JRE folder is not needed for the client residing on the server, the Linux server should have a JRE setup.

Installation instructions

The instructions below will allow the user to build an environment to match the one used by EVS. Some of the configurations are optional and will be labeled as such. The zip package contains folders for the client, Protégé server and explanation server. All of these should be copied to the Linux/Unix machine where you will be serving Protégé. A copy of the client should also reside on a local Windows machine for use in accessing the server data.

Metaproject

Any multi-user Protégé server-client configuration requires a metaproject. The server application references this metaproject to determine what ontologies to make available for editing and what users have rights to do various tasks. EVS has made available an example metaproject that includes all the privileges and groups required for the NCI edit tab. A base metaproject is located in the Protégé.Server-1.4.0.267/examples/server folder.

Protégé Client Folder

Before the vocabulary can be accessed by the server it must be loaded into a project using the Protégé Client GUI. The OWL source is loaded first as a Protégé file project and then converted to a database project. Steps to load a new Protégé project and convert in to a database project are below. The Protégé application will create the necessary tables in the database. You, or course, need to have a database ready and able to be written to before you start the load.

These steps can be performed on a Windows machine but all the files generated or modified would then need to be copied up to the Linux server. The directions below assume the user is running the client on the actual Linux server.

Files to configure in the Protégé client folder:

run_protege.sh (Linux)

1. JAVA_HOME = must be set to the proper path. Insert the path to your JDK folder here (NCI Protege uses Java 1.5).
2. For the client folder residing on the Linux server, set MAX_MEMORY value to a suitable size (NCI Prod - 7GB, QA - 7GB, Dev 2.5GB).

run_protege.bat (Windows)

Used only for windows machines. All that needs to be done here is to set MAX_MEMORY to 700MB. An important note is that if you are using the client from a Windows machine, be sure to insert a copy of a JRE folder within the Protégé client folder. The batch file will not run unless it can detect a JRE.

PromptNCIPlugins.properties

Examine the PromptNCIPlugins.properties file to verify correct database information. This is only necessary if Prompt is going to be used to export history records to the concept_history table:

nci.prompt.ipAddress=<hostname>:<port>
nci.prompt.dataBaseName=<db name>
nci.prompt.dbEVSHistoryTable=<tablename>
nci.prompt.userName=<username>
nci.prompt.password=<password>

Explanation Server Folder

The Explanation Server can connect with either the Protégé database or to a raw OWL file. Files to configure:

start_explanation_server.sh

1. Allocate a decent max_heap_size. (default is set to 12gigs, however, less memory can be used for test machines (i.e. 8GB for QA)).
2. Enusre that the explanation server is pointing to the correct Protege.Server folder: (protege_install=/path/to/new/Protege server folder). This allows the Explanation server to use jars found in the Protege.Server folder.
3. Make sure the path to the java server is correct (replace <PATH TO JRE BIN> with the actual jre/bin/java.exe location.
4. The command to connect the explanation server to a database project: **'./start_explanation_server.sh --port <PORT> --protege-standalone /path/to/databaseProject.pprj**'. The command to connect to a raw OWL file: *'./sh start_explanation_server.sh --urls file:///home/user/path/to/file.owl'*
5. The Explanation Server is ready when you see "Jena, Classification, and Extracting' as well as a final 'Server started, listening on port ....' message. The Explanation Server should be allowed to start completely before starting the Protege Server.

Protégé Server Folder

run_rmiregistry.sh

1. JAVA_HOME must be set in the script.
2. The very last line has a port number that will need to match the Protégé Server: $JAVA_PATH/rmiregistry **<PORT>** &

run_protege_server.sh

1. JAVA_HOME must be set.
2. Set MAX_MEMORY to a suitable size to run adequately (i.e. NCI Production - 7.5 GB, QA - 7GB, Dev - 3GB).
3. Ensure RMI_REG_PORT number matches port number in 'run_rmiregistry.sh' script.
4. Set RMI_SERV_PORT to any desired open port.
5. HOSTNAME_PARAM=-Djava.rmi.server.hostname=<hostname> should be edited with the actual server hostname where the rmi registry server will be running. The "localhost" address will not work properly, it needs to be the actual server name.
6. Set METAPROJECT with the location of the Metaproject file.

codegen.properties

The codegen.seed number should be set at a figure higher than the highest concept number in your vocabulary. The codegen prefix and codegen.suffix values are optional. The optional codegen delimeter will set the character to be inserted between prefix and code or code and suffix. Example: prefix=B, delimiter=-, suffix=cs, seed=100. The first code generated will be B-100-cs. A 'codegen.dat' file will then be generated within the Protégé server folder and will be updated to "100" and will iterate the concept code number from that point.

PromptNCIPlugin.properties

Verify correct database information for saving to the evs_history table. This information should match the PromptNCIplugin info residing in the Protégé client folder:

nci.prompt.ipAddress=<hostname>:<port>
nci.prompt.dataBaseName=<db name>
nci.prompt.dbEVSHistoryTable=<tablename>
nci.prompt.userName=<username>
nci.prompt.password=<password>

protege.properties

Verify that the protege.properties file points to the correct Protégé server as well as the URL for the explanation server:

edu.stanford.smi.protege.server.ServerPanel.host_name=serverhostname\:port
edu.stanford.smi.protegex.owl.jena.reasoner.URL=http\://serverhostname\:port/explain/

The first line should be the Protégé server hostname with the desired RMI port number. The second line is read by the explanation server, and should be the same address or hostname as the Protégé server. However, the port number will be different. The backslashes in the hostname represent escape character in Linux for the colons.

To edit, use your editor to open the properties file. Insert proper server hostname and port number values, save, and exit the properties file. Nothing else should be modified in this file.

Protégé Database Project/Configuration Files

All files should now be configured and ready to be used. At this point, a new database project will need to be created and configured for the Protégé server to connect with the database. Follow the Instructions for creating and configuring a new Protégé database project.

Loading the OWL file into a Project

Open the Protégé client, run 'run_protege.sh', and click on "Create New Project". Select OWL/RDF File, and check "Create from existing source". Click "Next". A window will display requesting the address of the source OWL file. Navigate to where the OWL file is stored and click "Finish". The OWL file may take some time to load into the client, dpending on the file size.

Converting a File-Based Project into a Database Project

Once the OWL file is visible in the GUI, it can now be converted to an OWL/RDF database. Click "File", "Convert Project to Format", and select OWL / RDF database. A new window will open requesting information on the database where the project should be created. Even a database project has a file component that holds settings for the project. The first field in the new window will request the path to the project file.

The remaining fields are for database info:
*JDBC Driver Class Name - for MySQL, the database used by NCI, this would be com.mysql.jdbc.Driver
*JDBC Driver URL - the URL of the database. In MySQL it will look like jdbc:mysql:<server>:<port>/<database>
*Table - the name of the table to be created for this project
*Username - username for access to the database
*Password - password to the database

After entering, click "Finish" and the client will the proceed to load the data into the specified database table. This process may take some time depending on the file size.The console will display a completion message indicating that the conversion process is successful. The database project should be visible in the client. The project is now ready to be configured.

Configuring the Database Project

Activating the NCIEdit Tab

Click "Project" on the menu bar, and select "Configure". On the Tab Widgets tab, enable the the NcIEdit tab.

<Optional>Within the same dialog, click on the Options tab and enable journaling and disable redo/undo. Click "OK" to finish.

Changing the Tree Display

To set the display slots, click the NCIEdit tab, and click the "F" tab on the very far top right section of the page. You will now be within the Form Editor section. Set the "Display Slot" drop down to the property you wish displayed for the concept (NCI Thesaurus uses "rdfs:label".

Changing the display of properties

Click the Properties tab, and click the "F" tab on the very right. On the left hand menu, click on "owl:DatatypeProperty", and set it's "Display Slot" drop down to "rdfs:label". Next, click on the owl:ObjectProperty on the left hand menu, and set its "Display Slot" drop down field to 'rdfs:label' as well.

OWL Preferences Window

Click "OWL" on the menu and select "Preferences". On the "General" tab set the explanation server URL to the url where the explanation server will be located. The <hostname> is typically "localhost" and the <port> is set by the user when starting the Explanation server. Example: http://localhost:1234 /explain/ <Optional>On this same page disable Drag and Drop. It can cause accidental re-treeing of concepts due to errant drags.

<Optional>In the "Tests" tab disable "List deprecated classes and properties". This will prevent testing of deprecated classes which can save a lot of time if there are many of them.

<Optional>In the "Visibility" tab disable all the boxes in the MetaClasses section. This can eliminate a lot of clutter in the editing view.

Click "Close" at the bottom of the dialog box.

Activating Other Essential Tabs

Click "Project" on the menu bar, and select "Configure". On the Tab Widgets tab, enable the following tabs, and order as shown:

- NCIEditTab
- LuceneQueryTab
- OWLPropertiesTab
- OWLMetadataTab
- ChangesTab
- ExplanationTab
- NCIWorkflowTab

Click "OK" to exit. You will notice a window that will automatically display prompting you to choose a source to store all changes and annotations ontology. Changes are records of any type of modification that is done to any concepts within the ontology.

The window will prompt you to select whether to save to a new or existing changes ontology file, or to a database. After selecting, follow the wizard to properly configure the changes ontology source.

Indexing the Ontology

The Lucene Query Tab (LQT) works off of a lucene index. To enable this, click on "Lucene" from the menu bar, and select "Index Ontology". A window will appear with a list of all the properties. By default, all properties are selected, and you may customize indexing by selecting the properties individually as well. Click 'Start', and you will notice a small status window at the top left hand of your screen. Once indexing has completed (Indexing usually completes when the status window disappears), a directory called "lucene" will have been created within the same directory as your new database project. It is essential to keep the newly created lucene directory at this location.

<Optional - but advised> Click on the "Properties" tab and click on the "Annotation" sub tab within the menu pane.
Click "code" in the Annotation Properties list. You should see the properties listed for code' within the "Property Editor" section of the GUI. Enusre that the property 'protégé:readOnly" is labeled as 'true'. This will prevent editors from changing the identifier for a concept once it is assigned.

Saving the New Database Project

Click "File", 'Save Project As.' Name the project and save to a directory on the server.
This act of saving will create an Annotations project in the same directory as the main project. Close the project but leave the Protégé client open.

Setting up the Metaproject security

The sample Metaproject should be found in the Protege.Server-1.2.3.25/examples/server folder. Open the nci_metaproject.pprj provided. Click on the "Instances" tab and select "Project" in the list to the left.

Select "Ontology" from the "Instance Browser" in the middle and click the "View Ontology" button just above (It looks like a magnifying glass). In the window that opens set the name of your ontology as you would like it to appear when people access it from the server. Just below that set the absolute path to the project file created in 3.3.3.7. Close just this subwindow using the red "X" in the upper right of the subwindow.

Now click on the Annotation_Ontology in the "Instance_Browser", name it and enter the path. Close the subwindow.

Click on "User" in the list to the left. Examine the existing users by clicking on them. Make a copy of the _test_editor user and give it a new name and password. Make note of this information for use in logging into the Protégé Server later.

Click "File", "Save Project" then "Exit".

'ncitab.xml' File

The Protégé Client also contains an ncitab.xml file which is used to populate drop down menus with NCI-specific values in the Protégé Client also provides other ontology-specific configurations. Towards the bottom of the file, curatorial authorities can be assigned to specific editors.

For example:

<authorization prop="curatorial-authority">
<entry>
<authority>enter_authority_here</authority>
<username>enter_username_here</username>
</entry>

Selected editors can be authrorized to handle specific concepts within the project.

NCIHeader.owl File

This file is optional, but can be copied over to your OWL file to provide the required NCI properties and concepts.
The codes can be adjusted as necessary, as long as it coincides with the ncitab.xml file.