Tree Pruner
Technical Spec: Files, Applet Interface and Communications
Latest Draft: September 04, 2009

Overview

Tree Pruner applet, based on the LGPL Archaeopteryx "Archae" applet (http://www.phylosoft.org/archaeopteryx/), will interact with the HTTP server retrieve and store the attributes using JSON with REST.

Launching the JAR from HTML

TreePrunerArchaeUnsigned.jar – for app_type 3 or signed version of TreePrunerArchaeUnsigned.jar – for app_type 1 or 2 take in following parameters and can be launched by using the following code snippet:

…
*<Applet ARCHIVE= "*TreePrunerArchaeUnsigned*.jar" CODE="org.forester.archaeopteryx.ArchaeopteryxA.class" WIDTH=600 HEIGHT=500>*
<param name="url_of_tree_to_load" value="<relative path of newick file to load> "/>`
<param name="config_file" value="<relative path to _tp_atv_configuration_file.txt>"/>
<param name="filename" value="<Filename with which the accessions to be removed will be saved and retrieved from the server>"/>
<param name="URLPrefix" value="<absolute path of URL prefix with which the accessions to be removed will be saved and retrieved from the serve> "/>
<param name="app_type" value="<Please see application type tables to choose your app_type >"/>
<param name="saved_acc_flag" value="<true or false>"/> <-- true: if accessions were saved before; false if there is nothing saved on the server-->
<param name ="tree_panel_tab_name" value="<This string appears on the tab of the tree panel of the Tree Pruner – Archae applet>"/>

<param name ="remote_user" value="<remote username which is sent back to the server in JSON to validate the user on the server. See application type tables for more details >"/> <!—for app_type 2 only-->

</APPLET>
…

You can find more information in example applet files for the respective app_type. Please see FilesAndApplicationTypeDocu.doc or FilesAndApplicationTypeDocu.pdf in the TreeArchaeSource/Archaeopteryx/INSTALL/ FilesAndApplicationTypeDocumentation/ folder for more details.

Newick File Format

Tree Pruner – Archae uses newick file format with the accessions of the sequences to display the tree on the applet. The accessions are used to uniquely distinguish the sequences from each other and are also passed back to the server to perform the pruning action on the server. Example format of the newick file accepted by Tree Pruner – Archae is stored in TreeArchaeSource/Archaeopteryx/INSTALL/ExampleFiles/ TreePrunerAllAppTypes/Example_newick_file.txt

JSON File Format

You can find example Tree Pruner JSON files for the respective app_type. Please see FilesAndApplicationTypeDocu.doc or FilesAndApplicationTypeDocu.pdf in the TreeArchaeSource/Archaeopteryx/INSTALL/ FilesAndApplicationTypeDocumentation/ folder for more details

Communication

Tree Pruner – Archae uses REST as architecture for communicating the above said file format back to the server and to retrieve these accessions back from the server. It uses the HTTP GET and POST request types to distinguish between getting the saved JSON string from server and sending the JSON string to server

For performing POST Tree Pruner - Archae uses the following URL, if URLPrefix passed as applet param is http://domain.com/treePruner : http://domain.com/treePruner/id.
For performing GET Tree Pruner - Archae uses the following URL, if URLPrefix passed as applet param is http://domain.com/treePruner and filename passed as applet param is 1234_56_789
http://domain.com/treePruner/id/1234_56_789

Below are the various parameters used by Tree Pruner – Archae to perform the http GET and POST methods with Java code snippets. The entire code can be found in com.lanl.application.treePruner.applet.TreePrunerCommunication for POST and in com.lanl.application.treePruner.applet.CrashRecovery for GET in src/

POST: AutoSave / Save, Commit, Discard
In order to pass the JSON file from client to server and perform autosave, save, commit and discard operations, Tree Pruner – Archae uses the HTTP POST method. The save, commit and discard actions are performed when the user clicks the Save, Commit changes; finish session and Discard all buttons respectively on the control

panel of the applet. AutoSave is performed every 10 minutes if the user has pruned the tree on the applet and left it unsaved.

Request from Client to Server

```
...
static String TPpostURL = AppletParams.URLprefix + "/id";
...
postURL  = new URL(TPpostURL);
HttpURLConnection postConn=(HttpURLConnection)postURL.openConnection();
postConn.addRequestProperty("Content-Type","text/JSON" );
postConn.setRequestMethod("POST");
postConn.setDoOutput(true);
OutputStreamWriter wr = new OutputStreamWriter(postConn.getOutputStream());
...
wr.write(JSONString);
wr.flush();
...
```

CASE: AutoSave/Save, Commit and Discard

In addition to the accessions to remove present under Sequence Accession to Remove key (only for AutoSave/Save and Commit) in the JSON file, this file has two or three additional keys depending upon the application type selected (*Please see application type tables for more details).* These additional keys are used to pass additional information from the client to the server.

For app_type 1 and 3:
"Action": "<Save or Commit or Discard>" (used to send back what action is being currently performed)
"Filename": "<Filename passed in as applet parameter>" (used to send back the filename under which the accessions to remove will be saved or committed or discarded)

For app_type 2:
"Action": "<Save or Commit or Discard>" (used to send back what action is being currently performed)
"Filename": "<Filename passed in as applet parameter>" (used to send back the filename under which the accessions to remove will be saved or committed or discarded)
"remote_user" : "<remote user passed in as applet parameter>" (used to send back the user who launched the applet and to verify whether his session is still active on the server before performing the POST)

You can find example Tree Pruner JSON files for the respective app_type. Please see FilesAndApplicationTypeDocu.doc or FilesAndApplicationTypeDocu.pdf in the TreeArchaeSource/Archaeopteryx/INSTALL/ FilesAndApplicationTypeDocumentation/ folder for more details.

Response from Server to Client

```
...
BufferedReader rd;
Rd = new BufferedReader (new InputStreamReader (postConn.getInputStream()));
String line;
while ((line = rd.readLine()) != null) {
        returnedString += line;
}
wr.close();
rd.close();
...
```

Case 1: Commit

On receiving the packet with HTTP POST, the server performs the necessary operations based on the Action key of JSON. Then the server must respond with HTTP 200 OK or 201 Created status code along with a success message in the payload of the response, which will be simply a string that reads as "Accessions successfully removed from DB". On receiving this, the applet will open a JOptionPane to inform the user that the task was successfully performed on the server. Failing to receive this string the applet will open a JoptionPane to tell the user that the action failed

Case 2: AutoSave / Save

On receiving the packet with HTTP POST, the server performs the necessary operations based on the Action key of the JSON. Then the server must respond with HTTP 200 OK or 201 Created status code along with a success message in the payload of the response, which will be simply a string that reads as "Accessions saved successfully". On receiving this, the applet will dump a statement saying that the task was successfully performed on the server. Failing to receive this the applet dump a statement saying that the action failed.

NOTE: The set on the server may need to be locked to avoid the saved accessions on the server from becoming stale. If you want to have the applet to tell the server when the applet is fired up and when the applet terminates to perform locking and unlocking of the set respectively, then please use app_type =1. Please read the next section for more details.

Case 3: Discard

On receiving the packet with HTTP POST, the server performs the necessary operations based on the Action key of the JSON. Then the server must respond with HTTP 200 OK or 201 Created status code along with a success message in the payload of the response, which will be simply a string that reads as "Accessions successfully discarded". On receiving this, the applet will dump a statement saying that the task was successfully performed on the server. Failing to receive this the applet dump a statement saying that the action failed

And finally the connection is closed.
postConn.disconnect();

POST: Lock And Unlock Communication (for app_type = 1 only)

In order to inform the server that the applet has started so that the server can lock the working set and to inform the server that the applet has terminated so that the serer can unlock he working set, Tree Pruner – Archae uses the HTTP POST method. The lock operation is performed every time the applet is fired up. The unlock operation is performed every time the applet terminates.

Request from Client to Server

```
...
static String TPpostURL = AppletParams.URLprefix + "/id";
...
postURL  = new URL(TPpostURL);
HttpURLConnection postConn=(HttpURLConnection)postURL.openConnection();
postConn.addRequestProperty("Content-Type","text/JSON" );
postConn.setRequestMethod("POST");
postConn.setDoOutput(true);
OutputStreamWriter wr = new OutputStreamWriter(postConn.getOutputStream());
…
wr.write(JSONString);
wr.flush();
...
```

CASE: Lock/Unlock

The JSON file always has two keys. These keys are used to pass information regarding working set and action from the client to the server.

"Action": "<Lock or Unlock>" (used to send back what action is being currently performed)
"Filename": "<Filename passed in as applet parameter>" (used to send back the filename using which the server can lock or unlock a particular working set)

You can find an example of this Tree Pruner JSON file in TreeArchaeSource/Archaeopteryx/INSTALL/ExampleFiles/ TreePrunerAppType1/exampleLock.json for Lock and TreeArchaeSource/Archaeopteryx/INSTALL/ExampleFiles/ TreePrunerAppType1/exampleUnlock.json for Unlock.

Response from Server to Client

```
...
BufferedReader rd;
Rd = new BufferedReader (new InputStreamReader (postConn.getInputStream()));
```

```
String line;
while ((line = rd.readLine()) != null) {
        returnedString += line;
}
wr.close();
rd.close();
...
```

Case 1: Lock

On receiving the packet with HTTP POST, the server performs the necessary operations based on the Action key of the JSON. Then the server must respond with HTTP 200 OK or 201 Created status code along with a success message in the payload of the response, which will be simply a string that reads as "Working Set locked successfully". On receiving this, the applet will dump a statement saying that the task was successfully performed on the server. Failing to receive this the applet dump a statement saying that the action failed.

Case 2: Unlock

On receiving the packet with HTTP POST, the server performs the necessary operations based on the Action key of the JSON. Then the server must respond with HTTP 200 OK or 201 Created status code along with a success message in the payload of the response, which will be simply a string that reads as "Working Set unlocked successfully". On receiving this, the applet will dump a statement saying that the task was successfully performed on the server. Failing to receive this the applet dump a statement saying that the action failed

And finally the connection is closed.

```
postConn.disconnect();
```

GET: Crash Recovery / Resuming an existing Session

In order to get the JSON file from the server and perform crash recovery or resume an existing session of the Pruner, Tree Pruner – Archae uses the HTTP GET method.

Request from Client to Server

```
...
String TPgetURL = AppletParams.URLprefix + "/id/"+AppletParams.filename;
savedAccURL = new URL(TPgetURL);
HttpURLConnection crashRecConn = (HttpURLConnection) savedAccURL.openConnection();
crashRecConn.setRequestMethod("GET");
...
```

The client sets the request method as GET and tries to connect to the GET URL with an empty payload. This action is performed only when the saved_acc_flag is set to true in applet parameters telling the applet that there is saved session from the past on the server.

Response from Server to Client

```
...
BufferedReader rd;
crashRecConn.setDoInput (true);
rd = new BufferedReader(new InputStreamReader(crashRecConn.getInputStream()));
String line;
while ((line = rd.readLine()) != null) {
        TPgetJSON += line;
}
rd.close();
...
```

On receiving the GET request, the server uses the filename in the URL to fetch the JSON format string on the server then responds with a packet that has HTTP 200 OK or 201 Created status code, the content type of the packet - text/JSON and JSON string saved on the server as the payload.

And finally the connection is closed.

```
crashRecConn.disconnect();
```

If something is wrong with the JSON format in the string got back from the payload of the response, JSONTokener will throw a JSONException on the client.

Enjoy the product!