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

Overview

Tree Decorator 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

TreeDecoratorArchaeUnsigned.jar – for app_type 6 or signed version of TreeDecoratorArchaeUnsigned.jar – for app_type 4 or 5 take in following parameters and can be launched by using the following code snippet:

…

<Applet ARCHIVE=" TreeDecoratorArchaeUnsigned.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 _td_atv_configuration_file.txt>"/>
<param name="filename" value="<Filename with which the decorations will be saved and retrieved from the server>"/>
<param name="URLPrefix" value="<absolute path of URL prefix with which the decorations 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 decorations 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 Decorator – Archae applet>"/>

<param name ="remote_user" value="<remote username which is sent back to the server in the JSON string to validate the user on the server. See application type tables for more details >"/> <!—for app_type 5 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 Decorator – 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 used to perform the decorating action on the client. Example format of the newick file accepted by Tree Decorator – Archae is stored in TreeArchaeSource/Archaeopteryx/INSTALL/ExampleFiles/ TreeDecoratorAllAppTypes/Example_newick_file.txt

JSON File Format

You can find example Tree Decorator 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 Decorator – Archae uses REST as architecture for communicating between the server and client. It uses the HTTP GET and POST request types to distinguish between getting the saved JSON string and sequence details JSON string from the server and sending the JSON string to server for save and discard.

For performing POST for saving/discarding the decorations on the server, Tree Decorator - Archae uses the following URL, if URLPrefix passed as applet param is http://domain.com/treeDecorator :
http://domain.com/treeDecorator/id.

For performing GET of the sequence details, Tree Decorator - Archae uses the following URL, if URLPrefix passed as applet param is http://domain.com/treeDecorator and filename passed as applet param is 1234_56_789
http://domain.com/treeDecorator/sd/1234_56_789

For performing GET of previously saved decorations, Tree Decorator - Archae uses the following URL, if URLPrefix passed as applet param is http://domain.com/treeDecorator and filename passed as applet param is 1234_56_789
http://domain.com/treeDecorator/id/1234_56_789

Below are the various parameters used by Tree Decorator – Archae to perform the HTTP GET and POST methods with Java code snippets. The entire code can be found in com.lanl.application.treeDecorator.applet.communication package in src/

POST: AutoSave / Save, Discard

In order to pass the JSON file from client to server and perform autosave, save, and discard operations, Tree Decorator – Archae uses the HTTP POST method. The save and discard actions are performed when the user clicks the Save and Discard all buttons on the control panel of the applet. AutoSave is performed every 10 minutes if the user has made some decorations on the applet and left it unsaved.

Request from Client to Server

```
...
static String postURLString = AppletParams.URLprefix+"/id";
...
postURL = new URL(postURLString);
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(JSONStringToPass);
wr.flush();
...
```

CASE : AutoSave/Save and Discard

In addition to the decorations performed on the applet present under the Node and Label keys (only for AutoSave/Save) 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 addition information from the client to the server.

For app_type 4 and 6:

"Action": "<Save 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 decorations will be saved or discarded)

For app_type 5:

"Action": "<Save 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 decorations will be saved 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 Decorator 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: 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 "Decorations 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.

Case 2: 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 "Decorations 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();
```

GET: Crash Recovery / Resuming an existing Session

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

Request from Client to Server

```
...
crashRecoveryURLString = AppletParams.URLprefix + "/id/" +
        AppletParams.filename;
...
crashRecoveryURL= new URL(crashRecoveryURLString);
HttpURLConnection crashRecoveryConn =
        (HttpURLConnection)crashRecoveryURL.openConnection();
crashRecoveryConn.setRequestMethod("GET");
...
```

The client sets the request method as GET and tries to connect to the GET URL for saved decorations 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;
crashRecoveryConn.setDoInput (true);
rd = new BufferedReader(new InputStreamReader(crashRecoveryConn.getInputStream()));
String line;
while ((line = rd.readLine()) != null) {
        savedDecorationsJSONString += 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. 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.

```
crashRecoveryConn.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.

GET: Sequence Details

In order to get the sequence details JSON string from the server Tree Decorator – Archae uses the HTTP GET method. These are the details using which the user can decorate his tree on the applet.

- JSON file to be generated on the server to be sent to the client

The JSON file that should be generated on the server consists of 2 keys.
The first key is the flu_type, which will specify the flu_type to be A, B or C. The applet makes the A/HA subtype and A/NA Subtype options of decoration enabled only for flu_type A in the Decoration Selections frame and Legend frame.
The second key is the accession itself. There can be 1 or more accession keys in this file. These accessions will map with the leaf nodes of the tree on the applet. This enables the applet to decide the node with this accession should be decorated with which decoration style based on the characteristic passed in through this key.
Snippet of the JSON file used to send the sequence details to the applet is shown below:

```
{"Sequence Details":{
    "flu_type":"A",
    "ISDN334864":{
      "Year":"2010",
      "Country":null,
      "HA":"1",
      "Host":"Avian",
      "NA":"1"
    }}}
```

Example format of the JSON file accepted by Tree Decorator – Archae is stored in TreeArchaeSource/Archaeopteryx/INSTALL/ExampleFiles/ TreeDecoratorAllAppTypes/ exampleSequenceDetails.json

Request from Client to Server

```
...
seqDetailsURLString  = AppletParams.URLprefix + "/sd/"+AppletParams.filename;
...
seqDetailsURL= new URL(seqDetailsURLString);
HttpURLConnection seqDetailsConn = (HttpURLConnection)
      seqDetailsURL.openConnection();
seqDetailsConn.setRequestMethod("GET");
...
```

The client sets the request method as GET and tries to connect to the GET URL for sequence details with an empty payload. This action is performed every time as soon as the applet is fired.

Response from Server to Client

```
...
BufferedReader rd;
```

```
seqDetailsConn.setDoInput (true);
rd = new BufferedReader(new InputStreamReader(seqDetailsConn.getInputStream()));
String line;
while ((line = rd.readLine()) != null) {
        seqDetailsJSONString += 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. 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 sequence details JSON string as the payload.

And finally the connection is closed.

```
seqDetailsConn.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. Also since without the sequence details, the tree decorator will not be able to perform decoration, if there is a communication failure or a problem with the JSON string format, the applet will throw an error and terminate.

Enjoy the product!