**EverNote-Implementation**

Entire implementation is done with sandbox account of evernote and converted into production account.

**BOT details:**

* Bot credentials used in class 'EverNote'

id **:** evernote@webaroo.com

password : \*\*\*\*\*\*\*\*\*

* Following Evernote api details are used in class 'EverNoteServlet'

Consumer Key : \*\*\*\*\*\*\*\*

Consumer Secret : \*\*\*\*\*\*\*\*\*

**Oauth:**

* visit <https://dev.evernote.com/doc/articles/authentication.php> and click the GET AN API KEY button on top right corner to get api-key and secret.
* Using this we can handle Evernote's sand box accouns.
* To make it global, we have to move it on to production server by requesting to activate the API KEY at <https://dev.evernote.com/support/> and click on 'Activate an Api key' button.
* At present, Evernote is using Oauth 1.0
* Used sribE api to make Oauth.
* Used database to store the users Oauth details(Teamchat mailID, access token).
* Intially, user will be redirected from Bot to EverNoteServlet where Authorization URL is called.
* After user authorizes, it'll be re-directed to callback url.
* From the call back URL o*auth\_verifier* is extracted and traded for accesstoken along with requesttoken.
* The obtained accesstoken is stored in database

**Evernote Structure:**

NoteStore --> Notebook -->Note

* Ever user will have a NoteStore(like root of the user's library).
* In that NoteStore we can create Notebooks.
* In those Notebooks we can create Notes.

**Keywords included:**

1. help – It'll displaythe rest of the keywords.
2. connect – It'll connect Teamchat with Evernote. It'll make Oauth.
3. disconnect – It'll delete user's details(mail-id,accessToken) from database
4. myevernote – It'll display a primary chatlet with a dropdown list of all the available features.

**Evernote features that are included:**

1. Create Notebook
2. Create Note
3. List Notebooks
4. List Notes
5. Search Notes
6. Edit Note
7. List Reminders

**API related to above features:**

1. Create Notebook

* Requesting user to specify a name for notebook.
* No two notebooks can have same name.
* After creating Notebook instance, set name for it as requested by user.
* In noteStore, there will be a method called 'createNotebook'.

1. Create Note

* Requesting the user to mention the following details.
* In which notebook he's willing to create the note.(If no notebook is selected then it'll be created in default notebook)
* Title of the note. If he didn't mention anything then it'll be titled as “Untitled”.
* Content of the note.
* Set some tags if he wish.(More than one tag are allowed by using seperator '-')
* More than one Note can have same name.
* Evernote stores its notes in XHTML format.
* Class named 'NoteContent' will play key role in creating Note.
* Content of the note provided by user will be induced into predefined XML format.
* In Note,
  + setNotebookGuid(String) is used to set the desired notebook in which we are intended to create note.
  + setContent(String) is used to set the content.
  + addToTagNames(String) is used to set tag names.
* In noteStore,
  + createNote(note) is used to create the note as all the attributes are already set.

1. List Notebooks

* In noteStore, listNotebooks() provides the list of Notebooks present in that noteStore.

1. List Notes

* In noteStore, listNotebooks() method provides a List of all notebooks.
* In notebook, findNotes() will provide the notes present in that particular notebook.
* In noteStore, getNote() will provide a note with more details like content.
* In that note, getContent() will provide the content in XML format.
  + So, method named 'extract' is created in 'NoteContent' class to parse the XML content and get the actual content of the note.

1. Search Notes

* Requesting the user to enter the query he want to search.
* User can search either only in tags or the entire notes.
* To search only in tags, he's supposed to give “tag:query”.
* Firstly, we are creating a NoteFilter that filters all the notes in the NoteStore.
* In that filter,
  + setWords(String) will make the filter to filter all the notes with that query.
* In noteStore,
  + findNotes() will make a list of all the notes with given filter attributes.
  + To extract content from each note, 'extract' method in 'NoteContent' class is used to get plain text.

1. Edit Note

* User have to select the notebook from the set of notebooks present.
* User have to select the note which he wants to edit.
* User will type the new content, so that it'll override the previous content of the note.

1. List Reminders

* It'll display,
  + Notebook name.
  + Note's title
  + Reminder date&time
* In a note,
  + getAtrributes() will provide the attributes.
  + In getAttributes,
    - getReminderTime() will provide the time at which it is supposed to remind.