ezEML Developer Notes- User Data

User data is stored on an ezEML server in directory `/home/pasta/ezeml/user-data`.

This directory contains a subdirectory for each user. The subdirectory's name is of the form

```
EDI-1a438b985e1824a5aa709daa1b6e12d2
```

where "EDI" in this example is the username and the hex value following the hyphen is a hash of the user's UID, which in this case is

```
uid=EDI,o=EDI,dc=edirepository,dc=org
```

See `auth.user.get_user_org()` and `auth.user_data.get_user_folder_name()` for details.

The purpose of the hash is to disambiguate cases where there are multiple accounts under the same username, typically because a given user has authenticated by different means at different times – e.g., by using Google credentials and GitHub credentials. These are considered distinct accounts.

In addition, `user-data` contains the following:

`__collaboration_backups`
: Backups created when submitting a package to EDI's curators, stored under the username plus hash as described above.

`__db`
: Contains the sqlite database file used to manage collaboration status information.

`ezEML_GC.log`
: A log file for ezEML garbage collection.

`reset_check_data_table_statuses.py`
: A utility to be run in the rare event that the Check Data Tables criteria change. It resets all of the data tables' statuses to "unchecked" so the checks will be run again.

Contents of a User's User-Data Directory

A user's user-data directory contains, for each data package in that user's account, an XML file and a JSON file containing the model's metadata, and a Pickle file containing the results of Check Metadata on that model. The latter is created whenever the model changes, so that the Check Metadata status can be current without having to recompute for every page access.

In addition, a user's directory contains the following:

`uploads`
: A directory with a subdirectory for each data package. The subdirectory contains the uploaded CSV files and other data entities.

For each uploaded data table, if Check Data Table has been run on the table, there is a file containing the results from Check Data Table. This results file is named as follows: if the data table file is `foo.csv`, the results file will have a name of the form `foo.csv_eval_8bfaf54411`, where `8bfaf54411` (for example; the value varies) is a hash of the data table's metadata. When we need the Check Data Table results, we see whether the eval file is present and whether the hash still matches our current metadata for the table. If so, we can use the file's contents without having to re-run the data table check, which is a potentially time-consuming task. If Check Data Table returned no errors, the file would be named `foo.csv_eval_8bfaf54411_ok`, where the `ok` lets us know there are no errors without our having to open the file and look inside.

`exports`

When Export ezEML Data Package is executed, ezEML saves the zip file that's created in a subdirectory of `exports`, with the package name as subdirectory name. Within the subdirectory, each version of the exported package is saved under a subdirectory named by the date and time at which the export was done. A URL provided to the user can then point to this zip file on the server, allowing it to be downloaded.

`backups`

In certain circumstances (for example, when cloning column properties) we make a backup of the metadata JSON file under a timestamped filename. This is available purely for troubleshooting.

`__uploaded_table_properties__.pkl`

A Pickle file containing a dictionary of properties of the uploaded data tables, such as the columns' variable types, the column names, and the categorical codes. This is used when doing Reupload, for example, to see if the properties have changed since the last upload.

`__user_properties__.json`

A JSON file containing various properties of the user and the user's currently active model. For example, whether the user needs to be shown the Welcome to ezEML dialog, whether the current model contains complex texttypes, what data tables have been uploaded, etc. Set a breakpoint on the return from `auth.user_data.get_user_properties()` to see its contents.

`active.pkl`

A Pickle file containing a dictionary of various properties of the active document, such as the owner, owner_login, and filename. See `auth.user.read_active_dict()`.