Class OpenBISScreeningML

```
java.lang.Object
  └OpenBISScreeningML
```

```
public class OpenBISScreeningML
extends java.lang.Object
```

Simple Matlab interface for openBIS for Screening. It is meant to be used in one Matlab session at a time, i.e. it is *not* multi-threading safe.

While written in Java, the API is idiomatic for Matlab, i.e. values are returned as multi-dimensional arrays. For the `get...` and `load...` methods the first index will contain the actual data, while the second index will contain per-row annotations. For `getFeatureMatrix`, the third index contains per-column annotations. This allows simple access with Matlab's slicing operator, see doc of e.g. `getFeatureMatrix(String)`.

A typical Matlab session looks like:

```
% Add the API jar file to the classpath
javaaddpath('/home/brinn/matlab/openbis_screening_api-batteries_included.jar')
% Login to server
OpenBISScreeningML.login('user', 'secret', 'https://www.infectome.org')

% ...perform calls on the server...

% Logout to close the session on the server
OpenBISScreeningML.logout()
```

Note: using this login your password will end up in the Matlab command history. An alternative that avoids this is to call the `Login` class. Logging in on the console will grant this class access to the openBIS server.

Author:
Bernd Rinn

Field Summary

`static java.lang.String`	**`REQUIRES_OPENBIS_AS_API`** The required version ("major.minor") of the screening API on the openBIS application server.
`static java.lang.String`	**`REQUIRES_OPENBIS_DSS_API`** The required version ("major.minor") of the screening API on the openBIS datastore server.
`static java.lang.String`	**`VERSION`** The version of the API.

Method Summary

`static java.lang.Object[][][]`	**`getFeatureMatrix`**`(java.lang.String gene)` Returns the feature matrix of all features for all locations (a location is one well position in one feature vector data set) connected to *gene* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.
`static java.lang.Object[][][]`	**`getFeatureMatrix`**`(java.lang.String experiment, java.lang.String gene)` Returns the feature matrix of all features for all locations in *experiment* (a location is one well position in one feature vector data set) connected to *gene* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.
`static java.lang.Object[][][]`	**`getFeatureMatrix`**`(java.lang.String gene, java.lang.String[] features)` Returns the feature matrix of the specified features for all locations (a location is one well position in one feature vector data set) in *experiment* connected to *gene* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.
`static java.lang.Object[][][]`	**`getFeatureMatrix`**`(java.lang.String experiment, java.lang.String gene, java.lang.String[] features)` Returns the feature matrix of the specified features for all locations in *experiment* (a location is one well position in one feature vector data set) in *experiment* connected to *gene* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.
`static java.lang.Object[][][]`	**`getFeatureMatrixForPlate`**`(java.lang.String plate)` Returns the feature matrix of all available features for all locations (a location is one well position in one feature vector data set) of all feature vector data sets of the given *plate* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.
`static java.lang.Object[][][]`	**`getFeatureMatrixForPlate`**`(java.lang.String plate, java.lang.String[] features)` Returns the feature matrix of the specified features for all locations (a location is one well position in one feature vector data set) of all feature vector data sets of the given *plate* in `[0]`,

	location annotations in `[1]` and feature annotation in `[2]`.
static java.lang.Object[][][]	**getGeneMappingForPlates**(java.lang.String[] platesCodes) Returns the gene mapping for the given *plateCodes* in `[0]` and location annotations in `[1]`.
static java.lang.Object[][]	**listChannels**(java.lang.String experiment) Lists all channels measured in *experiment*.
static java.lang.Object[][]	**listExperiments**() Lists all experiment.
static java.lang.Object[][]	**listFeatures**(java.lang.String experiment) Lists all features computed for *experiment*.
static java.lang.Object[][]	**listPlates**() Lists all plates.
static java.lang.Object[][]	**listPlates**(java.lang.String experiment) Lists the plates of *experiment*.
static java.lang.Object[][][]	**loadImages**(java.lang.String plate, int row, int col) Loads the TIFF images for the given well location and all channels and stores them in temporary files.
static java.lang.Object[][][]	**loadImages**(java.lang.String plate, int row, int col, java.lang.String[] channels) Loads the TIFF images for the given well location and list of channels and stores them in temporary files.
static void	**login**(java.lang.String user, java.lang.String password, java.lang.String url) Login to the openBIS server given as *url*.
static void	**logout**() Logs out and closes the session on the server.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

VERSION

```
public static final java.lang.String VERSION
```

The version of the API.

See Also:
Constant Field Values

REQUIRES_OPENBIS_AS_API

```
public static final java.lang.String REQUIRES_OPENBIS_AS_API
```

The required version ("major.minor") of the screening API on the openBIS application server.

See Also:
Constant Field Values

REQUIRES_OPENBIS_DSS_API

```
public static final java.lang.String REQUIRES_OPENBIS_DSS_API
```

The required version ("major.minor") of the screening API on the openBIS datastore server.

See Also:
Constant Field Values

Method Detail

login

```
public static void login(java.lang.String user,
                         java.lang.String password,
                         java.lang.String url)
```

Login to the openBIS server given as *url*.

Matlab example:

```
OpenBISScreeningML.login('user', 'secret', 'https://www.infectome.org')
```

Parameters:

`user` – The user id on the server
`password` – The password on the server
`url` – The URL, e.g. `https://www.infectome.org`

---

logout

```
public static void logout()
```

Logs out and closes the session on the server.

Matlab example:

```
OpenBISScreeningML.logout()
```

---

listExperiments

```
public static java.lang.Object[][] listExperiments()
```

Lists all experiment.

Matlab example:

```
% Get the experiments
exps = OpenBISScreeningML.listExperiments();
% How many experiments do we have?
length(exps)
% Get all information about experiment 3
exp3 = exps(3,:)
% Get the perm ids for all experiments
permids = exps(:,2)
```

Returns:

Each row contains information about one plate:

```
{ experiment augmented code, experiment perm id, experiment space code, experiment project code,
experiment code }
```

---

listPlates

```
public static java.lang.Object[][] listPlates()
```

Lists all plates.

Matlab example:

```
% Get the plates
plates = OpenBISScreeningML.listPlates();
% How many plates do we have?
length(plates)
% Get all information about plate 2
plate2 = plates(2,:)
% Get the simple plate codes for all plates
codes = plates(:,4)
```

Returns:

Each row contains information about one plate:

```
{ plate augmented code, plate perm id, plate space code, plate code, experiment augmented code,
experiment perm id, experiment space code, experiment project code, experiment code }
```

---

listPlates

```
public static java.lang.Object[][] listPlates(java.lang.String experiment)
```

Lists the plates of *experiment*.

Matlab example:

```
% Get the plates of experiment MYEXP in project PROJ of space SPACE
plates = OpenBISScreeningML.listPlates('/SPACE/PROJ/MYEXP');
% How many plates do we have?
length(plates)
% Get all information about plate 2
plate2 = plates(2,:)
% Get the augmented plate codes for all plates
acodes = plates(:,1)
```

Parameters:

`experiment` - The augmented code of the experiment to list the plates for

Returns:

Each row contains information about one plate:

```
{ plate augmented code, plate perm id, plate space code, plate code, experiment augmented code,
experiment perm id, experiment space code, experiment project code, experiment code }
```

---

listChannels

```
public static java.lang.Object[][] listChannels(java.lang.String experiment)
```

Lists all channels measured in *experiment*.

Matlab example:

```
% Get the channels of experiment MYEXP in project PROJ of space SPACE
channels = OpenBISScreeningML.listChannels('/SPACE/PROJ/MYEXP');
% How many channels do we have?
length(channels)
% What is the name of channel 1?
channels(1)
```

Parameters:

`experiment` - The augmented code of the experiment to list the channels for

Returns:

Each row contains information about one channel. Currently the only information available is the channel name.

---

listFeatures

```
public static java.lang.Object[][] listFeatures(java.lang.String experiment)
```

Lists all features computed for *experiment*.

Matlab example:

```
% Get the features of experiment MYEXP in project PROJ of space SPACE
features = OpenBISScreeningML.listFeatures('/SPACE/PROJ/MYEXP');
% How many features do we have?
length(features)
% What is the name of features 1?
features(1)
```

Parameters:

`experiment` - The augmented code of the experiment to list the features for

Returns:

Each row contains information about one feature. Currently the only information available is the feature name.

---

loadImages

```
public static java.lang.Object[][][] loadImages(java.lang.String plate,
                                                int row,
                                                int col)
```

Loads the TIFF images for the given well location and all channels and stores them in temporary files. The temporary files will be removed automatically when the Java Virtual Machine exits.

Matlab example:

```
% Load the images for all channels of well B10 of plate P005 in space SPACE
imginfo = OpenBISScreeningML.loadImages('/SPACE/P005', 2, 10)
% Get the plate-well descriptions of all locations
imginfo(2,:,3)
% Show the third image (assuming there are at least three images)
imtool(imginfo(1,3))
```

Parameters:

`plate` - The augmented plate code

`row` - The row in the plate to get the images for

`col` - The column in the plate to get the images for

Returns:

```
{ names of TIFF files, image annotation }
```

Each of `names of TIFF files` and `image annotation` is a vector of length of the number of images.

`image annotation` contains `{ channel name, tile number, plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }`

---

loadImages

```
public static java.lang.Object[][][] loadImages(java.lang.String plate,
                                                 int row,
                                                 int col,
                                                 java.lang.String[] channels)
```

Loads the TIFF images for the given well location and list of channels and stores them in temporary files. The temporary files will be removed automatically when the Java Virtual Machine exits.

Matlab example:

```
% Load the images for channel DAPI of well H10 of plate P005 in space SPACE
imginfo=OpenBISScreeningML.loadImages('/SPACE/P005', 8, 10, 'DAPI')
% Get the channel names and tile numbers of all locations
imginfo(2,:,1:2)
% Show the second image (assuming there are at least two images)
imtool(imginfo(1,2))
```

Parameters:

`plate` – The augmented plate code
`row` – The row in the plate to get the images for
`col` – The column in the plate to get the images for
`channels` – The names of the channels to get the images for

Returns:

`{ names of TIFF files, image annotation }`

Each of `names of TIFF files` and `image annotation` is a vector of length of the number of images.

`image annotation` contains `{ channel name, tile number, plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }`

---

getFeatureMatrix

```
public static java.lang.Object[][][] getFeatureMatrix(java.lang.String experiment,
                                                       java.lang.String gene)
```

Returns the feature matrix of all features for all locations in *experiment* (a location is one well position in one feature vector data set) connected to *gene* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.

One row in the matrix corresponds to one location (i.e. one well and one feature vector dataset), one column corresponds to one feature.

Matlab example:

```
% Get feature matrix for experiment /SPACE/PROJ/MYEXP for locations connected to GENENAME
fmatrix = OpenBISScreeningML.getFeatureMatrix('/SPACE/PROJ/MYEXP', 'GENENAME');
% Get the feature vector for the second location (assuming there are at least two locations)
loc2 = fmatrix(1,2,:)
% Get the values of the fifth feature for all locations (assuming there are at least 5 features)
feature5 = fmatrix(1,:,5)
% What are the features?
featureNames = fmatrix(3,:)
% Get the plate-well descriptions of the locations
locationDescriptions = fmatrix(2,:,1)
```

Parameters:

`experiment` – The augmented experiment code
`gene` – The gene name as stored as material code in openBIS

Returns:

`{ feature matrix, annotations per location, feature names }` where `annotations per location` contain:

`{ plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }`

---

getFeatureMatrix

```
public static java.lang.Object[][][] getFeatureMatrix(java.lang.String experiment,
                                                       java.lang.String gene,
                                                       java.lang.String[] features)
```

Returns the feature matrix of the specified features for all locations in *experiment* (a location is one well position in one feature vector data set) in *experiment* connected to *gene* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.

One row in the matrix corresponds to one location (i.e. one well and one feature vector dataset), one column corresponds to one feature.

Matlab example:

```
% Get feature matrix for features FEATURE1, FEATURE2 and FEATURE for
```

```
% experiment /SPACE/PROJ/MYEXP for locations connected to GENENAME
fmatrix = OpenBISScreeningML.getFeatureMatrix('/SPACE/PROJ/MYEXP', 'GENENAME', ('FEATURE1','FEATURE2','F
% Get the feature vector for the second location (assuming there are at least two locations)
loc2 = fmatrix(1,2,:)
% Get the values of the fourth feature for all locations (assuming there are at least 4 features)
feature5 = fmatrix(1,:,4)
% What are the features?
featureNames = fmatrix(3,:)
% Get the plate-well descriptions of the locations
locationDescriptions = fmatrix(2,:,1)
```

Parameters:

`experiment` – The augmented experiment code

`gene` – The gene name as stored as material code

`features` – The names of the features to contain the feature matrix

Returns:

`{ feature matrix, annotations per location, feature names }` where `annotations per location` contain:

`{ plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }`

---

getFeatureMatrix

```
public static java.lang.Object[][][] getFeatureMatrix(java.lang.String gene)
```

Returns the feature matrix of all features for all locations (a location is one well position in one feature vector data set) connected to *gene* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.

One row in the matrix corresponds to one location (i.e. one well and one feature vector dataset), one column corresponds to one feature.

Matlab example:

```
% Get feature matrix for GENENAME
fmatrix = OpenBISScreeningML.getFeatureMatrix('GENENAME');
% Get the feature vector for the second location (assuming there are at least two locations)
loc2 = fmatrix(1,2,:)
% Get the values of the fifth feature for all locations (assuming there are at least 5 features)
feature5 = fmatrix(1,:,5)
% What are the features?
featureNames = fmatrix(3,:)
% Get the plate-well descriptions of the locations
locationDescriptions = fmatrix(2,:,1)
```

Parameters:

`gene` – The gene name as stored as material code in openBIS

Returns:

`{ feature matrix, annotations per location, feature names }` where `annotations per location` contain:

`{ plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }`

---

getFeatureMatrix

```
public static java.lang.Object[][][] getFeatureMatrix(java.lang.String gene,
                                                       java.lang.String[] features)
```

Returns the feature matrix of the specified features for all locations (a location is one well position in one feature vector data set) in *experiment* connected to *gene* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.

One row in the matrix corresponds to one location (i.e. one well and one feature vector dataset), one column corresponds to one feature.

Matlab example:

```
% Get feature matrix for features FEATURE1, FEATURE2 and FEATURE for GENENAME
fmatrix = OpenBISScreeningML.getFeatureMatrix('GENENAME', ('FEATURE1','FEATURE2','FEATURE3'));
% Get the feature vector for the second location (assuming there are at least two locations)
loc2 = fmatrix(1,2,:)
% Get the values of the second feature ('FEATURE2' here) for all locations
feature2 = fmatrix(1,:,2)
% What are the features?
featureNames = fmatrix(3,:)
% Get the plate-well descriptions of the locations
locationDescriptions = fmatrix(2,:,1)
```

Parameters:

`gene` – The gene name as stored as material code

`features` – The names of the features to contain the feature matrix

Returns:

`{ feature matrix, annotations per location, feature names }` where `annotations per location` contain:

`{ plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }`

---

getFeatureMatrixForPlate

```
public static java.lang.Object[][][] getFeatureMatrixForPlate(java.lang.String plate)
```

Returns the feature matrix of all available features for all locations (a location is one well position in one feature vector data set) of all feature vector data sets of the given *plate* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.

One row in the matrix corresponds to one location (i.e. one well and one feature vector dataset), one column corresponds to one feature.

Matlab example:

```
% Get feature matrix for PLATECODE
fmatrix = OpenBISScreeningML.getFeatureMatrixForPlate('PLATECODE');
% Get the feature vector for the second location (assuming there are at least two locations)
loc2 = fmatrix(1,2,:)
% Get the values of the fourth feature for all locations (assuming there are at least 4 features)
feature5 = fmatrix(1,:,4)
% What are the features?
featureNames = fmatrix(3,:)
% Get the plate-well descriptions of the locations
locationDescriptions = fmatrix(2,:,1)
```

Parameters:

`plate` – The gene name as stored as material code

Returns:

`{ feature matrix, annotations per location, feature names }` where `annotations per location` contain:

`{ plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }`

---

getFeatureMatrixForPlate

```
public static java.lang.Object[][][] getFeatureMatrixForPlate(java.lang.String plate,
                                                               java.lang.String[] features)
```

Returns the feature matrix of the specified features for all locations (a location is one well position in one feature vector data set) of all feature vector data sets of the given *plate* in `[0]`, location annotations in `[1]` and feature annotation in `[2]`.

One row in the matrix corresponds to one location (i.e. one well and one feature vector dataset), one column corresponds to one feature.

Matlab example:

```
% Get feature matrix for features FEATURE1, FEATURE2 and FEATURE for PLATECODE
fmatrix = OpenBISScreeningML.getFeatureMatrixForPlate('PLATECODE', ('FEATURE1','FEATURE2','FEATURE3'));
% Get the feature vector for the second location (assuming there are at least two locations)
loc2 = fmatrix(1,2,:)
% Get the values of the second feature for all locations
feature5 = fmatrix(1,:,2)
% What are the features?
featureNames = fmatrix(3,:)
% Get the plate-well descriptions of the locations
locationDescriptions = fmatrix(2,:,1)
```

Parameters:

`plate` – The gene name as stored as material code

`features` – The names of the features to contain the feature matrix

Returns:

`{ feature matrix, annotations per location, feature names }` where `annotations per location` contain:

`{ plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column, experiment augmented code, experiment perm id, experiment space code, experiment project code, experiment code, data set code }`

---

getGeneMappingForPlates

```
public static java.lang.Object[][][] getGeneMappingForPlates(java.lang.String[] platesCodes)
```

Returns the gene mapping for the given *plateCodes* in `[0]` and location annotations in `[1]`.

One row in the matrix corresponds to one well.

Matlab example:

```
% Get feature matrix for features FEATURE1, FEATURE2 and FEATURE for PLATECODE
genes = getGeneMappingForPlate('PLATECODE');
% Get the plate well location description of the 10th wells
loc2 = genes(2,10,1)
% Get the gene ids that are in the 10th well
geneIds = genes(1,10,:)
```

Parameters:

`platesCodes` - The augmented codes of the plates to get the mapping for

Returns:

`{ gene ids, annotations per well }` where `gene ids` can be 0, 1 or more gene ids. `annotations per location` contain:

`{ plate well description, plate augmented code, plate perm id, plate space code, plate code, row, column }`

Package **Class** **Use** **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
