

---

# **Blackboard Sync**

***Release 0.9.11***

**Jacob Sánchez**

**Sep 26, 2023**



# CONTENTS

<b>1 Features</b>	<b>3</b>
<b>2 Installation</b>	<b>5</b>
2.1 From PyPI . . . . .	5
<b>3 API Documentation</b>	<b>7</b>
3.1 Blackboard API Reference . . . . .	7
3.2 Sync API Reference . . . . .	27
3.3 PyQt UI Reference . . . . .	29
<b>Python Module Index</b>	<b>35</b>
<b>Index</b>	<b>37</b>



**BlackboardSync** is a multiplatform desktop application written in Python that automatically downloads content from courses in your Blackboard account.

---



---

**CHAPTER  
ONE**

---

**FEATURES**

- Supported content:
  - Attachments of any type (e.g. .docx, .pptx, .pdf, etc.)
  - Internet links
  - Content descriptions (saved as html)
- Cross-platform
  - Linux, Windows, and macOS ready



---

**CHAPTER  
TWO**

---

## **INSTALLATION**

You can find all releases on [GitHub](#). Only MacOS (.dmg) and Windows (.exe) are supported at the moment.

### **2.1 From PyPI**

```
$ python3 -m pip install blackboardsync
$ python3 -m blackboard_sync # notice the underscore
```



## API DOCUMENTATION

If you are interested about contributing, or if you just want to understand the internals of BlackboardSync more.

### 3.1 Blackboard API Reference

#### 3.1.1 Blackboard REST API

REST API call parameters must be specified in keyword form. Blackboard API.  
an interface to make Blackboard REST API calls on a session basis

```
class blackboard_sync.blackboard.api.BlackboardSession(base_url: str, cookies: RequestsCookieJar)
```

Represents a user session in Blackboard.

```
property base_url: str
```

API base URL.

```
download(response)
```

Download the contents of a Content Item.

#### Parameters

- **course\_id** – The course or organization ID.
- **content\_id** – The Content ID.
- **attachment\_id** –

```
download_attempt_file_metadata(response)
```

Download the contents of the file for a Student Submission.

#### Parameters

- **course\_id** – The course or organization ID.
- **attempt\_id** –
- **attempt\_file\_id** –

```
download_webdav(response)
```

Downloads an arbitrary webdav file

```
fetch_announcements(response)
```

Return a list of System Announcements.

#### Parameters

- **announcement\_id** – Get a System Announcement.

**fetch\_associated\_nodes(*response*)**

Obtain a list of nodes to which a given course is directly associated.

**Parameters**

• **course\_id** – The course or organization ID.

**fetch\_attempt\_file\_metadata(*response*)**

Get the list of file metadata for a Submission associated to the course and attempt.

Get the file metadata for a Student Submission associated to the course and attempt.

**Parameters**

- **course\_id** – The course or organization ID.
- **attempt\_id** –
- **attempt\_file\_id** –

**fetch\_attendance\_data\_download\_url(*response*)**

Generate Download URL for Attendance Data.

**Parameters**

• **course\_id** – The course or organization ID.

**fetch\_attendance\_records\_by\_meeting\_id(*response*)**

Return a Course Meeting Attendance information for the given meeting and user Id.

**Parameters**

- **course\_id** – The course or organization ID.
- **meeting\_id** –
- **user\_id** – The user ID.

**fetch\_attendance\_records\_by\_user\_id(*response*)**

Return a list of Course Meeting Attendance for a given user id regardless of courses and meetings.

**Parameters**

- **course\_id** – The course or organization ID.
- **user\_id** – The user ID.

**fetch\_avatar(*response*)**

Get a user avatar image.

The response is an HTTP redirect rather than image raw data. It is up to the caller of the api to follow the redirect and download the image.

Not yet implemented. :param user\_id: The user ID.

**fetch\_calendar(*response*)**

Get the list of calendars. This endpoint will return all calendars viewable by the user.

All users can request a list of calendars viewable to them.

**fetch\_calendar\_items(*response*)**

Get a course calendar item.

**Parameters**

- **calendar\_item\_type** – One of (Course, GradebookColumn, Institution, OfficeHours, Personal).

- **calendar\_item\_id** –

**fetch\_categories(*response*)**

Get categories associated with the provided course.

**Parameters**

- **course\_id** – The course or organization ID.

**fetch\_category(*response*)**

Return a list of categories of the provided type (course or organization).

/ Return the category corresponding the provided type (course or organization) and ID.

**Parameters**

- **category\_type** – One of (Course, Organization).
- **category\_id** –

**fetch\_child\_categories(*response*)**

Return a list of categories which are children of the category...

corresponding to the provided type (course or organization) and Id

**Parameters**

- **category\_type** – One of (Course, Organization).
- **parent\_id** –

**fetch\_column\_attempts(*response*)**

Return a list of attempts associated with the specified grade column.

/ Load the grade column attempt for the specified id.

**Parameters**

- **course\_id** – The course or organization ID.
- **column\_id** – The grade column ID.
- **attempt\_id** –

**fetch\_column\_grade\_last\_changed(*response*)**

Load the grade column grade with the maximum change index.

**Parameters**

- **course\_id** – The course or organization ID.
- **column\_id** – The grade column ID.

**fetch\_column\_grades(*response*)**

Return a list of grades associated with the specified grade column.

/ Load the grade column grade for a specific user.

**Parameters**

- **course\_id** – The course or organization ID.
- **column\_id** – The grade column ID.
- **user\_id** – The user ID.

**fetch\_content\_children(response) → list[BBContentChild]**

List all child content items directly beneath another content item.

This is only valid for content items that are allowed to have children (e.g. Folders).

**Parameters**

- **course\_id** – The course or organization ID.
- **content\_id** – The Content ID.

**fetch\_content\_groups(response)**

Return a list of content group associations for the specified content.

**Parameters**

- **course\_id** – The course or organization ID.
- **content\_id** – The Content ID.
- **group\_id** – The group ID.

**fetch\_contents(response) → list[BBCourseContent]**

List top-level content items in a course.

**Parameters**

- **course\_id** – The course or organization ID.
- **content\_id** – The Content ID.

**fetch\_course\_announcements(response)**

Return a list of Course Announcements or Get a Course Announcement.

**Parameters**

- **course\_id** – The course or organization ID.
- **announcement\_id** –

**fetch\_course\_children(response)**

Return a list of course cross-listings.

/ Load a specific course cross-listing.

**Parameters**

- **course\_id** – The course or organization ID.
- **child\_course\_id** – The course or organization ID.

**fetch\_course\_meeting(response)**

Return a Course Meeting for the given meeting Id.

**Parameters**

- **course\_id** – The course or organization ID.
- **meeting\_id** –

**fetch\_course\_meetings(response)**

Return a list of course meetings for a given course id.

**Parameters**

- **course\_id** – The course or organization ID.

**fetch\_course\_memberships(*response*)**

Return a list of user memberships for the specified course or organization.

/ Load a user membership in the specified course.

**Parameters**

- **course\_id** – The course or organization ID.
- **user\_id** – The user ID.

**fetch\_course\_resource\_children(*response*)**

Return a list of Course Resources that are children of the specified Resource.

**Parameters**

- **course\_id** – The course or organization ID.
- **resource\_id** – The xythos resource ID.

**fetch\_course\_resources(*response*)**

Return a list of the top-level course resources.

or Load a Course Resource by Id.

**Parameters**

- **course\_id** – The course or organization ID.
- **resource\_id** – The xythos resource ID.

**fetch\_course\_roles(*response*)**

Return a list of course roles. / Return a single course role.

**Parameters**

**role\_id** – The course role ID.

**fetch\_courses(*response*) → *BBCourse* | list[*BBCourse*]**

Return a list of courses and organizations.

/ Loads a specific course or organization.

**Parameters**

**course\_id** – The course or organization ID.

**fetch\_cross\_list\_set(*response*)**

Return the course cross-listing set for the specified course.

This will return any and all parent/child associations regardless of the specified course being a parent or child course. The result will be empty if the specified course is not cross-listed.

**Parameters**

**course\_id** – The course or organization ID.

**fetch\_current\_active\_user(*response*)**

Display active session information for a specific user.

**Parameters**

**user\_id** – The user ID.

**fetch\_data\_sources(*response*)**

Return a list of data sources.

**Parameters**

**data\_source\_id** – The data source ID.

**fetch\_domain\_config(response)**

Return the list of LTI domain configs.

/ This endpoint returns the LTI domain config with the specified Id.

**Parameters**

**domain\_id** –

**fetch\_file\_attachments(response) → list[BBAttachment] | BBAttachment**

Get the file attachment meta data associated to the Content Item.

or Get the file attachment meta data by an attachment ID.

**Parameters**

- **course\_id** – The course or organization ID.
- **content\_id** – The Content ID.
- **attachment\_id** –

**fetch\_grade\_columns(response)**

Return a list of grade columns. / Load a specific grade column.

**Parameters**

- **course\_id** – The course or organization ID.
- **column\_id** – The grade column ID.

**fetch\_grade\_notations(response)**

Return a list of grade notations. / Return a specific grade notation.

**Parameters**

**course\_id** – The course or organization ID.

**Grade\_notation\_id**

**fetch\_grade\_schemas(response)**

Return a list of grade schemas associated with the specified course.

/ Load the grade schema associated with the specified course and schema Id.

**Parameters**

- **course\_id** – The course or organization ID.
- **schema\_id** – The grade schema ID.

**fetch\_gradebook\_categories(response)**

Return a list of gradebook categories in a particular course.

/ Return the details of a gradebook category.

**Parameters**

- **course\_id** – The course or organization ID.
- **category\_id** – the ID of the category to return

**fetch\_grading\_periods(response)**

Return a list of grading periods. / Return a specific grading period.

**Parameters**

- **course\_id** – The course or organization ID.

- **period\_id** –

**fetch\_group\_memberships(*response*)**

Return a list of group memberships objects for the specified group.

/ Loads a group membership in the specified group.

**Parameters**

- **course\_id** – The course or organization ID.
- **group\_id** – The group ID.
- **user\_id** – The user ID.

**fetch\_group\_set\_children(*response*)**

Return a list of all groups within a groupset.

**Parameters**

- **course\_id** – The course or organization ID.
- **group\_id** – The group ID.

**fetch\_group\_sets(*response*)**

Return a list of all groupsets / Load a groupset in the specified course.

**Parameters**

- **course\_id** – The course or organization ID.
- **group\_id** – The group ID.

**fetch\_groups(*response*)**

Return a list of all top-level groups in the specified course.

/ Load a group in the specified course.

**Parameters**

- **course\_id** – The course or organization ID.
- **group\_id** – The group ID.

**fetch\_institution\_roles(*response*)**

Return a list of institution roles. / Load a specific institution role.

**Parameters**

**role\_id** – The institution role ID.

**fetch\_memberships(*response*)**

Get courses associated with the provided category.

**Parameters**

- **category\_type** – One of (Course, Organization).
- **category\_id** –

**fetch\_node\_children(*response*)**

Return the children of the institutional hierarchy node corresponding...

to the provided ID.

**Parameters**

**node\_id** – The node ID.

**fetch\_node\_course\_associations(*response*)**

Return a list of node-course relationships for the specified node.

**Parameters**

**node\_id** – The node ID.

**fetch\_nodes(*response*)**

Return the Top-level institutional hierarchy nodes.

/ Return the institutional hierarchy node corresponding the provided ID.

**Parameters**

**node\_id** – The node ID.

**fetch\_observees(*response*)**

Return a list of users being observed by a given user.

**Parameters**

**user\_id** – The user ID.

**fetch\_observers(*response*)**

Return a list of users observing a given user.

**Parameters**

**user\_id** – The user ID.

**fetch\_performance\_review\_status(*response*)**

List the content review statuses for all the users enrolled in a course.

**Parameters**

**course\_id** – The course or organization ID.

**fetch\_placements(*response*)**

Return a list of LTI placements.

/ Returns the LTI placement with the specified Id.

**Parameters**

**placement\_id** –

**fetch\_policies(*response*)**

Return the links to the Blackboard and Institution privacy policies.

**fetch\_proctoring\_services(*response*)**

Return a list of proctoring service.

/ Return the proctoring service with the specified Id.

**Parameters**

**service\_id** –

**fetch\_questions(*response*)**

Get the list of questions for an Ultra Assessment or Get a question by Id from it.

**Parameters**

- **course\_id** – The course or organization ID.

- **assessment\_id** –

- **question\_id** –

**fetch\_review\_status(*response*)**

Obtain the review status for a content item.

This endpoint will only fetch the reviewStatus if the corresponding content was previously marked as reviewable.

**Parameters**

- **course\_id** – The course or organization ID.
- **content\_id** – The Content ID.
- **user\_id** – The user ID.

**fetch\_sessions(*response*)**

List active user sessions in Learn.

**fetch\_sis\_logs(*response*)**

Return a list of SIS Integration logs.

**Parameters**

**id** – dataSetUid of the integration

**fetch\_system\_roles(*response*)**

Return a list of system roles. / Get a specific system role by roleId.

**Parameters**

**role\_id** – The System Role ID.

**fetch\_system\_task(*response*)**

Get the background task by the given task Id.

**Parameters**

**task\_id** –

**fetch\_task(*response*)**

Check the status of a queued task associated with a Course.

Returns 200 unless task is complete.

**Parameters**

- **course\_id** – The course or organization ID.
- **task\_id** –

**fetch\_terms(*response*)**

Return a list of terms. / Load a term.

**Parameters**

**term\_id** – The term ID.

**fetch\_user\_grades(*response*)**

Load the course grades for a specific user.

**Parameters**

- **course\_id** – The course or organization ID.
- **user\_id** – The user ID.

**fetch\_user\_memberships(response) → list[BBMembership]**

Return a list of course and organization memberships for the specified user.

**Parameters**

**user\_id** – The user ID.

**fetch\_users(response)**

Return a list of users. / Load a user.

Properties returned will depend on the caller's entitlements.

**Parameters**

**user\_id** – The user ID.

**fetch\_version(response)**

Get the current Learn server version.

**get(version: int = 1, use\_api: bool = True, json: bool = True, \*\*g\_kwargs)**

Return a decorator (needed to use fancy notation).

**Parameters**

- **endpoint (string)** – Endpoint to make API call to, including placeholders
- **version (int)** – Version of the BB API used at endpoint (used as part of url)
- **json (bool)** – If false, returns raw requests response, otherwise returns JSON Object
- **g\_kwargs (dict)** – Any argument in this parameter will be passed on to the requests call

**property logger**

Logger for Blackboard API, set at level DEBUG.

**property timeout: int**

Request timeout.

**property username: str**

Username field used for API requests.

**class blackboard\_sync.blackboard.api.SafeFormat**

Custom dictionary object.

Needed to safely format endpoint strings with placeholders without having all parameters (those not present will be left blank)

### 3.1.2 Blackboard Data Classes

Blackboard Model Classes

**class blackboard\_sync.blackboard.blackboard.BBAttachment(\*, id: str | None = None, fileName: str | None = None, mimeType: str | None = None)**

Blackboard File Attachment.

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

```

fileName: str | None
id: str | None
mimeType: str | None
model_config: ClassVar[ConfigDict] = {'frozen': True}
    Configuration for the model, should be a dictionary conforming to [Config-Dict][pydantic.config.ConfigDict].
model_fields: ClassVar[dict[str, FieldInfo]] = {'fileName':
    FieldInfo(annotation=Union[str, NoneType], required=False), 'id':
    FieldInfo(annotation=Union[str, NoneType], required=False), 'mimeType':
    FieldInfo(annotation=Union[str, NoneType], required=False)}

```

Metadata about the fields defined on the model, mapping of field names to [Field-Info][pydantic.fields.FieldInfo].

This replaces *Model.\_fields* from Pydantic V1.

```

class blackboard_sync.blackboard.blackboard.BBAvailability(*, available: bool | None = None,
                                                               allowGuests: bool = False,
                                                               adaptiveRelease: dict = {}, duration:
                                                               BBDuration | None = None)

```

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

*\_\_init\_\_* uses *\_\_pydantic\_self\_\_* instead of the more common *self* for the first arg to allow *self* as a field name.

```

adaptiveRelease: dict
allowGuests: bool
available: bool | None
duration: BBDuration | None
model_config: ClassVar[ConfigDict] = {'frozen': True}
    Configuration for the model, should be a dictionary conforming to [Config-Dict][pydantic.config.ConfigDict].
model_fields: ClassVar[dict[str, FieldInfo]] = {'adaptiveRelease':
    FieldInfo(annotation=dict, required=False, default={}), 'allowGuests':
    FieldInfo(annotation=bool, required=False, default=False), 'available':
    FieldInfo(annotation=Union[bool, NoneType], required=False), 'duration':
    FieldInfo(annotation=Union[BBDuration, NoneType], required=False)}

```

Metadata about the fields defined on the model, mapping of field names to [Field-Info][pydantic.fields.FieldInfo].

This replaces *Model.\_fields* from Pydantic V1.

```
class blackboard_sync.blackboard.blackboard.BBContentChild(*, id: str | None = None, title: str | None = None, body: str | None = None, created: datetime | None = None, modified: datetime | None = None, position: int = 0, hasChildren: bool = False, launchInNewWindow: bool = False, reviewable: bool = False, availability: BBAvailability | None = None, contentHandler: BBCContentHandler | None = None, links: list[BBLink] = [], hasGradebookColumns: bool = False, hasAssociatedGroups: bool = False, parentId: str | None = None)
```

Blackboard Content Child.

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

**body:** str | None

**model\_config:** ClassVar[ConfigDict] = {'frozen': True}

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

**model\_fields:** ClassVar[dict[str, FieldInfo]] = {'availability': FieldInfo(annotation=Union[BBAvailability, NoneType], required=False), 'body': FieldInfo(annotation=Union[str, NoneType], required=False), 'contentHandler': FieldInfo(annotation=Union[BBCContentHandler, NoneType], required=False), 'created': FieldInfo(annotation=datetime, required=False), 'hasAssociatedGroups': FieldInfo(annotation=bool, required=False, default=False), 'hasChildren': FieldInfo(annotation=bool, required=False, default=False), 'hasGradebookColumns': FieldInfo(annotation=bool, required=False, default=False), 'id': FieldInfo(annotation=Union[str, NoneType], required=False), 'launchInNewWindow': FieldInfo(annotation=bool, required=False, default=False), 'links': FieldInfo(annotation=list[BBLink], required=False, default=[]), 'modified': FieldInfo(annotation=Union[datetime, NoneType], required=False), 'parentId': FieldInfo(annotation=Union[str, NoneType], required=False), 'position': FieldInfo(annotation=int, required=False, default=0), 'reviewable': FieldInfo(annotation=bool, required=False, default=False), 'title': FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [FieldInfo][pydantic.fields.FieldInfo].

This replaces `Model.__fields__` from Pydantic V1.

**parentId:** str | None

```
class blackboard_sync.blackboard.blackboard.BBContentHandler(*, id: BBResourceType | str | None
= None, url: str | None = None, file:
BBFile | None = None,
gradeColumnId: str | None = None,
groupContent: bool | None = None,
targetId: str | None = None,
targetType: str | None = None,
placementHandle: str | None =
None, assessmentId: str | None =
None, proctoring: BBProctoring |
None = None)
```

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

**assessmentId:** str | None

**file:** BBFile | None

**gradeColumnId:** str | None

**groupContent:** bool | None

**id:** BBResourceType | str | None

**property is\_not\_handled:** bool

Return true if resource should not be handled.

**model\_config:** ClassVar[ConfigDict] = {'frozen': True}

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

**model\_fields:** ClassVar[dict[str, FieldInfo]] = {'assessmentId':
FieldInfo(annotation=Union[str, NoneType], required=False), 'file':
FieldInfo(annotation=Union[BBFile, NoneType], required=False), 'gradeColumnId':
FieldInfo(annotation=Union[str, NoneType], required=False), 'groupContent':
FieldInfo(annotation=Union[bool, NoneType], required=False), 'id':
FieldInfo(annotation=Union[BBResourceType, str, NoneType], required=False),
'placementHandle': FieldInfo(annotation=Union[str, NoneType], required=False),
'proctoring': FieldInfo(annotation=Union[BBProctoring, NoneType], required=False),
'targetId': FieldInfo(annotation=Union[str, NoneType], required=False),
'targetType': FieldInfo(annotation=Union[str, NoneType], required=False), 'url':
FieldInfo(annotation=Union[str, NoneType], required=False)}

Metadata about the fields defined on the model, mapping of field names to [FieldInfo][pydantic.fields.FieldInfo].

This replaces `Model.__fields__` from Pydantic V1.

**placementHandle:** str | None

**proctoring:** BBProctoring | None

**classmethod resource\_parser(v: BBResourceType | str)**

Validate and parse an id resource type.

```
targetId: str | None
targetType: str | None
url: str | None

class blackboard_sync.blackboard.blackboard.BBCourse(*, id: str | None = None, courseId: str | None =
None, name: str | None = None, description:
str | None = None, modified: datetime | None =
None, organization: bool = False, ultraStatus:
str | None = None, closedComplete: bool =
False, availability: BBAvailability | None =
None, enrollment: BBEnrollment | None =
None, locale: BBLocale | None = None,
externalAccessUrl: str | None = None)

BlackboardCourse. Represents an academic course.

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic_core.ValidationError] if the input data cannot be validated to form a valid
model.

__init__ uses __pydantic_self__ instead of the more common self for the first arg to allow self as a field name.

availability: BBAvailability | None
closedComplete: bool
property code: str | None
    Parse course code.

courseId: str | None
description: str | None
enrollment: BBEnrollment | None
externalAccessUrl: str | None
id: str | None
locale: BBLocale | None
model_config: ClassVar[ConfigDict] = {'frozen': True}
    Configuration for the model, should be a dictionary conforming to [Config-
Dict][pydantic.config.ConfigDict].
model_fields: ClassVar[dict[str, FieldInfo]] = {'availability':
FieldInfo(annotation=Union[BBAvailability, NoneType], required=False),
'closedComplete': FieldInfo(annotation=bool, required=False, default=False),
'courseId': FieldInfo(annotation=Union[str, NoneType], required=False),
'description': FieldInfo(annotation=Union[str, NoneType], required=False),
'enrollment': FieldInfo(annotation=Union[BBEnrollment, NoneType], required=False),
'externalAccessUrl': FieldInfo(annotation=Union[str, NoneType], required=False),
'id': FieldInfo(annotation=Union[str, NoneType], required=False), 'locale':
FieldInfo(annotation=Union[BBLocale, NoneType], required=False), 'modified':
FieldInfo(annotation=datetime, required=False), 'name':
FieldInfo(annotation=Union[str, NoneType], required=False), 'organization':
FieldInfo(annotation=bool, required=False, default=False), 'ultraStatus':
FieldInfo(annotation=Union[str, NoneType], required=False)}
```

Metadata about the fields defined on the model, mapping of field names to [Field-Info][pydantic.fields.FieldInfo].

This replaces *Model.\_fields\_* from Pydantic V1.

**model\_post\_init**(\_\_context: Any) → None

This function is meant to behave like a `BaseModel` method to initialise private attributes.

It takes context as an argument since that's what pydantic-core passes when calling it.

**Args:**

self: The `BaseModel` instance. \_\_context: The context.

**modified**: `datetime | None`

**name**: `str | None`

**organization**: `bool`

**property title**: `str | None`

Parse course title.

**ultraStatus**: `str | None`

```
class blackboard_sync.blackboard.blackboard.BBCourseContent(*, id: str | None = None, title: str | None = None, body: str | None = None, created: datetime | None = None, modified: datetime | None = None, position: int = 0, hasChildren: bool = False, launchInNewWindow: bool = False, reviewable: bool = False, availability: BBAvailability | None = None, contentHandler: BBCContentHandler | None = None, links: list[BBLINK] = [], hasGradebookColumns: bool = False, hasAssociatedGroups: bool = False)
```

Blackboard Content.

Create a new model by parsing and validating input data from keyword arguments.

Raises `[ValidationError][pydantic_core.ValidationError]` if the input data cannot be validated to form a valid model.

`_init_` uses `pydantic_self_` instead of the more common `self` for the first arg to allow `self` as a field name.

**availability**: `BBAvailability | None`

**body**: `str | None`

**contentHandler**: `BBCContentHandler | None`

**created**: `datetime | None`

**hasAssociatedGroups**: `bool`

**hasChildren**: `bool`

**hasGradebookColumns**: `bool`

```
id: str | None
launchInNewWindow: bool
links: list[BBLINK]
model_config: ClassVar[ConfigDict] = {'frozen': True}
    Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].
model_fields: ClassVar[dict[str, FieldInfo]] = {'availability':
    FieldInfo(annotation=Union[BBAvailability, NoneType], required=False), 'body':
    FieldInfo(annotation=Union[str, NoneType], required=False), 'contentHandler':
    FieldInfo(annotation=Union[BBContentHandler, NoneType], required=False), 'created':
    FieldInfo(annotation=Union[datetime, NoneType], required=False),
    'hasAssociatedGroups': FieldInfo(annotation=bool, required=False, default=False),
    'hasChildren': FieldInfo(annotation=bool, required=False, default=False),
    'hasGradebookColumns': FieldInfo(annotation=bool, required=False, default=False),
    'id': FieldInfo(annotation=Union[str, NoneType], required=False),
    'launchInNewWindow': FieldInfo(annotation=bool, required=False, default=False),
    'links': FieldInfo(annotation=list[BBLINK], required=False, default=[]),
    'modified': FieldInfo(annotation=Union[datetime, NoneType], required=False),
    'position': FieldInfo(annotation=int, required=False, default=0), 'reviewable':
    FieldInfo(annotation=bool, required=False, default=False), 'title':
    FieldInfo(annotation=Union[str, NoneType], required=False)}
```

Metadata about the fields defined on the model, mapping of field names to [Field-Info][pydantic.fields.FieldInfo].

This replaces *Model.\_fields\_* from Pydantic V1.

```
modified: datetime | None
```

```
position: int
```

```
reviewable: bool
```

```
title: str | None
```

```
property title_path_safe: str
```

Return a path safe version of the title.

```
class blackboard_sync.blackboard.blackboard.BBDuration(*, type: str | None = None)
```

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

*\_\_init\_\_* uses *\_\_pydantic\_self\_\_* instead of the more common *self* for the first arg to allow *self* as a field name.

```
model_config: ClassVar[ConfigDict] = {'frozen': True}
```

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

```
model_fields: ClassVar[dict[str, FieldInfo]] = {'type':
    FieldInfo(annotation=Union[str, NoneType], required=False)}
```

Metadata about the fields defined on the model, mapping of field names to [Field-Info][pydantic.fields.FieldInfo].

This replaces *Model.\_fields\_* from Pydantic V1.

```

type: str | None

class blackboard_sync.blackboard.blackboard.BBDurationType(value, names=None, *, module=None,
qualname=None, type=None, start=1,
boundary=None)
    Blackboard Course Duration Type.

    Continuous: str = 'Continuous'

    DateRange: str = 'DateRange'

    FixedNumDays: str = 'FixedNumDays'

    Term: str = 'Term'

class blackboard_sync.blackboard.BBEnrollment(*, type: str | None = None)
    Create a new model by parsing and validating input data from keyword arguments.

    Raises [ValidationError][pydantic_core.ValidationError] if the input data cannot be validated to form a valid
    model.

    _init_ uses __pydantic_self__ instead of the more common self for the first arg to allow self as a field name.

    model_config: ClassVar[ConfigDict] = {'frozen': True}
        Configuration for the model, should be a dictionary conforming to [Config-
        Dict][pydantic.config.ConfigDict].

    model_fields: ClassVar[dict[str, FieldInfo]] = {'type':
        FieldInfo(annotation=Union[str, NoneType], required=False)}
        Metadata about the fields defined on the model, mapping of field names to [Field-
        Info][pydantic.fields.FieldInfo].
        This replaces Model._fields from Pydantic V1.

    type: str | None

class blackboard_sync.blackboard.BBFile(*, fileName: str | None = None)
    Blackboard File.

    Create a new model by parsing and validating input data from keyword arguments.

    Raises [ValidationError][pydantic_core.ValidationError] if the input data cannot be validated to form a valid
    model.

    _init_ uses __pydantic_self__ instead of the more common self for the first arg to allow self as a field name.

    fileName: str | None

    model_config: ClassVar[ConfigDict] = {'frozen': True}
        Configuration for the model, should be a dictionary conforming to [Config-
        Dict][pydantic.config.ConfigDict].

    model_fields: ClassVar[dict[str, FieldInfo]] = {'fileName':
        FieldInfo(annotation=Union[str, NoneType], required=False)}
        Metadata about the fields defined on the model, mapping of field names to [Field-
        Info][pydantic.fields.FieldInfo].
        This replaces Model._fields from Pydantic V1.

```

```
class blackboard_sync.blackboard.blackboard.BBLink(*, href: str | None = None, rel: str | None = None, title: str | None = None, type: str | None = None)
```

Blackboard Link.

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`_init_` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

**href: str | None**

**model\_config: ClassVar[ConfigDict] = {'frozen': True}**

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

**model\_fields: ClassVar[dict[str, FieldInfo]] = {'href':**

`FieldInfo(annotation=Union[str, NoneType], required=False), 'rel':`

`FieldInfo(annotation=Union[str, NoneType], required=False), 'title':`

`FieldInfo(annotation=Union[str, NoneType], required=False), 'type':`

`FieldInfo(annotation=Union[str, NoneType], required=False)}`

Metadata about the fields defined on the model, mapping of field names to [FieldInfo][pydantic.fields.FieldInfo].

This replaces `Model._fields_` from Pydantic V1.

**rel: str | None**

**title: str | None**

**type: str | None**

```
class blackboard_sync.blackboard.blackboard.BBLocale(*, force: bool = False)
```

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`_init_` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

**force: bool**

**model\_config: ClassVar[ConfigDict] = {'frozen': True}**

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

**model\_fields: ClassVar[dict[str, FieldInfo]] = {'force':**

`FieldInfo(annotation=bool, required=False, default=False)}`

Metadata about the fields defined on the model, mapping of field names to [FieldInfo][pydantic.fields.FieldInfo].

This replaces `Model._fields_` from Pydantic V1.

```
class blackboard_sync.blackboard.blackboard.BBMembership(*, id: str | None = None, userId: str | None = None, courseId: str | None = None, dataSourceId: str | None = None, created: datetime | None = None, modified: datetime | None = None, availability: BBAvailability | None = None, courseRoleId: str | None = None, lastAccessed: datetime | None = None, childCourseId: str | None = None)
```

Blackboard Membership. Represents relation between student and course.

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

**availability:** `BBAvailability | None`

**childCourseId:** `str | None`

**courseId:** `str | None`

**courseRoleId:** `str | None`

**created:** `datetime | None`

**dataSourceId:** `str | None`

**id:** `str | None`

**lastAccessed:** `datetime | None`

**model\_config:** `ClassVar[ConfigDict] = {'frozen': True}`

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

**model\_fields:** `ClassVar[dict[str, FieldInfo]] = {'availability': FieldInfo(annotation=Union[BBAvailability, NoneType], required=False), 'childCourseId': FieldInfo(annotation=Union[str, NoneType], required=False), 'courseId': FieldInfo(annotation=Union[str, NoneType], required=False), 'courseRoleId': FieldInfo(annotation=Union[str, NoneType], required=False), 'created': FieldInfo(annotation=Union[datetime, NoneType], required=False), 'dataSourceId': FieldInfo(annotation=Union[str, NoneType], required=False), 'id': FieldInfo(annotation=Union[str, NoneType], required=False), 'lastAccessed': FieldInfo(annotation=Union[datetime, NoneType], required=False), 'modified': FieldInfo(annotation=Union[datetime, NoneType], required=False), 'userId': FieldInfo(annotation=Union[str, NoneType], required=False)}`

Metadata about the fields defined on the model, mapping of field names to [FieldInfo][pydantic.fields.FieldInfo].

This replaces `Model.__fields__` from Pydantic V1.

**modified:** `datetime | None`

**userId:** `str | None`

```
class blackboard_sync.blackboard.blackboard.BBProctoring(*, secureBrowserRequiredToTake: bool = False, secureBrowserRequiredToReview: bool = False, webcamRequired: bool = False)
```

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`__init__` uses `__pydantic_self__` instead of the more common `self` for the first arg to allow `self` as a field name.

`model_config: ClassVar[ConfigDict] = {'frozen': True}`

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

```
model_fields: ClassVar[dict[str, FieldInfo]] = {'secureBrowserRequiredToReview': FieldInfo(annotation=bool, required=False, default=False), 'secureBrowserRequiredToTake': FieldInfo(annotation=bool, required=False, default=False), 'webcamRequired': FieldInfo(annotation=bool, required=False, default=False)}
```

Metadata about the fields defined on the model, mapping of field names to [FieldInfo][pydantic.fields.FieldInfo].

This replaces `Model.__fields__` from Pydantic V1.

`secureBrowserRequiredToReview: bool`

`secureBrowserRequiredToTake: bool`

`webcamRequired: bool`

```
class blackboard_sync.blackboard.blackboard.BBResourceType(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None)
```

Different resource types on Blackboard.

`asmt_test_link = 'x-bb-asmt-test-link'`

`assignment = 'x-bb-assignment'`

`blankpage = 'x-bb-blankpage'`

`bltiplacement_portal = 'x-bb-bltiplacement-Portal'`

`courselink = 'x-bb-courselink'`

`document = 'x-bb-document'`

`externallink = 'x-bb-externallink'`

`file = 'x-bb-file'`

`folder = 'x-bb-folder'`

`syllabus = 'x-bb-syllabus'`

`toollink = 'x-bb-toollink'`

`turnitin_assignment = 'x-turnitin-assignment'`

---

```
class blackboard_sync.blackboard.blackboard.ImmutableModel
```

Model with const attributes.

Create a new model by parsing and validating input data from keyword arguments.

Raises [ValidationError][pydantic\_core.ValidationError] if the input data cannot be validated to form a valid model.

`_init_` uses `_pydantic_self_` instead of the more common `self` for the first arg to allow `self` as a field name.

```
model_config: ClassVar[ConfigDict] = {'frozen': True}
```

Configuration for the model, should be a dictionary conforming to [ConfigDict][pydantic.config.ConfigDict].

```
model_fields: ClassVar[dict[str, FieldInfo]] = {}
```

Metadata about the fields defined on the model, mapping of field names to [FieldInfo][pydantic.fields.FieldInfo].

This replaces `Model._fields_` from Pydantic V1.

## 3.2 Sync API Reference

### 3.2.1 BlackboardSync

BlackboardSync.

Automatically sync content from Blackboard

```
class blackboard_sync.sync.BlackboardSync
```

Represents an instance of the BlackboardSync application.

Create an instance of the program.

```
_sync_task() → None
```

Constantly check if the data is outdated and if so start a download job.

Method run by Sync thread.

```
auth(cookie_jar: RequestsCookieJar) → bool
```

Create a new Blackboard session with the given credentials.

Will start syncing automatically if login successful.

#### Parameters

- `persistence (bool)` – If true, login will be saved in the OS designated keyring.

```
change_download_location(new_dir: Path, redownload: bool = False) → None
```

Set new sync location.

#### Parameters

- `dir (Path)` – The path of the sync dir.
- `redownload (bool)` – If true, ALL content will be re-downloaded to the new location.

```
property download_location: Path
```

Location to where all Blackboard content will be downloaded.

**force\_sync()** → `None`

Force Sync thread to start download job ASAP.

**property is\_active: bool**

Indicate the state of the sync thread.

**property is\_logged\_in: bool**

Indicate if a user session is currently active.

**property is\_syncing: bool**

Flag raised everytime a download job is running.

**property last\_sync\_time: datetime | None**

Datetime right before last download job started.

**log\_out()** → `None`

Stop syncing and forget user session.

**property logger: Logger**

Logger for BlackboardSync, set at level WARN.

**property min\_year: int**

**property next\_sync: datetime**

Time when last sync will be outdated.

**property outdated: bool**

Return true if last download job is outdated.

**setup(university\_index: int, download\_location: Path, min\_year: int | None = None)** → `None`

Setup the university information.

**start\_sync()** → `None`

Starts Sync thread.

**stop\_sync()** → `None`

Stop Sync thread.

**property sync\_interval: int**

Time to wait between download jobs.

**property university\_index**

**property username: str | None**

### 3.2.2 BlackboardDownload

BlackboardDownload, mass download all user content from Blackboard

**class blackboard\_sync.download.BlackboardDownload(sess: BlackboardSession, download\_location: Path, last\_downloaded: datetime | None = None, data\_sources: list[str] = [], min\_year: int | None = None)**

Blackboard download job.

BlackboardDownload constructor

Download all files in blackboard recursively to download\_location, only if they have been altered since specified datetime

Keyword arguments:

#### Parameters

- **sess** (`BlackboardSession`) – UCLan BB user session
- **download\_location** ((`str / Path`)) – Where files will be stored
- **last\_downloaded** (`str`) – Files modified before this will not be downloaded
- **data\_sources** – List of valid data sources
- **min\_year** – Only courses created on or after this year will be downloaded

`_create_desktop_link(path: Path, url: str, comment: str = '') → None`

Creates a platform-aware internet shortcut

`_handle_file(content: BBCourseContent, parent_path: Path, course_id: str, depth: int = 0) → None`

Download BBCContent recursively, depending on filetype

`cancel() → None`

Cancel the download job.

`property data_sources: list[str]`

Filter for courses.

`download() → datetime | None`

Retrieve the user's courses, and start download of all contents

#### Returns

Datetime when method was called.

`property download_location: Path`

The location where files will be downloaded to.

`property files_processed: int`

Number of files that have been downloaded.

`property logger: Logger`

Logger for BlackboardDownload, set at level DEBUG.

`property user_id: str`

User ID used for API calls.

## 3.3 PyQt UI Reference

### 3.3.1 Qt UI Elements

BlackboardSync Qt GUI.

`class blackboard_sync.qt.qt_elements.Assets`

Helper class to get the path of app assets.

`classmethod icon() → QIcon`

*QIcon* of application logo.

```
classmethod load_ui(qt_obj)
    Load a UI file for a QObject.
classmethod watermark() → QPixmap
    QPixmap of application watermark.
class blackboard_sync.qt.qt_elements.LoginWebView(start_url: str, target_url: str)
    Blackboard login widget.
Create instance of LoginWebView.
clear_cookie_store() → None
    Clear the HTTP cache and cookies.
property cookie_jar: RequestsCookieJar
    Contains session cookies of the current session.
property login_complete_signal
    Fire when the login flow has completed.
restore() → None
property url: str
    URL of current website.
class blackboard_sync.qt.qt_elements.LoginWindow
    Deprecated widget previously used to login.
class blackboard_sync.qt.qt_elements.OSUtils
    static add_to_startup() → None
        Add the app to start up on macOS.
    static open_dir_in_file_browser(dir_to_open: Path) → None
        Start a subprocess to open the default file explorer at the given location.
class blackboard_sync.qt.qt_elements.PersistenceWarning
    QDialog shown if user chooses to store their login details on their device.
Create instance of PersistenceWarning Dialog.
class blackboard_sync.qt.qt_elements.RedownloadDialog
    QMessageBox shown after a change in download location.
    It consults the user about whether files should be redownloaded to the new location or not.
Create a RedownloadDialog.
property redownload: bool
    Indicate if files have to be redownloaded.
class blackboard_sync.qt.qt_elements.SettingsWindow
    Settings window UI element.
Create instance of SettingsWindow.
property download_location: Path
    Path of download location.
```

```

property log_out_signal
    Fire when user chooses to log out.

property save_signal
    Fire when settings are saved.

property setup_wiz_signal
    Fire when user wants to redo initial setup.

property sync_frequency: int
    Seconds to wait between each sync job.

property username: str
    Username of current session.

class blackboard_sync.qt.qt_elements.SetupWizard(institutions: list[str])
    Initial setup wizard.

    Create a SetupWizard.

    Parameters
        institutions (list[str]) – List of institution names

class Pages(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None)
    Pages contained in the wizard.

    DOWNLOAD_LOCATION = 2
    DOWNLOAD_SINCE = 3
    INSTITUTION = 1
    INTRO = 0
    LAST = 3

    property download_location: Path
        Sync location path selected by user.

    initializePage(self, id: int)

    property institution: str
        Text of item selected in institution combo box.

    property institution_index: int
        Index of item selected in institution combo box.

    property min_year: int | None
        Courses from this year onward will be downloaded.

    validateCurrentPage() → bool
        Override QWizard method to validate pages.

class blackboard_sync.qt.qt_elements.SyncPeriod(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None)
    Enum containing all valid Sync intervals for this UI.

```

```
HALF_HOUR = 1800
ONE_HOUR = 3600
SIX_HOURS = 21600

class blackboard_sync.qt.qt_elements.SyncTrayIcon
    BlackboardSync system tray icon.

    Create a QSystemTrayIcon.

    property login_signal
        Fire once user is authenticated.

    property open_dir_signal
        Fire once user wants to open download directory.

    property quit_signal
        Fire once user decides to quit app.

    property reset_setup_signal
        Fire when the user wants to reset the initial setup.

    set_logged_in(logged: bool) → None
        Set logged-in status in menu.

    property settings_signal
        Fire when the settings menu is opened.

    property show_menu_signal
        Fire when menu is about to be shown.

    show_msg(title: str, msg: str, severity: int = 1, duration: int = 10) → None
        Show the user a message through the tray icon.

    property sync_signal
        Fire if user forces sync.

    toggle_currently_syncing(syncing: bool) → None
        Toggle currently syncing indicator in menu.

    update_last_synced(last: str) → None
        Update last sync time in menu.

class blackboard_sync.qt.qt_elements.SyncTrayMenu(logged_in: bool = False, last_synced: str = '')
    QMenu associated with app system tray icon.

    Create the menu for a SyncTrayIcon.
```

#### Parameters

- **logged\_in** (*bool*) – Whether user is currently logged in.
- **last\_synced** (*str*) – Last sync time shown.

**set\_logged\_in(logged: *bool*) → None**

Set the UI to reflect logged-in status.

**toggle\_currently\_syncing(syncing: *bool*) → None**

Toggle the currently syncing indicator.

**update\_last\_synced**(*last*: str) → None

Update the time of last download shown to user.

**class** blackboard\_sync.qt.qt\_elements.UniNotSupportedDialog(*help\_url*: str)

*QDialog* about unsupported Blackboard partners.

Create instance of dialog.

**Parameters**

**help\_url** (str) – URL to help website

**class** blackboard\_sync.qt.qt\_elements.UpdateFoundDialog

*QMessageBox* shown after a more recent version was found.

Create a *UpdateFoundDialog*.

**property** should\_update: bool

Indicate if BBSync should be updated.



## PYTHON MODULE INDEX

### b

`blackboard_sync.blackboard.api`, 7  
`blackboard_sync.blackboard.blackboard`, 16  
`blackboard_sync.download`, 28  
`blackboard_sync.qt.qt_elements`, 29  
`blackboard_sync.sync`, 27



# INDEX

## Symbols

<code>_create_desktop_link()</code> ( <i>blackboard_sync.download.BlackboardDownload method</i> ), 29	<code>BBAttachment</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 16
<code>_handle_file()</code> ( <i>blackboard_sync.download.BlackboardDownload method</i> ), 29	<code>BBAvailability</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 17
<code>_sync_task()</code> ( <i>blackboard_sync.sync.BlackboardSync method</i> ), 27	<code>BBContentChild</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 17
	<code>BBContentHandler</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 18
	<code>BBCourse</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 20
	<code>BBCourseContent</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 21
	<code>BBDuration</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 22
	<code>BBDurationType</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 23
	<code>BBEnrollment</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 23
	<code>BBFile</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 23
	<code>BBLINK</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 23
	<code>BBLocale</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 24
	<code>BBContentHandler</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 24
	<code>BBMembership</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 24
	<code>BBResourceType</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 24
	<code>BBProctoring</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 25
	<code>BBResourceType</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 25
	<code>BBCourse</code> ( <i>class in blackboard_sync.blackboard.blackboard</i> ), 26
	<code>blackboard_sync.blackboard.api</code>
	<code>BBCourseContent</code> ( <i>module in blackboard_sync.blackboard.blackboard</i> ), 7
	<code>BBMembership</code> ( <i>module in blackboard_sync.blackboard.download</i> ), 16
	<code>BBAvailability</code> ( <i>module in blackboard_sync.qt.qt_elements</i> ), 28
	<code>BBResourceType</code> ( <i>module in blackboard_sync.qt.qt_elements</i> ), 29
	<code>blackboard_sync.sync</code>
	<code>base_url</code> ( <i>blackboard_sync.blackboard.api.BlackboardSession property</i> ), 7
	<code>BlackboardDownload</code> ( <i>class in blackboard_sync.download</i> ), 28

BlackboardSession (class in blackboard\_sync.blackboard.api), 7  
BlackboardSync (class in blackboard\_sync.sync), 27  
blankpage (blackboard\_sync.blackboard.blackboard.BBResourceType attribute), 23  
attribute), 26  
bltipplacement\_portal (blackboard\_sync.blackboard.BBResourceType attribute), 26  
body (blackboard\_sync.blackboard.blackboard.BBContentContent attribute), 18  
body (blackboard\_sync.blackboard.blackboard.BBCourseContent attribute), 21

**C**

cancel() (blackboard\_sync.download.BlackboardDownload method), 29  
change\_download\_location() (blackboard\_sync.sync.BlackboardSync method), 27  
childCourseId (blackboard\_sync.blackboard.blackboard.BBMembership attribute), 25  
clear\_cookie\_store() (blackboard\_sync.qt.qt\_elements.LoginWebView method), 30  
closedComplete (blackboard\_sync.blackboard.blackboard.BBCourse attribute), 20  
code (blackboard\_sync.blackboard.blackboard.BBCourse property), 20  
contentHandler (blackboard\_sync.blackboard.blackboard.BBCourseContent attribute), 21  
Continuous (blackboard\_sync.blackboard.blackboard.BBDurationType attribute), 23  
cookie\_jar (blackboard\_sync.qt.qt\_elements.LoginWebView property), 30  
courseId (blackboard\_sync.blackboard.blackboard.BBCourse attribute), 20  
courseId (blackboard\_sync.blackboard.blackboard.BBMembership attribute), 25  
courseLink (blackboard\_sync.blackboard.blackboard.BBResourceType attribute), 26  
courseRoleId (blackboard\_sync.blackboard.blackboard.BBMembership attribute), 25  
created (blackboard\_sync.blackboard.blackboard.BBCourse attribute), 21  
created (blackboard\_sync.blackboard.blackboard.BBMembership attribute), 25

**D**

data\_sources (blackboard\_sync.download.BlackboardDownload property), 29

dataSourceId (blackboard\_sync.blackboard.blackboard.BBMembership attribute), 25  
DateRange (blackboard\_sync.blackboard.blackboard.BBDurationType attribute), 23  
description (blackboard\_sync.blackboard.blackboard.BBCourse attribute), 20  
document (blackboard\_sync.blackboard.blackboard.BBResourceType attribute), 26

download() (blackboard\_sync.blackboard.api.BlackboardSession method), 7  
download() (blackboard\_sync.download.BlackboardDownload method), 29  
download\_attempt\_file\_metadata() (blackboard\_sync.blackboard.api.BlackboardSession method), 7  
download\_location (blackboard\_sync.download.BlackboardDownload property), 29  
download\_location (blackboard\_sync.qt.qt\_elements.SettingsWindow property), 30  
download\_location (blackboard\_sync.qt.qt\_elements.SetupWizard property), 31  
DOWNLOAD\_LOCATION (blackboard\_sync.qt.qt\_elements.SetupWizard.Pages attribute), 31  
download\_location (blackboard\_sync.sync.BlackboardSync property), 27  
DOWNLOAD\_SINCE (blackboard\_sync.qt.qt\_elements.SetupWizard.Pages attribute), 31  
download\_webdav() (blackboard\_sync.blackboard.api.BlackboardSession method), 7  
duration (blackboard\_sync.blackboard.blackboard.BBAvailability attribute), 17

**E**

enrollment (blackboard\_sync.blackboard.blackboard.BBCourse attribute), 20  
externalAccessUrl (blackboard\_sync.blackboard.blackboard.BBCourse attribute), 20  
externalLink (blackboard\_sync.blackboard.blackboard.BBResourceType attribute), 26

**F**

fetch\_announcements() (blackboard\_sync.blackboard.api.BlackboardSession method), 7  
fetch\_associated\_nodes() (blackboard\_sync.blackboard.api.BlackboardSession

<code>method), 8</code>	<code>method), 10</code>
<code>fetch_attempt_file_metadata()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 8</code>	<code>fetch_course_meeting()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 10</code>
<code>fetch_attendance_data_download_url()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 8</code>	<code>fetch_course_meetings()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 10</code>
<code>fetch_attendance_records_by_meeting_id()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 8</code>	<code>fetch_course_memberships()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 10</code>
<code>fetch_attendance_records_by_user_id()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 8</code>	<code>fetch_course_resource_children()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 11</code>
<code>fetch_avatar()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 8</code>	<code>fetch_course_resources()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 11</code>
<code>fetch_calendar()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 8</code>	<code>fetch_course_roles()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 11</code>
<code>fetch_calendar_items()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 8</code>	<code>fetch_courses()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 11</code>
<code>fetch_categories()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 9</code>	<code>fetch_cross_list_set()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 11</code>
<code>fetch_category()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 9</code>	<code>fetch_current_active_user()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 11</code>
<code>fetch_child_categories()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 9</code>	<code>fetch_data_sources()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 11</code>
<code>fetch_column_attempts()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 9</code>	<code>fetch_domain_config()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 12</code>
<code>fetch_column_grade_last_changed()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 9</code>	<code>fetch_file_attachments()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 12</code>
<code>fetch_column_grades()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 9</code>	<code>fetch_grade_columns()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 12</code>
<code>fetch_content_children()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 9</code>	<code>fetch_grade_notations()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 12</code>
<code>fetch_content_groups()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 10</code>	<code>fetch_grade_schemas()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 12</code>
<code>fetch_contents()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 10</code>	<code>fetch_gradebook_categories()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 12</code>
<code>fetch_course_announcements()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 10</code>	<code>fetch_grading_periods()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession method), 12</code>
<code>fetch_course_children()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession</code>	<code>fetch_group_memberships()</code> ( <code>blackboard_sync.blackboard.api.BlackboardSession</code>

method), 13  
`fetch_group_set_children()` (blackboard\_sync.blackboard.api.BlackboardSession method), 13  
`fetch_group_sets()` (blackboard\_sync.blackboard.api.BlackboardSession method), 13  
`fetch_groups()` (blackboard\_sync.blackboard.api.BlackboardSession method), 13  
`fetch_institution_roles()` (blackboard\_sync.blackboard.api.BlackboardSession method), 13  
`fetch_memberships()` (blackboard\_sync.blackboard.api.BlackboardSession method), 13  
`fetch_node_children()` (blackboard\_sync.blackboard.api.BlackboardSession method), 13  
`fetch_node_course_associations()` (blackboard\_sync.blackboard.api.BlackboardSession method), 13  
`fetch_nodes()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_observees()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_observers()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_performance_review_status()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_placements()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_policies()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_proctoring_services()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_questions()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_review_status()` (blackboard\_sync.blackboard.api.BlackboardSession method), 14  
`fetch_sessions()` (blackboard\_sync.blackboard.api.BlackboardSession method), 15  
`fetch_sis_logs()` (blackboard\_sync.blackboard.api.BlackboardSession method), 15  
`fetch_system_roles()` (blackboard\_sync.blackboard.api.BlackboardSession method), 15  
`fetch_system_task()` (blackboard\_sync.blackboard.api.BlackboardSession method), 15  
`fetch_task()` (blackboard\_sync.blackboard.api.BlackboardSession method), 15  
`fetch_terms()` (blackboard\_sync.blackboard.api.BlackboardSession method), 15  
`fetch_user_grades()` (blackboard\_sync.blackboard.api.BlackboardSession method), 15  
`fetch_user_memberships()` (blackboard\_sync.blackboard.api.BlackboardSession method), 15  
`fetch_users()` (blackboard\_sync.blackboard.api.BlackboardSession method), 16  
`fetch_version()` (blackboard\_sync.blackboard.api.BlackboardSession method), 16  
`file` (blackboard\_sync.blackboard.blackboard.BBContentHandler attribute), 19  
`file` (blackboard\_sync.blackboard.blackboard.BBResourceType attribute), 26  
`fileName` (blackboard\_sync.blackboard.blackboard.BBAttachment attribute), 16  
`fileName` (blackboard\_sync.blackboard.blackboard.BBFile attribute), 23  
`files_processed` (blackboard\_sync.download.BlackboardDownload property), 29  
`FixedNumDays` (blackboard\_sync.blackboard.blackboard.BBDurationType attribute), 23  
`folder` (blackboard\_sync.blackboard.blackboard.BBResourceType attribute), 26  
`force` (blackboard\_sync.blackboard.blackboard.BBLocale attribute), 24  
`force_sync()` (blackboard\_sync.sync.BlackboardSync method), 27

## G

`get()` (blackboard\_sync.blackboard.api.BlackboardSession method), 16  
`gradeColumnId` (blackboard\_sync.blackboard.blackboard.BBContentHandler attribute), 19  
`groupContent` (blackboard\_sync.blackboard.blackboard.BBContentHandler attribute), 19

**H**

HALF\_HOUR (*blackboard\_sync.qt.qt\_elements.SyncPeriod attribute*), 31  
**hasAssociatedGroups** (*blackboard\_sync.blackboard.BBCourseContent attribute*), 21  
**hasChildren** (*blackboard\_sync.blackboard.blackboard.BBCourseContent attribute*), 21  
**hasGradebookColumns** (*blackboard\_sync.blackboard.BBCourseContent attribute*), 21  
**href** (*blackboard\_sync.blackboard.blackboard.BBLink attribute*), 24

**I**

**icon()** (*blackboard\_sync.qt.qt\_elements.Assets class method*), 29  
**id** (*blackboard\_sync.blackboard.blackboard.BBAttachment attribute*), 17  
**id** (*blackboard\_sync.blackboard.blackboard.BBContentHandler attribute*), 19  
**id** (*blackboard\_sync.blackboard.blackboard.BBCourse attribute*), 20  
**id** (*blackboard\_sync.blackboard.blackboard.BBCourseContent attribute*), 21  
**id** (*blackboard\_sync.blackboard.blackboard.BBMembership attribute*), 25  
**ImmutableModel** (*class in blackboard\_sync.blackboard.blackboard*), 26  
**initializePage()** (*blackboard\_sync.qt.qt\_elements.SetupWizard method*), 31  
**institution** (*blackboard\_sync.qt.qt\_elements.SetupWizard property*), 31  
**INSTITUTION** (*blackboard\_sync.qt.qt\_elements.SetupWizard attribute*), 31

**INSTITUTION\_index**

(*blackboard\_sync.qt.qt\_elements.SetupWizard property*), 31

**INTRO** (*blackboard\_sync.qt.qt\_elements.SetupWizard.Pages attribute*), 31

**is\_active** (*blackboard\_sync.sync.BlackboardSync property*), 28

**is\_logged\_in** (*blackboard\_sync.sync.BlackboardSync property*), 28

**is\_not\_handled** (*blackboard\_sync.blackboard.blackboard.BBContentHandler property*), 19

**is\_syncing** (*blackboard\_sync.sync.BlackboardSync property*), 28

**L**

**LAST** (*blackboard\_sync.qt.qt\_elements.SetupWizard.Pages attribute*), 31

**last\_sync\_time** (*blackboard\_sync.sync.BlackboardSync property*), 28

**lastAccessed** (*blackboard\_sync.blackboard.blackboard.BBMembership attribute*), 25

**launchInNewWindow** (*blackboard\_sync.blackboard.blackboard.BBCourseContent attribute*), 22

**links** (*blackboard\_sync.blackboard.blackboard.BBCourseContent attribute*), 22

**load\_ui()** (*blackboard\_sync.qt.qt\_elements.Assets class method*), 29

**locale** (*blackboard\_sync.blackboard.blackboard.BBCourse attribute*), 20

**log\_out()** (*blackboard\_sync.sync.BlackboardSync method*), 28

**log\_out\_signal** (*blackboard\_sync.qt.qt\_elements.SettingsWindow property*), 30

**logger** (*blackboard\_sync.blackboard.api.BlackboardSession property*), 16

**logger** (*blackboard\_sync.download.BlackboardDownload property*), 29

**logger** (*blackboard\_sync.sync.BlackboardSync property*), 28

**login\_complete\_signal** (*blackboard\_sync.qt.qt\_elements.LoginWebView property*), 30

**login\_signal** (*blackboard\_sync.qt.qt\_elements.SyncTrayIcon property*), 32

**LoginWebView** (*class in blackboard\_sync.qt.qt\_elements*), 30

**LoginWindow** (*class in blackboard\_sync.qt.qt\_elements*), 30

**M** **mimeType** (*blackboard\_sync.blackboard.blackboard.BBAttachment attribute*), 17

**min\_year** (*blackboard\_sync.qt.qt\_elements.SetupWizard property*), 31

**min\_year** (*blackboard\_sync.sync.BlackboardSync property*), 28

**model\_config** (*blackboard\_sync.blackboard.blackboard.BBAttachment attribute*), 17

**model\_config** (*blackboard\_sync.blackboard.blackboard.BBAvailability attribute*), 17

**model\_config** (*blackboard\_sync.blackboard.blackboard.BBContentChild attribute*), 18

**model\_config** (*blackboard\_sync.blackboard.blackboard.BBContentHandler attribute*), 19

**model\_config** (*blackboard\_sync.blackboard.blackboard.BBCourse attribute*), 20

**model\_config** (*blackboard\_sync.blackboard.blackboard.BBCourseContent attribute*), 22

model\_config (blackboard\_sync.blackboard.blackboard.BBDDownload blackboard\_sync.blackboard.api, 7  
attribute), 22  
blackboard\_sync.blackboard.blackboard, 16

model\_config (blackboard\_sync.blackboard.blackboard.BBEnrollment blackboard\_sync.download, 28  
attribute), 23  
blackboard\_sync.qt.qt\_elements, 29

model\_config (blackboard\_sync.blackboard.blackboard.BBFile blackboard\_sync.sync, 27  
attribute), 23

model\_config (blackboard\_sync.blackboard.blackboard.BBLINK  
attribute), 24  
name (blackboard\_sync.blackboard.blackboard.BBCourse  
attribute), 21

model\_config (blackboard\_sync.blackboard.blackboard.BBLocale attribute), 24  
next\_sync (blackboard\_sync.sync.BlackboardSync  
attribute), 28

model\_config (blackboard\_sync.blackboard.blackboard.BBMembershipProperty), 25  
attribute), 25

model\_config (blackboard\_sync.blackboard.blackboard.BBProctoring  
attribute), 26  
ONE\_HOUR (blackboard\_sync.qt.qt\_elements.SyncPeriod  
attribute), 32

model\_config (blackboard\_sync.blackboard.blackboard.ImmutableModel  
attribute), 27  
open\_dir\_in\_file\_browser() (blackboard\_sync.qt.qt\_elements.OSUtils  
attribute), 30  
static

model\_fields (blackboard\_sync.blackboard.blackboard.BBAttachment  
attribute), 17  
method), 30

model\_fields (blackboard\_sync.blackboard.blackboard.BBAvailabilitySignal  
attribute), 17  
open\_file\_signal (blackboard\_sync.qt.qt\_elements.SyncTrayIcon  
attribute), 32

model\_fields (blackboard\_sync.blackboard.blackboard.BBContentChild  
attribute), 18  
organization (blackboard\_sync.blackboard.blackboard.BBCourse  
attribute), 21

model\_fields (blackboard\_sync.blackboard.blackboard.BBContentHandler  
attribute), 19  
OSUtils (class in blackboard\_sync.qt.qt\_elements), 30

model\_fields (blackboard\_sync.blackboard.blackboard.BBContent  
attribute), 20  
property), 28

model\_fields (blackboard\_sync.blackboard.blackboard.BBCourseContent  
attribute), 22

P

model\_fields (blackboard\_sync.blackboard.blackboard.BBDuration  
attribute), 22  
parent\_id (blackboard\_sync.blackboard.blackboard.BBContentChild  
attribute), 18

model\_fields (blackboard\_sync.blackboard.blackboard.BBPersistenceWarning  
attribute), 23  
(class in blackboard\_sync.qt.qt\_elements), 30

model\_fields (blackboard\_sync.blackboard.blackboard.BBFilementHandle  
attribute), 23  
blackboard\_sync.blackboard.blackboard.BBContentHandler  
attribute), 19

model\_fields (blackboard\_sync.blackboard.blackboard.BBLINK  
attribute), 24  
position (blackboard\_sync.blackboard.blackboard.BBCourseContent  
attribute), 22

model\_fields (blackboard\_sync.blackboard.blackboard.BBLocale  
attribute), 24  
proctoring (blackboard\_sync.blackboard.blackboard.BBContentHandler  
attribute), 25

model\_fields (blackboard\_sync.blackboard.blackboard.BBMembership  
attribute), 19

model\_fields (blackboard\_sync.blackboard.blackboard.BBProctoring  
attribute), 26  
quit\_signal (blackboard\_sync.qt.qt\_elements.SyncTrayIcon  
attribute), 32

model\_post\_init() (black- R  
board\_sync.blackboard.blackboard.BBCourse  
method), 21  
redownload (blackboard\_sync.qt.qt\_elements.RedownloadDialog  
property), 30

modified (blackboard\_sync.blackboard.blackboard.BBCourse  
attribute), 21  
RedownloadDialog (class in blackboard\_sync.qt.qt\_elements), 30

modified (blackboard\_sync.blackboard.blackboard.BBCourseContent  
attribute), 22  
redownloadDialog (blackboard\_sync.blackboard.blackboard.BBLINK at-  
tribute), 24

modified (blackboard\_sync.blackboard.blackboard.BBMembership  
attribute), 25  
reset\_setup\_signal (black-  
board\_sync.qt.qt\_elements.SyncTrayIcon  
property), 32

module

**S**  
 resource\_parser() (blackboard\_sync.blackboard.blackboard.BBContentHandler attribute), 26  
 class method), 19  
 restore() (blackboard\_sync.qt.qt\_elements.LoginWebView method), 30  
 reviewable (blackboard\_sync.blackboard.blackboard.BBContentHandler attribute), 22  
**S**  
 SafeFormat (class in blackboard\_sync.blackboard.api), 16  
 save\_signal (blackboard\_sync.qt.qt\_elements.SettingsWindow property), 31  
 secureBrowserRequiredToReview (blackboard\_sync.blackboard.blackboard.BBProctoring attribute), 26  
 secureBrowserRequiredToTake (blackboard\_sync.blackboard.blackboard.BBProctoring attribute), 26  
 set\_logged\_in() (blackboard\_sync.qt.qt\_elements.SyncTrayIcon method), 32  
 set\_logged\_in() (blackboard\_sync.qt.qt\_elements.SyncTrayMenu method), 32  
 settings\_signal (blackboard\_sync.qt.qt\_elements.SyncTrayIcon property), 32  
 SettingsWindow (class in blackboard\_sync.qt.qt\_elements), 30  
 setup() (blackboard\_sync.sync.BlackboardSync method), 28  
 setup\_wiz\_signal (blackboard\_sync.qt.qt\_elements.SettingsWindow property), 31  
 SetupWizard (class in blackboard\_sync.qt.qt\_elements), 31  
 SetupWizard.Pages (class in blackboard\_sync.qt.qt\_elements), 31  
 should\_update (blackboard\_sync.qt.qt\_elements.UpdateFoundDialog property), 33  
 show\_menu\_signal (blackboard\_sync.qt.qt\_elements.SyncTrayIcon property), 32  
 show\_msg() (blackboard\_sync.qt.qt\_elements.SyncTrayIcon type (blackboard\_sync.blackboard.blackboard.BBDuration attribute), 22  
 SIX\_HOURS (blackboard\_sync.qt.qt\_elements.SyncPeriod attribute), 32  
 start\_sync() (blackboard\_sync.sync.BlackboardSync method), 28  
 stop\_sync() (blackboard\_sync.sync.BlackboardSync method), 28

**T**  
 syllabus (blackboard\_sync.blackboard.blackboard.BBResourceType sync\_frequency (blackboard\_sync.qt.qt\_elements.SettingsWindow property), 31  
 sync\_interval (blackboard\_sync.sync.BlackboardSync property), 28  
 sync\_signal (blackboard\_sync.qt.qt\_elements.SyncTrayIcon property), 32  
 SyncPeriod (class in blackboard\_sync.qt.qt\_elements), 31  
 SyncTrayIcon (class in blackboard\_sync.qt.qt\_elements), 32  
 SyncTrayMenu (class in blackboard\_sync.qt.qt\_elements), 32  
**T**  
 targetId (blackboard\_sync.blackboard.blackboard.BBContentHandler attribute), 19  
 targetType (blackboard\_sync.blackboard.blackboard.BBContentHandler attribute), 20  
 Term (blackboard\_sync.blackboard.blackboard.BBDurationType attribute), 23  
 timeout (blackboard\_sync.blackboard.api.BlackboardSession property), 16  
 title (blackboard\_sync.blackboard.blackboard.BBCourse property), 21  
 title (blackboard\_sync.blackboard.blackboard.BBCourseContent attribute), 22  
 title (blackboard\_sync.blackboard.blackboard.BBLink attribute), 24  
 title\_path\_safe (blackboard\_sync.blackboard.blackboard.BBCourseContent property), 22  
 toggle\_currently\_syncing() (blackboard\_sync.qt.qt\_elements.SyncTrayIcon method), 32  
 toggle\_currently\_syncing() (blackboard\_sync.qt.qt\_elements.SyncTrayMenu method), 32  
 tomlink (blackboard\_sync.blackboard.blackboard.BBResourceType attribute), 26  
 turnitin\_assignment (blackboard\_sync.blackboard.blackboard.BBResourceType attribute), 26  
 type (blackboard\_sync.blackboard.blackboard.BBEnrollment attribute), 23  
 type (blackboard\_sync.blackboard.blackboard.BBLink attribute), 24

**U**  
 ultraStatus (blackboard\_sync.blackboard.blackboard.BBCourse

*attribute), 21*  
**UniNotSupportedDialog** (class in *blackboard\_sync.qt.qt\_elements*), 33  
**university\_index** (*blackboard\_sync.sync.BlackboardSync* property), 28  
**update\_last\_synced()** (*blackboard\_sync.qt.qt\_elements.SyncTrayIcon* method), 32  
**update\_last\_synced()** (*blackboard\_sync.qt.qt\_elements.SyncTrayMenu* method), 32  
**UpdateFoundDialog** (class in *blackboard\_sync.qt.qt\_elements*), 33  
**url** (*blackboard\_sync.blackboard.blackboard.BBContentHandler* attribute), 20  
**url** (*blackboard\_sync.qt.qt\_elements.LoginWebView* property), 30  
**user\_id** (*blackboard\_sync.download.BlackboardDownload* property), 29  
**userId** (*blackboard\_sync.blackboard.blackboard.BBMembership* attribute), 25  
**username** (*blackboard\_sync.blackboard.api.BlackboardSession* property), 16  
**username** (*blackboard\_sync.qt.qt\_elements.SettingsWindow* property), 31  
**username** (*blackboard\_sync.sync.BlackboardSync* property), 28

## V

**validateCurrentPage()** (*blackboard\_sync.qt.qt\_elements.SetupWizard* method), 31

## W

**watermark()** (*blackboard\_sync.qt.qt\_elements.Assets* class method), 30  
**webcamRequired** (*blackboard\_sync.blackboard.blackboard.BBProctoring* attribute), 26