# XBlock User Service (11/26/2014)

Jason Bau

## What is it?

This service, when available to XBlocks, returns common information about users such as email address, full name, and username. This has been a long requested feature that has almost been implemented several times, but thanks to the open edX conference hackathon there is now merged code for this in the xblock repo (<https://github.com/edx/XBlock/pull/273/files>) and an active PRs in edx-platform (<https://github.com/edx/edx-platform/pull/6013/files>).

A PR to xblock-sdk should happen shortly after Thanksgiving.

## What information does it expose?

Reference: <https://github.com/edx/XBlock/blob/master/xblock/reference/user_service.py>

As currently implemented, the service only provides information for the currently logged-in user (get\_current\_user()), though it can be extended to add information about arbitrary users later.

The user information returned by get\_current\_user()is packaged in an python object that is guaranteed to contain the following data attributes, whose values may be None, as can happen with AnonymousUsers

class XBlockUser(object):

def \_\_init\_\_(self, \*\*kwargs):  
 # Set standardized attributes  
 self.is\_authenticated = kwargs.get('is\_authenticated')  
 self.email = kwargs.get('email')  
 self.full\_name = kwargs.get('full\_name')  
 self.username = kwargs.get('username')  
 self.user\_id = kwargs.get('user\_id')

## This data is Personally Identifiable. How are you handling that?

After some discussion at the hackathon between Ned, some other community members, and myself, we decided

1. This service should explicitly return PII.
2. This service should not return any user ids meant to be anonymized (pseudonymized)

The rationale here is that since xblocks currently execute in-process and can in fact import django, it’s impossible right now to prevent malicious xblocks from accessing PII anyway. So instead we wanted to prepare for future where xblock execution is isolated and user data is only available via runtime services. Considering that it is likely easier for a runtime to control an xblock’s access to distinct individual services rather than individual functionality within services, we decided that this service shouldn’t mix PII and anonymized identifiers, since we can envision PII user data and anonymized user data being offered as 2 different services in the future, with different access control rationale. This separation of services also makes it somewhat harder for a malicious xblock to deanonymize a large set of students.

The current PRs only offer the PII service, which is what has been in demand by XBlock authors.

## What refactoring of similar services has been done?

There was a StudioUserService which provided a single field, user\_id, to common/lib/xmodule/xmodule/split\_test\_module.py. After discussing with Andy Armstrong I’ve replaced that service with the new one and changed the call within split\_test\_module.py

## How’s it work?

The service code itself, with the implementation of get\_current\_user()and a converter function between Django and XBlock user objects, lives in common/djangoapps.

In lms, the service is instantiated with request.user in courseware/module\_render.py

In studio, the service is instantiated with request.user in contentstore/views/preview.py

In modulestore, the service doesn’t have an easy way to get request.user so it relies on the django-crum library (<https://pypi.python.org/pypi/django-crum>) already present in edx-platform to get the current user, and then instantiates the service.