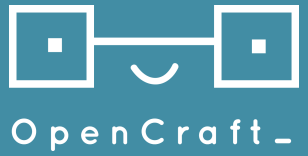
Building an XBlock



Eugeny Kolpakov
eugeny@opencraft.com

Tim Krones
tim@opencraft.com

Agenda

Agenda

- Setting up development environment
- Scaffolding project using XBlock cookiecutter template
- Setting up and configuring development tools
  - Dependency management (pip-tools)
  - Tests (pytest and tox)
  - Static code analysis, code style checks (pycodestyle, pylint, isort)
- XBlock development
  - XBlock Workbench integration
  - XBlock fields and field scopes
  - Views
  - Action handlers
  - Installing XBlock into devstack

Dev environment setup

Development environment

- Install python and pip
  - Debian/Ubuntu: *sudo apt-get install python-pip*
  - Mac: *brew install python pip*
  - Windows: http://bit.ly/20T7SZA
- Install virtualenv
  - Debian/Ubuntu/Mac: *(sudo) pip install virtualenv*
  - Windows: same link
- Create virtualenv for the project
  - All systems: *virtualenv env_name*
- Activate virtualenv
  - Debian/Ubuntu/Mac: *source venv/bin/activate*
  - Windows: *venv\Scripts\activate*
- Install cookiecutter
  - All systems: *pip install cookiecutter*

Baking the project

Baking the project

- *cookiecutter https://github.com/edx/cookiecutter-xblock.git*
- Asks some questions:
  - **project_desc** - short description of your XBlock
  - **package_name** - the name of python package for your XBlock; **Caveat:** No dashes, no spaces
  - **repo_name** - the name of git repository for your XBlock
  - **tag_name** - computer-readable name of your XBlock
  - **class_name** - name of python class for your XBlock
- Creates <repo_name> directory in current working directory
  - **setup.py** - script that installs your block into python environment
  - **Makefile** - some helpful automation commands, i.e. *make dev.run*
  - **Dockerfile** - instructions on building Docker image for the XBlock
  - <**package_name**> - main contents of your XBlock

Version control

- Running *cookiecutter* does not create a git repository
- Run the following commands to create one yourself:

  cd <repo-name>

  git init

- This step is fine to skip if you don’t have git installed

Initial fixes

- Apply changes from https://github.com/open-craft/quote-of-the-day-xblock/pull/2
- Commit them


Development tools

Dependency management

- Two mechanisms to do dependency management:
  - setup.py - manages installation when your package is “transient dependency”, i.e. when installing an XBlock into workbench/edx-platform runtime
  - requirements files - manages installation when your package is installed in development environment and tries to achieve “repeatable installs”
- Pip - python package manager
- Pip-tools - extensions to pip to address some shortcomings:
  - Hard to keep dependency packages up-to-date; almost impossible for transient dependencies
  - Hard to remove stale dependencies (except destroying virtualenv and installing fresh)
- Pip-tools workflow:
  - Declare dependencies in *requirements/*.in* files
  - Compile dependencies list into *requirements/*.txt* files using *pip-compile requirements/*.in*
  - Install dependencies using *pip-sync requirements/*.txt*

Installation

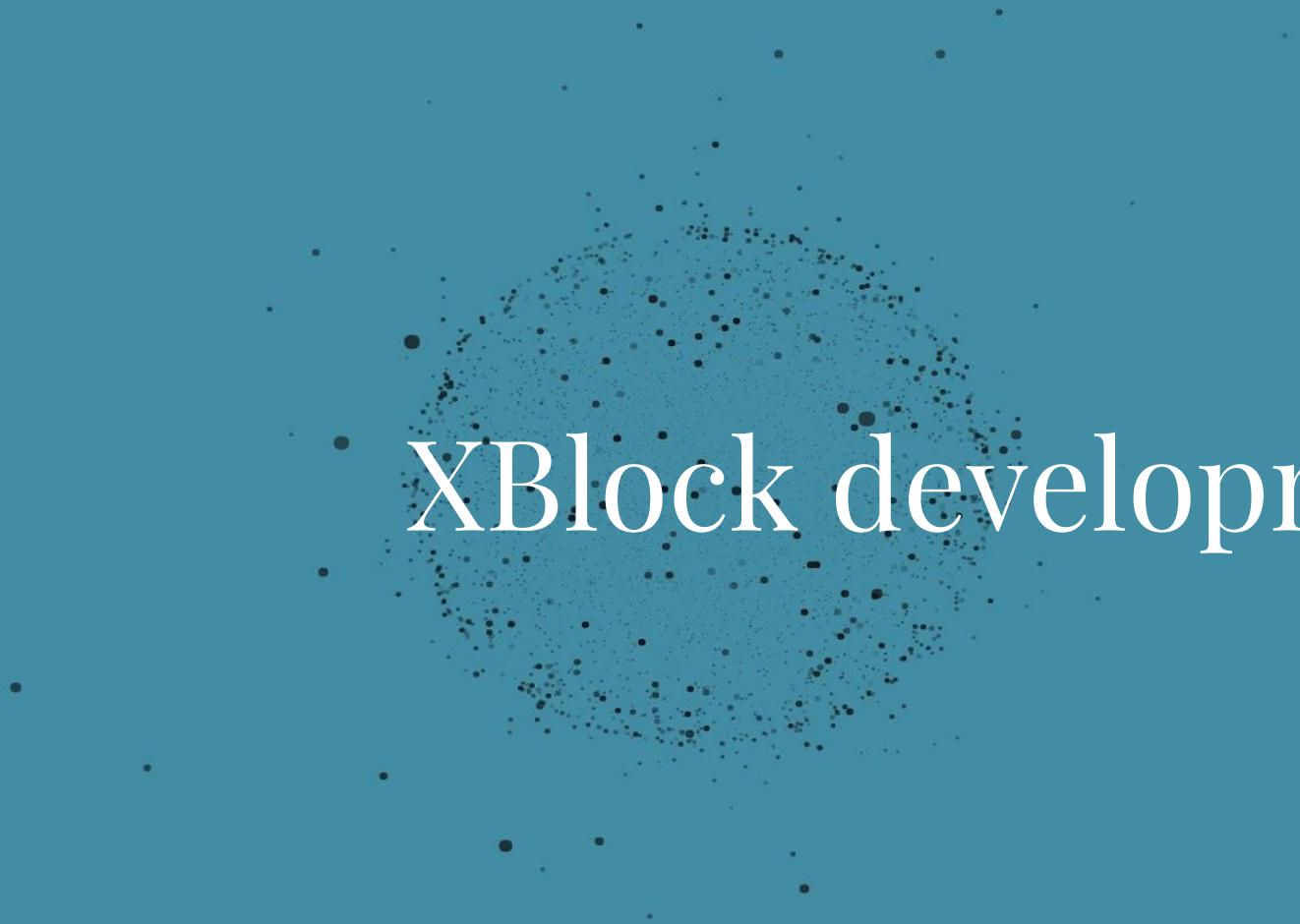
- Apply changes from https://github.com/open-craft/quote-of-the-day-xblock/pull/3
- Make sure to use **tabs** for indentation in Makefile, **not spaces**!
- Run *make dev.update* to install development tools

Tests

- Unit and integration tests
  - Unit tests - class/module is tested in isolation, all dependencies are mocked/stubbed
  - Integration tests - workflows are tested as a whole, real dependencies (i.e. actual DB)
- Pytest - python test framework
  - Documentation: https://docs.pytest.org/en/latest/
  - At a glance: name functions/classes/modules with *test_* prefix, assert using *assert x == y*
- Tox - python test automation tools
  - Runs tests in different environments, i.e. python 2.7 and python 3.5
  - tox.ini can be used to provide configuration for many other tools (i.e. pylint, pycodestyle etc.)
- The result should resemble https://github.com/open-craft/quote-of-the-day-xblock/pull/4

Static code analysis and code style tools

- These tools analyse your code without actually running it
- Detect common errors and caveats
- Code style tools enforce coding standards:
  - Tabs vs spaces (where applicable)
  - Formatting
  - Naming conventions
  - And so on
- Pycodestyle (former pep8) - checks if code conforms to PEP8 guidelines
- Pylint - static code analyzer; detects potential problems, also does code style checks
- isort - tiny tool to make sure import statements are sorted
- The result should resemble https://github.com/open-craft/quote-of-the-day-xblock/pull/5



XBlock development

Goal: develop a sample XBlock and integrate it with devstack

Sample XBlock

To illustrate different aspects of a XBlock, we will implement sample XBlock that will pull random quotes from a 3rd-party quotes API

Features:

- Displays random quote each time it is shown on the page
- Allows user to “star” quotes - those will be remembered and always displayed
- (if time allows) Allow course authors to configure API URL and parameters.

Reference implementation:
https://github.com/open-craft/quote-of-the-day-xblock

XBlock workbench

- Simple XBlock runtime for development and testing your blocks
- Slightly different from actual edx-platform runtime, but it won't affect us
- Dockerfile contains instructions on creating a workbench box
  - … but we'll have to extend them a bit to have our XBlock actually available there
- Useful for development and testing
  - Some of more sophisticated XBlocks use workbench to run integration tests
- Scenarios - XML snippets specifying XBlocks and their settings
  - Our XBlock already contains two scenarios: the most basic one and one with multiple instances of the block on the same page
- Caveat: comes pre-bundled with a couple of simple XBlocks and scenarios - don't get confused

XBlock fields and field scopes

- Fields are attributes of your XBlock
  - Different data types: String, Integer, DateTime etc.
  - Also contain meta information: help text, description etc.
- Scopes specify what **kind** of attribute the field is
  - Docs on scopes
- Simply put there are two major groups of fields:
  - Content and settings - provided by course authors, same for each student, exported with the block
  - Student data - provided by the student (i.e. answers), different between students

Views

- Views are instructions to render your XBlock
- Predefined views:
  - student_view - how block is presented to student
  - studio_view - confusing name: editor interface presented to course author
  - author_view - how block is presented to author in Studio (optional, defaults to student_view)
- Ok to define custom views, but runtime won't know how to use them - so those views should be called from one of the predefined views
- Fragments - chunks of HTML+CSS+JS code to be rendered
  - Used to be part of XBlock package, recently moved to dedicated python package

Action handlers

- Most XBlocks need to react to student actions
  - Most if not all are AJAX calls
- Action handlers are methods on the XBlock that handle answering those calls
  - XBlock.handler - decorator for basic action handler (just marks a method as action handler)
  - XBlock.json_handler - automatically parses request body as JSON and formats return value as JSON
- XBlock frontend code will need to know where those handlers are:
  - runtime.handlerUrl("handler_method_name") - returns URL of the handler
- Side note: to support editing XBlock in Studio a handler must be present
  - However, this was tedious to replicate in each XBlock - we (OpenCraft) created a couple of xblock-utils helpers to automate it (but it implies using Django template engine)
  - https://github.com/edx/xblock-utils/blob/master/xblockutiltemplates/studio_editable.py

Installing XBlock into devstack

- Development install - good for developing (faster feedback cycle)
  - Set up shared directory in Vagrantfile (i.e. /home/you/xblocks => /edx/xblocks)
  - SSH into devstack
  - *sudo su edxapp*
  - *pip install -e /edx/xblocks/your_xblock*
- Production install - good for deployment
  - Add XBlock to requirements/custom.txt
  - SSH into devstack
  - Run *paver install_prereqs* (make sure to have NO_PREREQ_INSTALL env var unset)
- Using in the course:
  - Remember that tag_name parameter? This is XBlock's tag (if you don't remember it - look it up in setup.py entry_points)
  - In Studio: Course Settings => Advanced Settings => Advanced XBlocks; add XBlock tag to the list and save
  - XBlock will appear in "Advanced" menu in course outline

Thanks!

Eugeny Kolpakov
eugeny@opencraft.com

Tim Krones
tim@opencraft.com