SCIOER Course Builder Guide

SCIOER Content

A SCIOER resource can contain five different kinds of content: - A wiki - Pre recorded video lectures - Practice problems and solutions - Interactive tutorials - Official documentation for a programming language if the resource is a programming course.

The wiki is the central organizing mechanism for the course that provides links to the other resource types as appropriate.

The wiki, practice problems, videos and interactive tutorials must be created by the OER author before the SCIOER software can create the final SCIOER resource. The wiki, practice problems and interactive tutorials must be housed in a git repository that the software will access during the creation process. The video lectures may be housed in a directory on the computer being using to create the SCIOER resource.

Detailed information about content creation is provided later in this guide.

The SCIOER Builder software

`scioer-builder` is a python program that allows an instructor to build a SCIOER open education resource without the need to learn anything about defining docker images.

This user guide describes the builder, how to use it, and what the different components of a SCIOER resource are.

Installation

SCIOER resources are designed to be used in technology-heavy courses, such as programming courses. As a result, a certain amount of technology is required to create the resources. You will need a computer capable of running Docker and will also need a recent version of Python.

Prerequisites

- All of the SCIOER resources require the installation of Docker Desktop
- Python 3.7 or newer
- Pip (python package manager)

Installing SCIOER Builder

The SCIOER builder software is installed using the python package manager pip

- ensure that Docker is running
- type `pip install scioer-builder`
- if you have python2 and python3 installed
  - `pip3 install scioer-builder`

- when the install finishes you may test the install by typing `scioer-builder --help` into a terminal

Scioer builder

Before running the software, the course content should be prepared and stored in git repositories for scioer-builder to access.

Prerequisites

- Wiki, tutorial and example content must be in git repositories
  - repository must be public access or must be configured to allow access via ssh key
- Video lectures are gathered from a local directory to avoid trying to upload large files to a git repository

If you wish to push your container to dockerhub for easy distribution to your students you must: - have a dockerhub account - be logged into the dockerhub account locally and logged into the correct registry - be building an image that belongs to you / is avalible / you have permissions to push to

note: be careful to only reuse image names and tags if you intend to write over the previously existing image

Working with the builder

The scioer-builder provides two ways to construct a SCIOER resource. 1. You can follow the interactive cli by entering `scioer-builder --interactive` which will walk you through a series of prompts to source course content. - the interactive prompts allow for only the most common features 1. course content locations and other features can be provided via flags to the scioer-builder.

Each course is created as a layer on top of a base container for a specific purpose. At this time base containers have been created for the java programming language, the c programming language and the SQL programming language.

Interactive Mode

- Ensure that Docker is running
- Type `scioer-builder --interactive` into a terminal
  - default responses will be capitalized [Y/n]
  - in most cases the default values will be the correct choice.
- Information about the container to be built.
  - Enter the name of the image to be built, including the tag: [e.g. scioer/oo-java:latest]
  - Enter the base image that should be used for this course: [e.g. scioer/java-resource:latest]
  - Should the latest base image be fetched? [Y/n]
  - Do you want to push the generated image to dockerhub? [y/N]
- General options for content git repositories.
  - Should the SSH host keys be verified? [Y/n]
  - Should the git histories be kept after they are cloned? [y/N]
  - Enter the path to the SSH private key used to clone the git repos if one is being used:
- Information about the wiki to be created.
  - Enter the title for the Wiki:
  - Enter the git repository URL for the wiki:
  - Enter the branch name for the wiki repository: [main]
  - Do you want to clone the wiki repository using ssh? [Y/n]
  - Should the SSL certificates be verified when cloning the wiki? [Y/n]

- Where the rest of the content is being loaded from.
    - Enter the git repository that holds the Jupyter Notebook files (leave blank if not being used)
    - Enter the git repository that holds the pre-recorded lecture files (leave blank if not being used)
    - Enter the directory that contains the pre-recorded lecture files (leave blank if not being used)
    - Enter a git repository that contains examples and worked problem sets (leave blank if not being used)

Using Flags

The entire list of flags available to use with scioer-builder can be found by typing `scioer-builder --help` on a command line. To build a resource using flags, first ensure that docker is running. Then run the program by typing `scioer-builder` and include any desired flags in the same command.

The command and flags used to create the scioer/oo-java-course are shown below.

```
scioer-builder --jupyter-repo=https://github.com/sci-oer/oo-course-tutorials.git  --example=https://git
```

Content Creation Details

A SCIOER container can contain the following components: * Video lecture content * Example problems with solutions * A wiki using wiki.js * Interactive tutorials using JupyterLab

Video Lecture Content

At this time, sci-oer builder only provides a basic web server to serve the video lectures. You'll can set up a basic html page to display lectures to the user, or you can create a more sophisticated viewer. The oo-java-course and the c-programming-course provide the video file, the transcript, and a separate audio file for each prerecorded lecture.

Video lectures are hosted within the SCIOER resource at `localhost:8000/lectures` followed by whatever file/folder hierarchy you design. The wiki can provide links to the lectures using that URI if you wish.

Lecture content video files can also be hosted on an external web server and served through a reverse proxy in the image instead to reduce the image size at the cost of requiring internet access and a separate video hosting server. The `--static-url` flag indicates the location of the external server.

Video lectures are the only course materials that are typically not housed in a git repository. Instead, provide scioer-builder the path to a local directory.

Example Project Repositories

Example projects are used to provide students with projects for them to explore. Project can have complete, partial, or no solutions. The projects are not available via the wiki but are available through the mounted directory on the student's host computer as well as from inside the SCIOER container.

- practice problems are provided as a git repository to the sci-oer builder
- you may include as many different sets of practice problems as you wish
- Accessible via the shell or the mounted drive
- placed in a directory titled "practiceProblems"

Wiki Content

The wiki is served by wiki.js The wiki is the e-textbook portion of the resource and also usually contains links to the other types of the resources. The wiki is accessed from a running SCIOER course by pointing a browser at `http://localhost:3000`

The SCIOER content is imported into wiki.js from a set of markdown files provided by the content creator. Wiki.js produces the most pleasing result when the markdown files and directory structure match the default used by the system..

Wiki Directory Structure

Wikijs prefers that subdirectories have a markdown file with the same name that is contained in the parent directory. You can use other structures but the breadcrumbs at the top of the pages will not work properly.

For example, consider a wiki that includes two chapters: firstTopic, secondTopic. Each chapter contains 3 subtopics: firstSubOne, firstSubTwo, firstSubThree, secondSubOne, secondSubTwo, secondSubThree

The preferred file/folder structure looks like this.

```
firstTopic
firstTopic.md
secondTopic
secondTopic.md

|-firstTopic
firstSubOne.md
firstSubThree.md
firstSubTwo.md

|-secondTopic
secondSubOne.md
secondSubThree.md
secondSubTwo.md
```

If you do not wish to write markdown files manually, you can install wiki.js, create the wiki using the built in editor, and export the entire wiki to markdown when you are done.

Interactive Tutorials

JupyterLab is a web-based interactive development environment often used for data science. A SCIOER resource employs JupyterLab to provide interactive tutorials for students. A Jupyter notebooks can contain blocks of executable code in a variety of languages with interspersed comments, which facilitates interactive explaination and demonstration of .

You can experiment with Jupyter online and even use the online development environment to create your Jupyter Tutorials for your SCIOER resource.

The jupyter notebooks for a SCIOER resource must be housed in a git repository. The full set of notebooks will be available to learners at `http://localhost:8888`. In most cases you will also want to provide links to individual notebooks in the wiki pages.

Programming Language documentation

If you are developing a SCIOER resource for a specific programming language, the help pages for the language can be included in the container. For example, the javadoc documentation that comes with the Java programming language is included with the oo-java-course image. The language documentation is available at `http://localhost:8000/docs`

The documentation is included in the base container for the programming language. The most up to date list of available base reources can be found at https://hub.docker.com/u/scioer/. If you can't find the base-resource you need, please contact us at `info@scioer.ca` and we'll help you create an appropriate base.

Using Your SCIOER Resource

The docker image that `scioer-builder` creats will be a fully interactive e-text containg the information you provide.

At this time the mapping of the ports in the container cannot be changed so you will have to have the following ports free in order to run the container: 3000, 8000, 8888

The container can be run from the command line easily. A volume will be mounted in the directory from which the command is run. On the command line type `docker run -p 3000:3000 -p 8888:8888 -p 8000:8000 -v "$(pwd)/course:/course" -it <THE_NAME_OF_YOUR_IMAGE>`

For example, to run the oo-java-course container from the command line, one would type: `docker run -p 3000:3000 -p 8888:8888 -p 8000:8000 -v "$(pwd)/course:/course" -it scioer/oo-java:W23`

When the container is running you can point a web browser at the different ports to see your resources. - wiki: (http://localhost:3000) - programming language documentation (http://localhost:8000/docs) - prerecorded lectures (http://localhost:8000/lectures) - interactive tutorials(http://localhost:8888)

Because every student has their own copy of this resource, the need for any sort of password protection is low. The scioer-builder sets the password as "password" for the tutorials and the wiki. The login name for the wiki is admin@example.com.

The scioer command line interface

SCIOER provides a python program for students that eliminates the need for them to remember the command for running the SCIOER resource. You can also use that program to run your resource if you wish. More information about the student CLI can be found here: https://github.com/sci-oer/student-cli