innoConv Documentation

Release 0.0.2

Innocampus

Mar 13, 2019

Contents

1 Table of contents **1**

1.1 What is innoConv? . . . 1
1.1.1 Demo course . . . 1
1.1.2 Course structure . . . 1
1.1.3 Viewers . . . 1
1.2 Getting started . . . 2
1.2.1 Prerequisites . . . 2
1.2.2 Dependencies . . . 2
1.2.2.1 Python 3.7 . . . 2
1.2.2.2 Pandoc . . . 2
1.2.3 Installation . . . 2
1.2.3.1 Using pip . . . 2
1.2.3.2 In a virtual environment . . . 2
1.3 How to use `innoconv` . . . 2
1.3.1 Command line arguments . . . 3
1.3.1.1 Positional Arguments . . . 3
1.3.1.2 Named Arguments . . . 3
1.4 Module overview . . . 3
1.4.1 innoconv.constants . . . 3
1.4.2 innoconv.runner . . . 3
1.4.3 innoconv.extensions . . . 4
1.4.4 innoconv.extensions.abstract . . . 4
1.4.5 innoconv.extensions.copy_static . . . 5
1.4.5.1 Translation . . . 5
1.4.5.2 Relative and absolute reference . . . 5
1.4.6 innoconv.extensions.generate_toc . . . 6
1.4.7 innoconv.extensions.join_strings . . . 6
1.4.7.1 Example . . . 6
1.4.8 innoconv.extensions.write_manifest . . . 7
1.4.9 innoconv.manifest . . . 7
1.4.9.1 Example . . . 7
1.4.10 innoconv.utils . . . 8

2 Indices and tables **9**

Python Module Index **11**

CHAPTER 1

Table of contents

1.1 What is innoConv?

innoConv is a converter for educational content. It is used to turn content into a intermediate format that can be used with a content viewer.

1.1.1 Demo course

TODO

tub_base

1.1.2 Course structure

- `toc.md`
- language directories - chapters and sections in sub-directories - `content.md`

1.1.3 Viewers

At the moment there are two viewers in development.

innodoc-webapp A React-based HTML5 web app. (Website)

innodoc-app A React Native-based Smartphone App (Website)

1.2 Getting started

1.2.1 Prerequisites

innoConv is mainly used on Linux machines. It might work on Mac OS and Windows/Cygwin/WSL. You are invited to share experiences in doing so.

1.2.2 Dependencies

The only dependencies you have to provide yourself is Pandoc and the Python interpreter.

1.2.2.1 Python 3.7

While other versions of Python might work, innoConv was tested with **Python 3.7**. Make sure you have it available.

1.2.2.2 Pandoc

You need to make sure to have a recent version of the `pandoc` binary available in `PATH` (**Pandoc 2.2.1** at the time of writing). There are several ways on installing Pandoc.

1.2.3 Installation

1.2.3.1 Using pip

TODO

1.2.3.2 In a virtual environment

It's possible to install innoConv into a virtual environment. Setup and activate a virtual environment in a location of your choice.

```
$ python3 -m venv /path/to/venv
$ source /path/to/venv/bin/activate
```

Install innoConv in your virtual environment using pip.

```
$ pip install innoconv
```

If everything went fine you should now have access to the `innoconv` command.

The next time you login to your shell make sure to activate your virtual environment before using `innoconv`.

1.3 How to use `innoconv`

Run the converter in your content directory.

```
$ innoconv .
```

This will trigger the conversion for this content folder.

1.3.1 Command line arguments

```
usage: innoconv [-h] [-o OUTPUT_DIR] [-v] [-e EXTENSIONS] source_dir
```

1.3.1.1 Positional Arguments

source_dir Content source directory

1.3.1.2 Named Arguments

-o, --output-dir Output directory (default: ./innoconv_output)

Default: "./innoconv_output"

-v, --verbose Print verbose messages (default: False)

Default: False

-e, --extensions Enabled extensions (default: join_strings,copy_static,generate_toc,write_manifest)

Default: "join_strings,copy_static,generate_toc,write_manifest"

1.4 Module overview

1.4.1 innoconv.constants

Project constants.

innoconv.constants.**DEFAULT_OUTPUT_DIR_BASE**
Default innoconv output directory

innoconv.constants.**DEFAULT_EXTENSIONS**
Default enabled extensions

innoconv.constants.**ENCODING**
Encoding used in this project

innoconv.constants.**CONTENT_BASENAME**
Basename for the content file in a section

innoconv.constants.**MANIFEST_BASENAME**
Manifest filename

innoconv.constants.**STATIC_FOLDER**
Static folder name

1.4.2 innoconv.runner

The innoConv runner is the core of the conversion process.

It traverses the source directory recursively and finds all content files. These are converted one-by-one to JSON. Under the hood is uses Pandoc.

It receives a list of extensions that are instantiated and notified upon certain events. The events are documented in *AbstractExtension*.

class innoconv.runner.**InnoconvRunner**(*source_dir*, *output_dir*, *manifest*, *extensions*)
Convert content files in a directory tree.

run()
Start the conversion by iterating over language folders.

1.4.3 innoconv.extensions

innoConv extensions.

Extensions are a way of extending the functionality of innoConv in a modular way. They can be enabled on a one-by-one basis.

innoconv.extensions.**EXTENSIONS = {'copy_static': <class 'innoconv.extensions.copy_static.C**
List of available extensions

1.4.4 innoconv.extensions.abstract

Base class for all other extensions.

The AbstractExtension is not meant to be instantiated directly.

class innoconv.extensions.abstract.**AbstractExtension**(*manifest*)
Abstract class for extensions.

The class all extensions inherit from. The to-be-implemented methods document the available events that are triggered during the conversion process.

Extension classes should have a `_helptext` attribute. It is shown in the CLI as a brief summary what the extension accomplishes.

classmethod helptext()
Return a brief summary of what the extension is doing.

start(*output_dir*, *source_dir*)
Conversion is about to start.

Parameters

- **output_dir** (*str*) – Base output directory
- **source_dir** (*str*) – Content source directory

pre_conversion(*language*)
Conversion of a single language folder is about to start.

Parameters **language** (*str*) – Language that is currently being converted.

pre_process_file(*path*)
Conversion of a single file is about to start.

Parameters **path** (*str*) – Output path

post_process_file(*ast*, *title*)
Conversion of a single file finished. The AST can be modified.

Parameters

- **ast** (*List of content nodes*) – File content as parsed by pandoc.
- **title** (*str*) – Section title (localized)

post_conversion(*language*)
Conversion of a single language folder finished.

Parameters **language** (*str*) – Language that is currently being converted.

finish()
Conversion finished.

1.4.5 innoconv.extensions.copy_static

Extension that copies static files.

Content can include figures, images and videos. Static files can be included in a special folder named `_static`. The files will be copied to the output directory automatically.

1.4.5.1 Translation

It's possible to have language-specific versions of a static file.

For that to work you need to have a `_static` folder beneath the language folder. Files in this folder will take precedence over the common `_static` folder for that language.

Example: `en/_static/example.png` takes precedence over `_static/example.png` in the English version.

1.4.5.2 Relative and absolute reference

Files can be referenced using relative or absolute paths.

Absolute paths are resolved to the root folder, either the common (`_static`) or language-specific (`en/_static`) folder.

Relative paths are resolved to the root folder but have the chapters path fragment appended.

Example

This example shows how a reference to an image is resolved. The references happen inside the section `chapter01` in the English language version.

Type	Reference	Resolved to
Relative	`subdir/my_picture.png`	`en/_static/chapter01/subdir/my_picture.png`
Absolute	`/subdir/my_picture.png` (with leading `/`)	`en/_static/subdir/my_picture.png`

class innoconv.extensions.copy_static.**CopyStatic**(**args*, ***kwargs*)
Bases: *innoconv.extensions.abstract.AbstractExtension*

Copy static files to the output folder.

This extension copies checks the AST for references to static files and copies them from the content source directory to the output directory.

start(*output_dir*, *source_dir*)
Remember directories.

pre_conversion(*language*)
Remember current conversion language.

pre_process_file(*path*)
Remember file path.

post_process_file(*ast*, _)
Generate list of files to copy.

finish()
Copy static files to the output folder.

1.4.6 innoconv.extensions.generate_toc

Extension that generates a table of contents.

A table of contents is generated from the course sections and added to the *Manifest*.

class innoconv.extensions.generate_toc.**GenerateToc**(**args*, ***kwargs*)
Bases: *innoconv.extensions.abstract.AbstractExtension*

Generate a TOC from content sections.

start(*output_dir*, _)
Remember output directory.

pre_conversion(*language*)
Remember current conversion language.

pre_process_file(*path*)
Remember current path.

post_process_file(_, *title*)
Add this section file to the TOC.

1.4.7 innoconv.extensions.join_strings

Merge consecutive sequences of strings and spaces into a single string element.

The motivation behind this extension is to make the AST more readable and also to save space by compressing the representation. The actual appearance in a viewer should remain completely untouched.

This extension modifies the AST.

1.4.7.1 Example

Before	`{"t": "Str", "c": "Foo"},{"t": "Space"},{"t": "Str", "c": "b!"}]`
After	`{"t": "Str", "c": "Foo b!"}]`

class innoconv.extensions.join_strings.**JoinStrings**(**args*, ***kwargs*)
Bases: *innoconv.extensions.abstract.AbstractExtension*

Merge consecutive strings and spaces in the AST.

post_process_file(*ast*, _)
Process AST in-place.

1.4.8 innoconv.extensions.write_manifest

Extension that writes a `manifest.json` to the output folder.

Every course needs a *Manifest*. Additionally to the fields from the source manifest it can include a table of contents and a glossary.

class innoconv.extensions.write_manifest.**WriteManifest**(**args*, ***kwargs*)
Bases: *innoconv.extensions.abstract.AbstractExtension*

Write a manifest file when conversion is done.

start(*output_dir*, _)
Remember output directory.

finish()
Output course manifest.

1.4.9 innoconv.manifest

The manifest comprises course metadata.

A `manifest.yml` file needs to exist in every course content and resided at the content root directory. It is usually written by hand.

There is also a representation in the JSON format. It is generated automatically by the converter and additionally includes the table of contents that is generated from the section structure. It can be found in the output folder.

1.4.9.1 Example

```
title:
  en: Example title
  de: Beispiel-Titel
languages: en,de
```

class innoconv.manifest.**Manifest**(*data*)
Represents course metadata.

classmethod from_yaml(*yaml_data*)
Create a manifest from YAML data.

Parameters **yaml_data** (*str*) – YAML representation of a manifest

class innoconv.manifest.**ManifestEncoder**(*, *skipkeys=False*, *ensure_ascii=True*, *check_circular=True*, *allow_nan=True*, *sort_keys=False*, *indent=None*, *separators=None*, *default=None*)
JSON encoder that can handle Manifest objects.

default(*o*)
Return dict for Manifest objects.

1.4.10 innoconv.utils

Utility module.

innoconv.utils.**to_ast**(*filepath*)

Convert a file to abstract syntax tree using pandoc.

Parameters **filepath** (*str*) – Path of file

Return type (list of dicts, str)

Returns (Pandoc AST, title)

Raises

- **RuntimeError** – if pandoc exits with an error
- **ValueError** – if no title was found

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

i

innoconv.constants, 3
innoconv.extensions, 4
innoconv.extensions.abstract, 4
innoconv.extensions.copy_static, 5
innoconv.extensions.generate_toc, 6
innoconv.extensions.join_strings, 6
innoconv.extensions.write_manifest, 7
innoconv.manifest, 7
innoconv.runner, 3
innoconv.utils, 8

Index

A

AbstractExtension (*class in innoconv.extensions.abstract*), 4

C

CONTENT_BASENAME (*in module innoconv.constants*), 3
CopyStatic (*class in innoconv.extensions.copy_static*), 5

D

default() (*innoconv.manifest.ManifestEncoder method*), 7
DEFAULT_EXTENSIONS (*in module innoconv.constants*), 3
DEFAULT_OUTPUT_DIR_BASE (*in module innoconv.constants*), 3

E

ENCODING (*in module innoconv.constants*), 3
EXTENSIONS (*in module innoconv.extensions*), 4

F

finish() (*innoconv.extensions.abstract.AbstractExtension method*), 5
finish() (*innoconv.extensions.copy_static.CopyStatic method*), 6
finish() (*innoconv.extensions.write_manifest.WriteManifest method*), 7
from_yaml() (*innoconv.manifest.Manifest class method*), 7

G

GenerateToc (*class in innoconv.extensions.generate_toc*), 6

H

helptext() (*innoconv.extensions.abstract.AbstractExtension class method*), 4

I

innoconv.constants (*module*), 3
innoconv.extensions (*module*), 4
innoconv.extensions.abstract (*module*), 4
innoconv.extensions.copy_static (*module*), 5
innoconv.extensions.generate_toc (*module*), 6
innoconv.extensions.join_strings (*module*), 6
innoconv.extensions.write_manifest (*module*), 7
innoconv.manifest (*module*), 7
innoconv.runner (*module*), 3
innoconv.utils (*module*), 8
InnoconvRunner (*class in innoconv.runner*), 3

J

JoinStrings (*class in innoconv.extensions.join_strings*), 6

M

Manifest (*class in innoconv.manifest*), 7
MANIFEST_BASENAME (*in module innoconv.constants*), 3
ManifestEncoder (*class in innoconv.manifest*), 7

P

post_conversion() (*innoconv.extensions.abstract.AbstractExtension method*), 4
post_process_file() (*innoconv.extensions.abstract.AbstractExtension method*), 4
post_process_file() (*innoconv.extensions.copy_static.CopyStatic method*), 6
post_process_file() (*innoconv.extensions.generate_toc.GenerateToc method*), 6

post_process_file() (*innoconv.extensions.join_strings.JoinStrings method*), 6
pre_conversion() (*innoconv.extensions.abstract.AbstractExtension method*), 4
pre_conversion() (*innoconv.extensions.copy_static.CopyStatic method*), 6
pre_conversion() (*innoconv.extensions.generate_toc.GenerateToc method*), 6
pre_process_file() (*innoconv.extensions.abstract.AbstractExtension method*), 4
pre_process_file() (*innoconv.extensions.copy_static.CopyStatic method*), 6
pre_process_file() (*innoconv.extensions.generate_toc.GenerateToc method*), 6

R

run() (*innoconv.runner.InnoconvRunner method*), 4

S

start() (*innoconv.extensions.abstract.AbstractExtension method*), 4
start() (*innoconv.extensions.copy_static.CopyStatic method*), 5
start() (*innoconv.extensions.generate_toc.GenerateToc method*), 6
start() (*innoconv.extensions.write_manifest.WriteManifest method*), 7
STATIC_FOLDER (*in module innoconv.constants*), 3

T

to_ast() (*in module innoconv.utils*), 8

W

WriteManifest (*class in innoconv.extensions.write_manifest*), 7