Stellar

Courseware environment
Documentation

Paolo Bettelini

Contents

Part I

Stellar

1 Introduction

2 Data structure

```
data/
├── snippets/
├── pages/
├── courses/
└── universes/
```

3 PDF Stellar Format

3.1 Format commands

Stellar is able to interpret special PDFs with certain text commands written in the page. This is referred to as `PDF Stellar Format`. The file represents a Stellar page containing snippets, or just a collection of snippets. Stellar will crop the PDF into multiple pieces to generate every snippet individually, and is also able to generate the corresponding page containing them.

PDF Stellar Format commands	
Command	**Description**
`!id <ID>`	Set the ID of this page.
`!snippet <ID>`	Start a PDF snippet here.
`!endsnippet <JSON meta>`	End the current snippet here. The JSON metadata is optional.
`!gen-page <bool>`	Set to true to generate the HTML page for this PDF.
`!include <ID> <Params>`	Include a snippet here given its ID. The parameters string is optional.
`!plain`	Write plain HTML to the page here.
`!section`	Add a level 1 heading `<h1>`.
`!subsection`	Add a level 2 heading `<h2>`.
`!subsubsection`	Add a level 3 heading `<h3>`.

Furthermore, it is possible to reference another snippet by adding an annotation with a link of the form `/snippet/snippet-id` or `/snippet/snippet-id|Label`.

Stellar will crop the PDF between `!snippet` amd `!endsnippet`.

3.2 LaTeXPackage

The file `stellar.sty` provides the LaTeXpackage `stellar`, which implements PDF Stellar Format.

3.2.1 Usage

In order to use the package is it necessary to use the following documentclass:

```
\documentclass[preview]{standalone}
```

and import the `stellar` package

```
\usepackage{stellar}
```

The command `\title` needs to be inserted within the `document`. The commands `\section`, `\subsection` and `\subsubsection` can be used normally as their behavior is overwritten by the package.

The commands `\id`, `\genpage` and `\plain` are available.

It is possible to reference another snippet by using the `\snippetref` command:

```
\snippetref[snippet-identifier][Some text].
```

A snippet can be included using the following command:

```
\includesnpt{snippet-identifier}
\includesnpt[param1=value1|param2=value2]{snippet-identifier}
```

Snippets can be created using environments. A plain snippet can be created as follows:

```
\begin{snippet}{snippet-identifier}
    % snippet content
\end{snippet}
```

There are other types of snippets such as the following

```
\begin{snippetdefinition}{snippet-definition1}{Definition 1}
    % snippet content
\end{snippetdefinition}

\begin{snippettheorem}{snippet-theorem1}{Theorem 1}
    % snippet content
\end{snippettheorem}

\begin{snippetproof}{snippet-proof1}{snippet-theorem1}{Proof 1}
    % snippet content
\end{snippetproof}
```

3.2.2 Example

```
\documentclass[preview]{standalone}

\usepackage{stellar}

\begin{document}

\id{page-identifier}
\genpage

\section{Section 1}

\subsection{Subsection 1}

\begin{snippetdefinition}{snippet1-definition}{Some definition}
    This is some definition
\end{snippetdefinition}

\subsection{Subsection 2}

\section{Section 2}

\begin{snippetdefinition}{snippet2-definition}{Some definition}
    This is some definition, the sequel
\end{snippetdefinition}

\includesnpt{snippet2-definition}

\begin{snippet}{text-snippet}[\{"meta": "json"\}]
    Some text as a snippet!
\end{snippet}

\end{document}
```

4 Metadata

A snippet may contain a `metadata.json` file containing some metadata properties.

4.1 JSON structure

Metadata fields		
Field	**Type**	**Description**
`generalizations`	Array of snippet IDs	Mathematical generalizations.
`requires`	Array of snippet IDs	Necessary libraries.
`default-params`	Params string	Default parameters for snippet.

4.2 Example

```
{
    "default-params": "width=70%|src=https://youtu.be/dQw4w9WgXcQ&",
    "requires": ["some-lib-snippet"],
}
```

Part II

LaTeX-driven implementation

5 Structure

```
source/
├── latex/
├── courses/
├── pages/
├── snippets/
└── universes/
```

6 Snippet build scripts

By default, snippets in the `snippets/` folder are just copied to the `data/` folder by the compiler. However, it is possible to add a custom build script `build.py` to the snippet directory to implement custom behavior. The build script will receive a single parameter representing the target folder where the files need to be written to.

6.1 Example

The following is an example build script to compile a Rust nannou project to wasm.

```
import sys
import subprocess
import os
import shutil

# This build script compiles a nannou project into a snippet

def main():
    target_folder = sys.argv[1]

    try:
        # Execute wasm-pack build --release
        subprocess.run(["wasm-pack", "build", "--release"], check=True)

        # Change directory to website
        os.chdir("website")

        # Execute npm install
        subprocess.run(["npm", "install"], check=True)

        # Execute npm run build
        subprocess.run(["npm", "run", "build"], check=True)

        # Generate snippet folder
        dist_folder = "dist"
        target_files = os.listdir(dist_folder)

        # Remove "dist/index.js"
        index_file_path = os.path.join(dist_folder, "index.js")
        if os.path.exists(index_file_path):
            os.remove(index_file_path)

        # Move all files from "dist" to target_folder
        for file_name in target_files:
            file_path = os.path.join(dist_folder, file_name)
            if os.path.isfile(file_path):
                shutil.move(file_path, target_folder)

        # Remove the "dist" directory
        shutil.rmtree(dist_folder, ignore_errors=True)

        print(f"Successfully moved files to {target_folder}")

    except subprocess.CalledProcessError as e:
        print(f"An error occurred while executing: {e.cmd}")
        print(f"Return code: {e.returncode}")
        sys.exit(1)
    except Exception as e:
        print(f"An unexpected error occurred: {str(e)}")
        sys.exit(1)

    if __name__ == "__main__":
        main()
```