entrinsicon

Publishing with Informer

How to programmatically create handbooks and a course catalog

Brian Gasperosky

Purveu, LLC

1. The Challenge
2. Requirements
3. The Solution
4. A Quick Demo
5. Tips, Tricks, and Gotchas

1. The Challenge

Create a process to automatically generate a course catalog or a handbook as both a website and a PDF.

1. The Challenge

- Multiple authors write content, course descriptions, curriculum charts, and policies
- The content is not static and can be updated anytime
  - The course catalog and handbooks must be updated quickly or synced automatically
- The content is stored in multiple databases
  - 25 Live
    - Academic Calendar
  - Colleague
    - Course Descriptions (CRSE)
    - Curriculum Charts (CUTK)
  - WordPress
    - Policies are stored in one site and managed by many authors
    - Remaining content is stored across several department sites and managed by many authors

2. Requirements

entrinsicon

Use Informer Templates to write WPML (XML) and TeX files.

2. Requirements

- The process
  1. Pull data from sources into datasets.
  2. Parse through content and write output files
     1. TeX
     2. WPML
  3. Import generated text files into destination applications
     1. LaTeX Document Preparation System (Overleaf)
     2. Content Management System (WordPress)

2. Requirements

entrinsicon

This presentation is about how to create a PDF from content stored in CRMs and ERPs.

The process is similar for generating a text file that can be used to generate a website such as a WPML file for WordPress import.

3. The Solution

- Software
  - Data Sources
  - Informer
  - A LaTeX Document Preparation System
    - Overleaf
      https://www.overleaf.com
    - An open source locally installed system:
      https://www.latex-project.org/get/
  - A Content Management System
    - WordPress
      https://wordpress.org

3. The Solution

- Data Sources
  - 25 Live Web Query
    - Use the API to get the academic calendar
  - Colleague Datasets
    - COURSES
    - COURSE.BLOCKS
  - WordPress Dataset
    - WP_Posts

3. The Solution

- 25 Live Web Query
  - Method & URL
    - GET https://25livepub.collegenet.com/calendars/YOUR_EDU-announcements.json
  - Response Parser
    - JSON Object Array

3. The Solution

- COURSES Dataset
  - *No* **Criteria** – select all so we can use the same Dataset year over year. The dataset will be refreshed periodically as new courses are added. Selections will be made with Template processors.
  - **Order** by CRS.NAME so output will already be sorted, i.e. AAA-0001 through ZZZ-9999
  - **Fields** – These are needed at a minimum, others may be needed
    - CRS.NAME
    - CRS.DESC
    - CRS.MIN.CRED
    - CRS.TITLE

3. The Solution

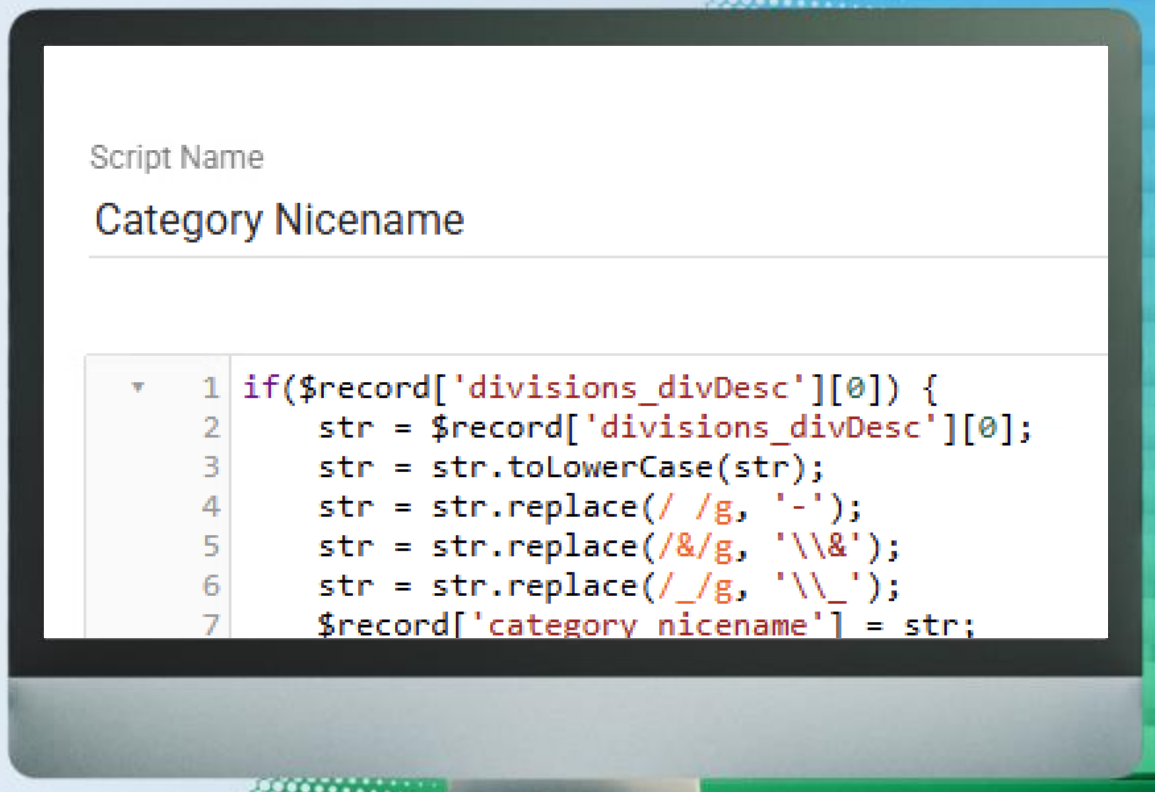
- COURSES Dataset (continued)
  - **PowerScript**
    - A series of Regular Expressions to clean data coming from Colleague.
      - Colleague injects an escape character to indicate a line feed. Informer converts that character to a comma. The result,can be,paragraphs that look,like this. Use Regular Expressions to remove unwanted commas and retain correct punctuation.
        - First replace all the commas with spaces:
          `String($record['crsDesc'].replace(/,/g, ' ')`
        - Then replace all double spaces with a comma and a space:
          `String($record['crsDesc'].replace(/  /g, ', ')`
        - Then replace all sequences of ., with .
          `String($record['crsDesc'].replace(/\., /g, '. ')`
    - Escape reserved TeX characters

```
String($record['crsDesc'].replace(/&/g, '\\&')
String($record['crsDesc'].replace(/\$/g, '\\$')
String($record['crsDesc'].replace(/_/g, '\\_')
```

    - Other possible data cleansing tasks
      - Convert HTML, Markdown, or other scripting to TeX
    - Similar cleansing needs to be done for WPML output

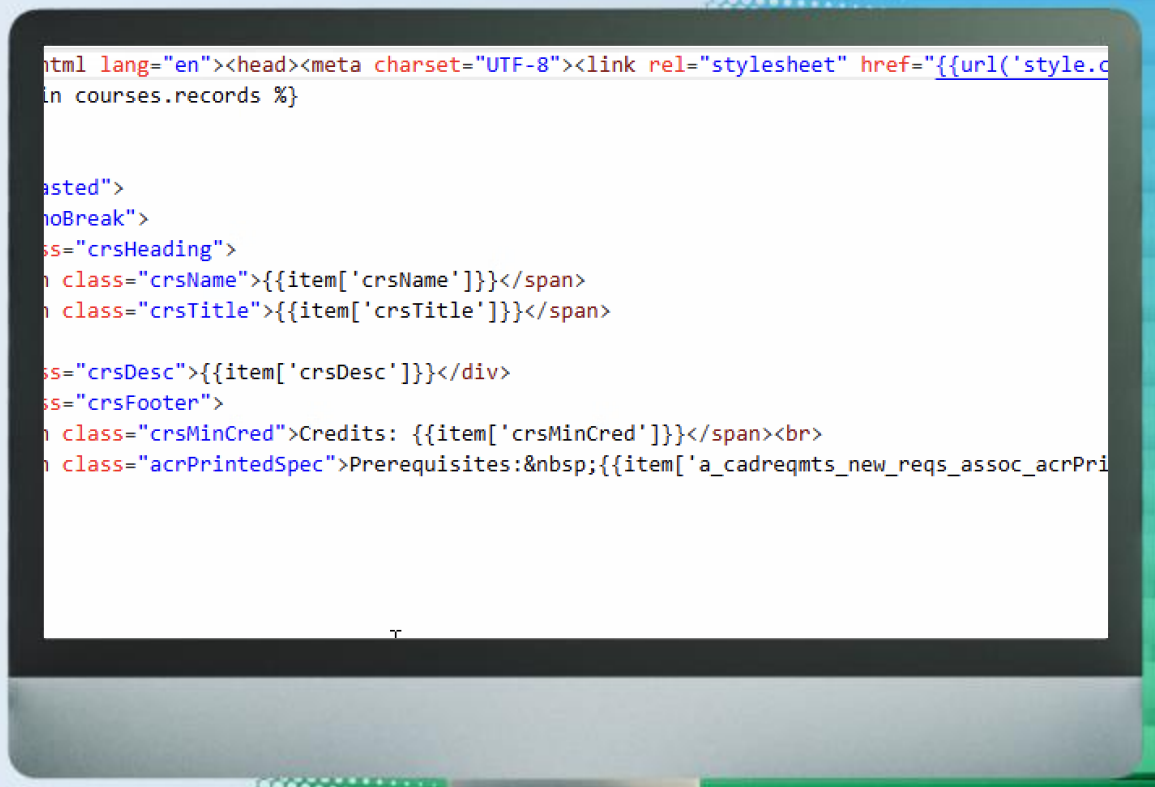
3. The Solution

- COURSES Dataset (Continued)
  - Text will need to be cleansed to escape reserved characters
    - Categories
    - Course Titles
    - Course Descriptions



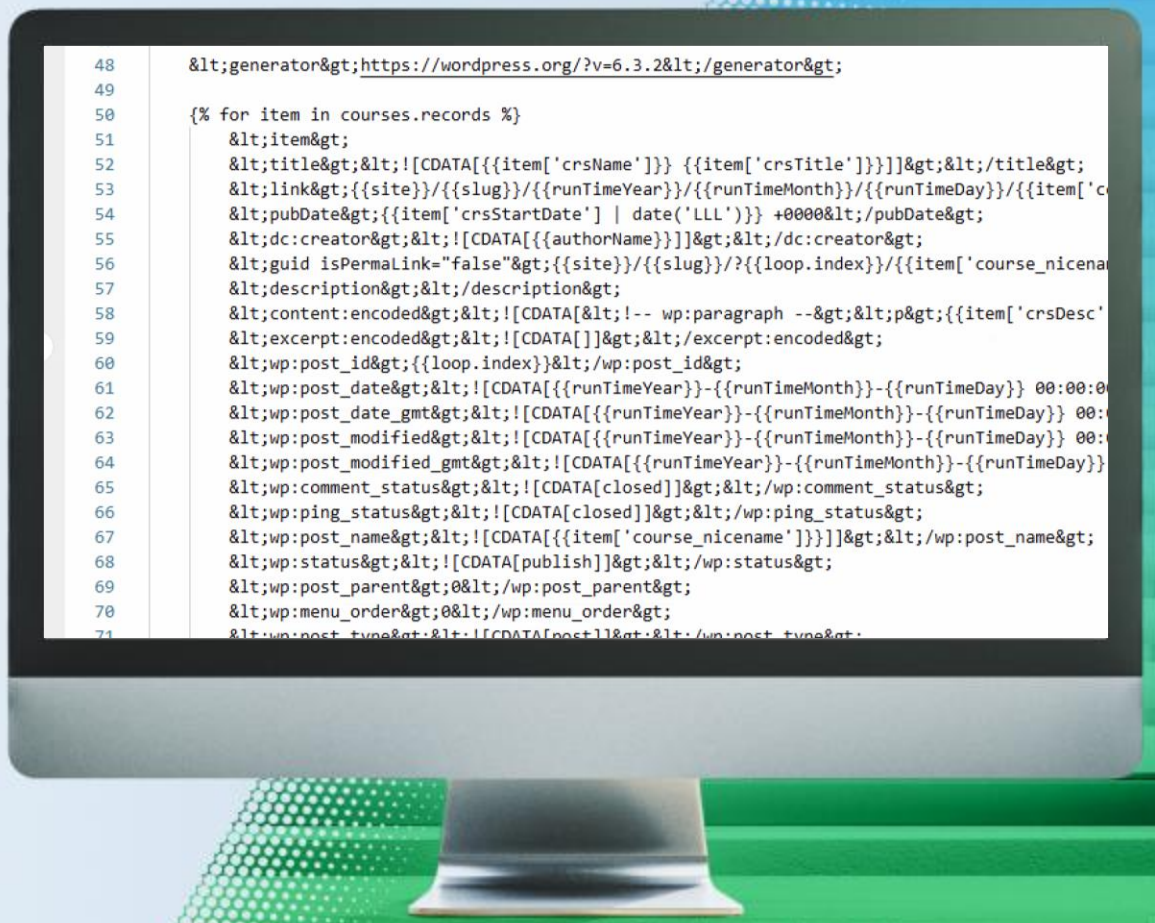
3. The Solution

- COURSES Dataset (Continued)
  - Course Listing Template
    - Loop through course records
    - Generate output (Tex, WPML)
      - Templates use HTML, so use HTML to format the TeX or WPML file so the output so you can use select all, cut, and paste into a text editor to save the file.



3. The Solution

- COURSES Dataset (Continued)
  - It would be nice if Informer Templates had a Text output to greatly simplify this part of the process, because writing XML (in this case WPML) is painful when everything needs to be escaped!



3. The Solution

- COURSE.BLOCKS Dataset
  - Criteria: CBL.CURRICULUM.TRACKS is not empty
  - Order the following Ascending By
    - CBL.CURRICULUM.TRACKS
    - COURSE.BLOCKS.ID
    - COURSES CRS.NAME
  - Normalize: Split on Curriculum Track

3. The Solution

- COURSE.BLOCKS Dataset (Continued)
  - Semester Order PowerScript
    - Make academic level and semester sortable by numeric

```
$record['semesterOrder'] = getOrderNumber($record['courseBlocksId']);

function getOrderNumber(code) {
    const semMatch = code.match(/SEM(\d+)/i);
    if (semMatch) {
        return parseInt(semMatch[1], 10);  // check for "SEMESTER #"
    }
    const levelMap = { FR: 0, SO: 2, JU: 4, SN: 6 };
    const acadMatch = code.match/(FR|SO|JU|SN)([12]/i);
    if (acadMatch) {
        const level = acadMatch[1].toUpperCase();  // Make Uppercase
        const semester = parseInt(acadMatch[2], 10);
        if (level in levelMap) {
            return levelMap[level] + semester;
        }
    }
    return 999;  // default if no match found
}
```

3. The Solution

- COURSE.BLOCKS Dataset (continued)
  - Tex Course Block
    - This will be unique to each project. This block loops through each course block starting with the first semester though the last semester of a selected curriculum track. It then writes a curriculum chart in TeX.

3. The Solution

- CMS Content Entries
  - WP_POSTS Dataset
    - Criteria: Type matches Page or Post; Post Status is Published.
    - Sort by Menu Order
    - Fields
      - ID
      - Post Title
      - Menu Order
      - Post Name
      - Post Parent
      - Post Content

3. The Solution

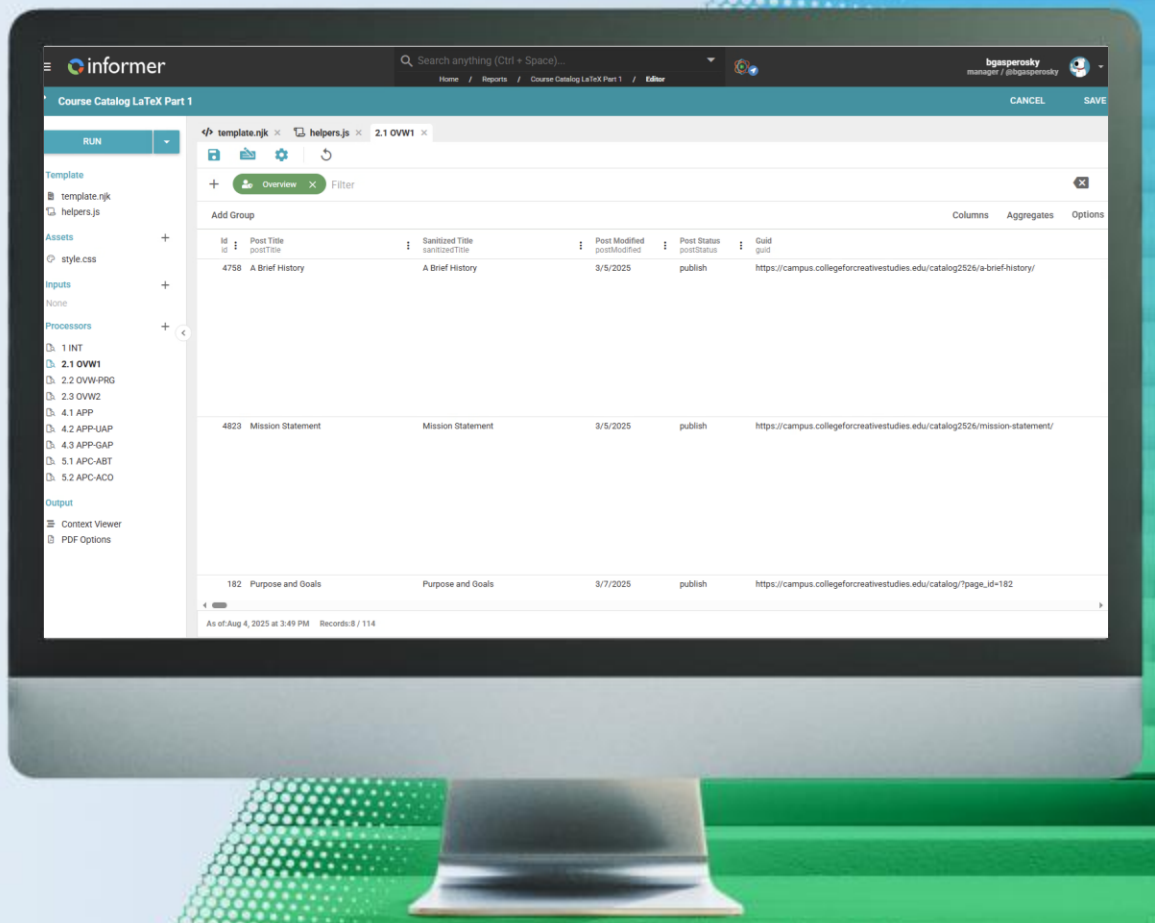
- CMS Content Entries (continued)
  - PowerScript
    - Convert HTML to TeX
    - Escape reserved TeX characters
    - Write entries in Tex in an arbitrary order (chapters, sections, etc.)

3. The Solution

- Course Catalog TeX Generator Template
  - Template.njk
    - A combination of HTML and TeX. The HTML is used to format the output at runtime so that the TeX file can be copied and pasted as a text file. If Informer adds a TXT output option for templates, the HTML will not be necessary.
    - Each article is laid out in order in this file
      - ```
        {% for item in Overview.records %}
            \chapter{{'{'}}{{item['sanitizedTitle']}}{{'}'}}
            {{item['latexPost']}}
        {% endfor %}
        ```
  - helpers.js
    - Reusable snippets of TeX stored in JavaScript variables.
      - `disclaimer = `\\textit{a reusable disclaimer is stored here}`;`
  - Processors
    - Each processor is a record or a set of records from one of the datasets filtered to output specific items like:
      - An article or a section of articles
      - A set of course blocks for a curriculum track
      - A list of courses for a program
      - A code snippet (LaTeX or WPML/HTML)
        ```

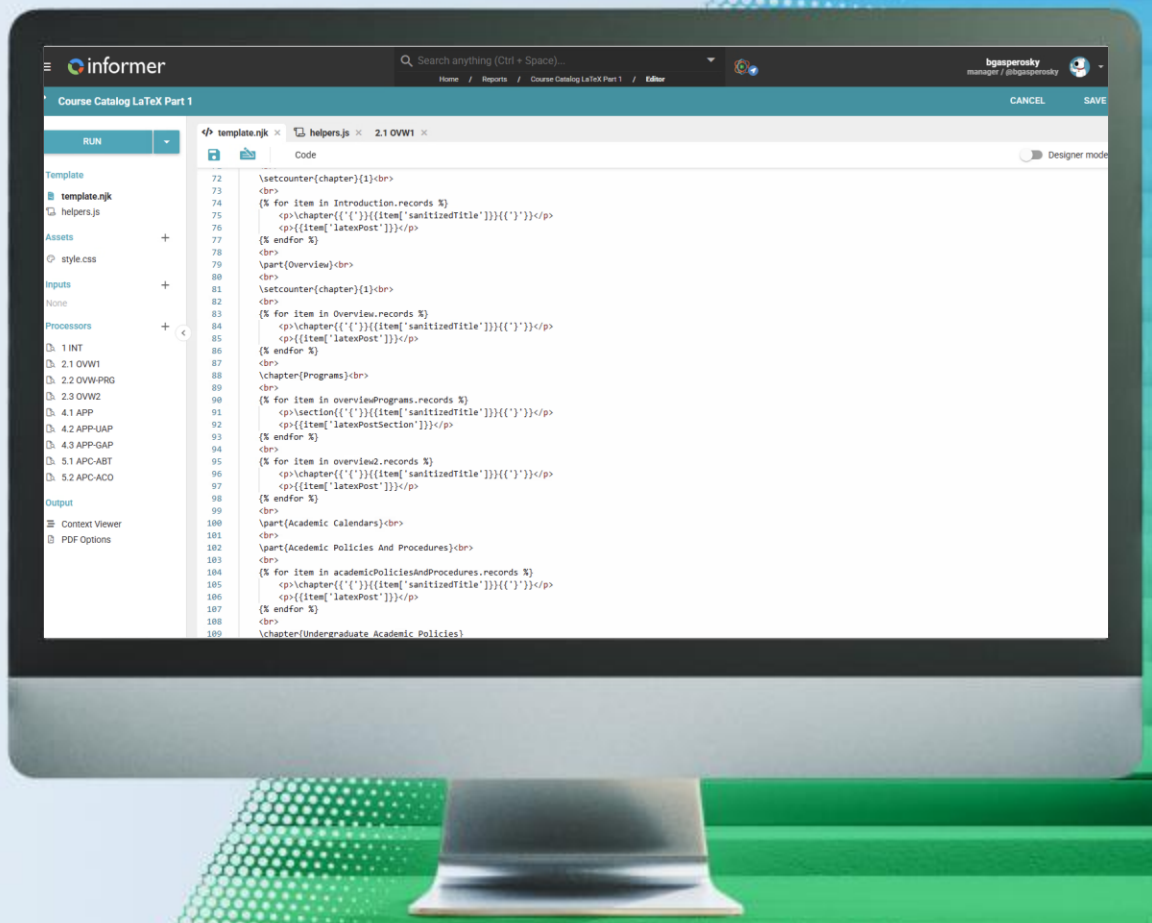
3. The Solution

- Course Catalog TeX Generator Template (continued)
  - Each section has its own processor
  - There are many processors using the same three datasets
    - Each processor has its own filter
  - This makes it easy to organize the content in the template



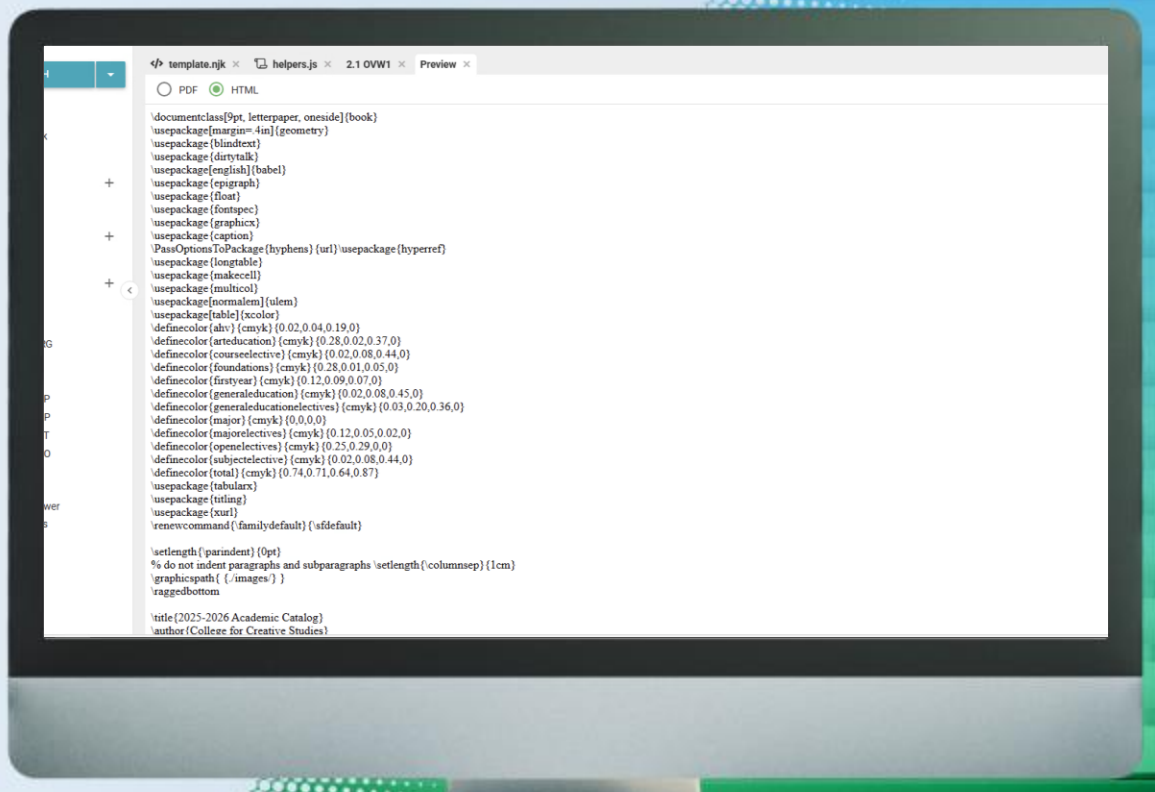
3. The Solution

- Course Catalog TeX Generator Template (continued)
  - The template code is TeX wrapped in HTML
  - Each loop represents a section (one or many articles)
    - Each processor selects a single section
  - Each loop is entered in any arbitrary order
  - LaTeX has a Table Of Contents feature that automatically generates the Table Of Contents.

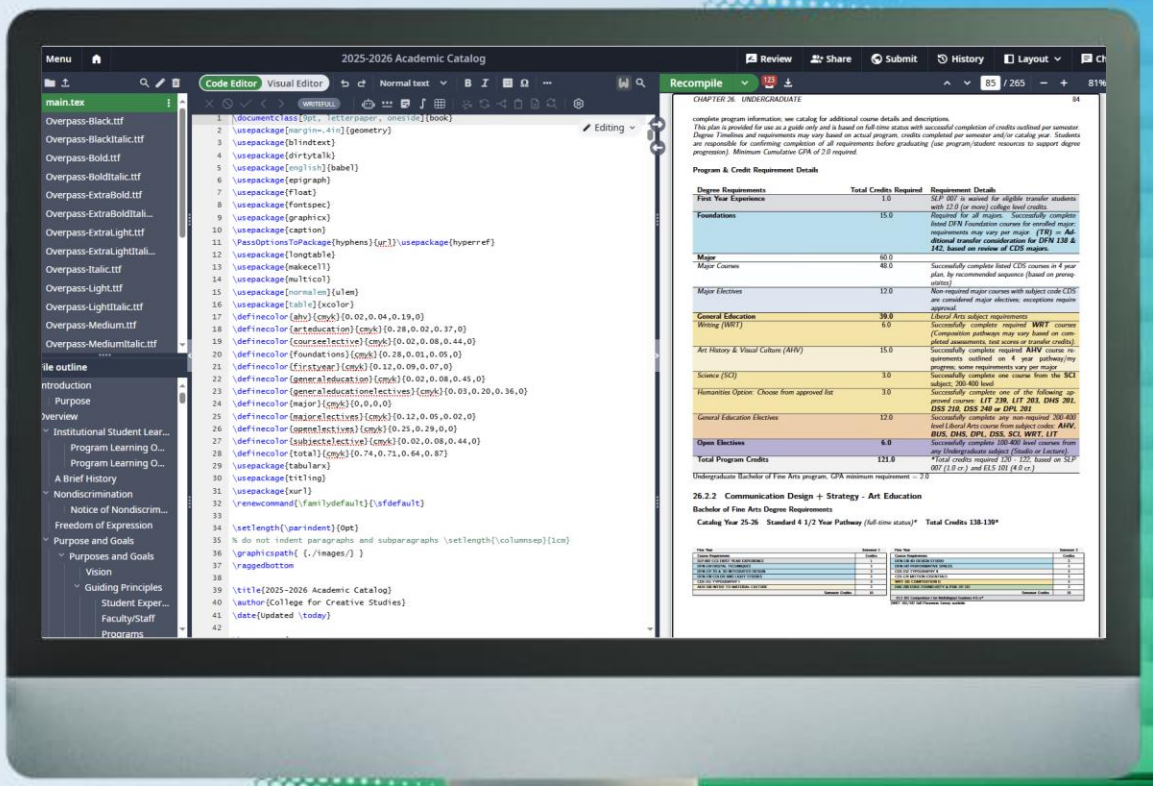


3. The Solution

- The TeX File
  - Run the completed template in HTML mode to generated the TeX file.
  - Select All of the TeX file and Copy it.



- The TeX File (continued)
  - In your LaTeX environment (in this case, Overleaf), do the following:
    1. Paste the TeX file generated in Informer into the editor.
    2. Upload any images referenced in the TeX file.
    3. Upload any fonts used by the TeX file.
    4. Click the Recompile button.
    5. Download the PDF.



3. The Solution

- The completed process for exporting to WordPress
    1. Run the completed WPML template as HTML in Informer.
    2. Copy the file output.
    3. Save the file as a .WXR file.
    4. Spin up a new WordPress site.
    5. Import the .WXR file
        1. This will create the pages and posts from the entries you selected from your datasets and Informer template.

4. A Quick Demo

entrinsicon

Generating a course catalog

5. Tips, Tricks, and Gotchas

- “baby step onto the elevator... baby step into the elevator... I'm in the elevator.” – Bob Wiley, ‘What About Bob?’ (1991 movie)
  - Current Course Catalog project stats
    - 3 datasets
    - 5 templates
      - I kept pushing the limits of Informer so I broke the TeX catalog template into three parts.
      - 9 Powerscipts
        - Over 1,000 lines of JavaScript
      - 5 Nunjuct Template Files
        - Over 50 Processors
        - Over 1,600 lines of Nunjucks code

5. Tips, Tricks, and Gotchas

entrinsicon

- ”AI is about as good as a B- Grad Student” - anonymous
  - Ask AI to write things like regular expressions. For example, Google gave me this in a few seconds:

```
const strongPasswordRegex = /^(?=.*[a-z])(?=.*[A-
Z])(?=.*\d)(?=.*[!@#$%^&*()_+{}\[\]:;<>,.?~\\/-]).{8,16}$/;

// Explanation of the regex:
// ^                                  - Start of the string.
// (?=.*[a-z])                        - Positive lookahead: asserts that there is at least one
lowercase letter.
// (?=.*[A-Z])                        - Positive lookahead: asserts that there is at least one
uppercase letter.
// (?=.*\d)                           - Positive lookahead: asserts that there is at least one
digit.
// (?=.*[!@#$%^&*()_+{}\[\]:;<>,.?~\\/-]) - Positive lookahead: asserts that there is at
least one special character.
// .                                  - Matches any character (except newline).
// {8,16}                             - Quantifier: ensures the string length is between 8 and
16 characters.
// $                                  - End of the string.
```

  - If the first answer doesn’t work right, have a conversation with an AI Chat Bot to refine the code until it works.
  - The 80/20 rule applies. The AI can get you close, but you’ll need to fine tune the results.
  - Be skeptical of the AI’s answers, it can hallucinate and produce wacky code.

5. Tips, Tricks, and Gotchas

- "It is impossible to make anything foolproof because fools are so ingenious” - anonymous
  - Humans are great at:
    - Making mistakes
    - Being inconsistent
    - Typos
    - Not following directions
    - Not communicating when they should
  - Much of my time on this project has been spent writing exception code to deal with data entry errors.
- Do not attempt a large project like this with a deadline!