PrairieLearn Element Extension

Senior Design May 2024 - 33

Mitch Hudson
Tyler Weberski
Chris Costa
Andrew Winters
Carter Murawski
Matt Graham

Table of Contents

Introduction

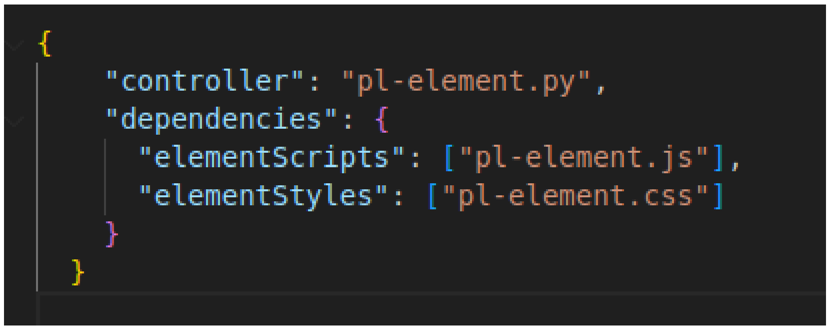
This document outlines how to create element extensions in the PrairieLearn environment to create additional functionality for specific PrairieLearn elements. This document will include the steps to set up and create an element extension with an example from current experience within the project. Finally, it will show where some potential issues may arise during implementation.

Element Information

Element Anatomy

Elements for the current PrairieLearn server are in the Elements folder in the PrairieLearn directory. Each element resides in a folder corresponding to the element's name. Course-specific elements are allowed and must be located in a directory at the root of the edited course repository. All element files are named the same as the element it belongs to. Within the element directory, there will be an info.json file that contains metadata regarding the given element. The info.json file contains the controller and any other dependencies for the element. Each element has a .py controller file that will house the functions that allow the element to work. The controller will render the element, parse submissions, and manipulate the submission as needed. Below is the file structure and info.json file for an example element:

```
Pl-element
+-- info.json
|-- pl-element.py
|--pl-element.mustache
|--pl-element.js
|--pl-element.css
```



```
{
    "controller": "pl-element.py",
    "dependencies": {
      "elementScripts": ["pl-element.js"],
      "elementStyles": ["pl-element.css"]
    }
  }
```

Element Dependencies

Each element depends on specific client-side assets like scripts and stylesheets. These dependencies can be placed into other files to separate HTML, CSS, and JS. A list of dependencies required by all elements on a page will be rendered by PrairieLearn, and ensure the elements are rendered on the page. The Element anatomy shows that dependencies are listed in the elements info.json file. A list of dependency types can be found on the elements page linked in references.

Creating Element Extensions

Extension File Tree

Element Extensions are set up and maintained similarly to the anatomy of an element. Extensions are defined inside a course rather than the PrairieLearn server, where elements are defined. Extensions are located in a folder within the course directory where elements are contained in subfolders, each of which holds the extensions for the respective element. An example folder structure for an example extension can be seen below:

```
–Course Directory
  |-->/ElementExtensions
     |-->/ExampleElement
        |-->/ExampleExtension
            |->ExtensionFiles
```

Inside the subfolder “ExampleExtension,” there needs to be an info.json file that follows the same format as the element info.json mentioned in the previous section. In the info.json file, the Python controller and all dependencies must be defined in the respective fields.

Python Controller

Each extension requires a Python controller script that will house the functionality of the extension that is being created. The controller does not follow a standard structure but is defined by the element that is being extended, where the original elements' global functions and variables can be accessed and manipulated. To gather the original elements' global information, the load_extension or load_all_extensions function must be called. It is important to know that there is a two-way flow of logic and information between the elements and the newly created extensions.

Several functions can be extended and edited from the base element inside the Python controller. These functions will override the original element functionality, replacing the old element. The functionality can then be tested once the Python file is registered with the base element, and any and all issues will be traceable, similar to regular error traces in the PrairieLearn environment.

Import Extension From an Element

To work on an element extension, PrairieLearn must be imported to call the load functions to gather all the global functions and variables. The following statement must be added to gather element functionality inside the Python controller: “import prairielearn as pl”. Once this statement is added, the pl.load_extension function can be called where the parameters are “data” and the respective extension name. These steps will help render the newly created extension inside of PrairieLearn.

Extra Files

Other client-side files, such as images or downloadable files, may need to be loaded for an extension, depending on the required functionality. Extra client-side files must be placed in a “clientFilesExtension” directory, and insert the full URL for that folder into the extension as a dependency.

Javascript

Depending on the type of element being extended, a Javascript file may need to be added to the client side of the extension. Like the Python controller, each element extension will inherit a base javascript class. The javascript file will primarily add cosmetic differences for the element being extended to help add extra functionality or a different format of the respective element.

Connecting Extension Files

To have the element extension functional, all files need to be registered and picked up by the element that is being extended.

Python

Each new class defined by the Python extension scripts must be placed into a global dictionary that maps the element's name to the new class. Once the scripts are placed into the global dictionary, the base element can pick up the extension for use in PraireLearn. Examples of registering the extensions into the dictionary are in the links provided below.

JavaScript

All elements that were extended must be registered with the pl-element before they are able to be used. This process is similar to the Python class registering, with some differences. Registering the new element can be done within the given Javascript file, which helps track and clean up the file structure. The extension name must match the folder in the “element extension” directory to register the JavaScript files.

Extension Issues

Many issues can arise when creating, testing, and using an element extension. One major issue not addressed by documentation is where extensions are registered on the server side. Testing was performed by trying different locations to find the correct placement defining the newly created server-side files. The extension will continue to throw errors if the files are not correctly implemented or registered. Further information is needed and required to develop and adapt an extension successfully. With more time working on the project, one can successfully add an extension with functionality to the PrairieLearn environment. Examples of extensions can be found within the current build of the PrairieLearn server inside an extensions file. While these examples are helpful, they do not address every question and piece of information needed to implement the extension.

Resources

PrairieLearn Resources

Question Element information:
https://prairielearn.readthedocs.io/en/latest/devElements/#element-dependencies
Element Extension Informtion: https://prairielearn.readthedocs.io/en/latest/elementExtensions/
Additional Helpful Element Information(end of page):
https://prairielearn.readthedocs.io/en/latest/elementExtensions/