OregonFlora - symbosu codebase
June 2025

The OregonFlora codebase is sizable and can be overwhelming to digest at first. Fortunately, large swaths of it are hardly ever used and will not be relevant in much of your day-to-day work. In this document, I will give an overview of the most important components, how they function, and some notable things to be aware of.

Many of the root-level directories contain dead or rarely-used code. These are the important ones to be familiar with:

- **classes**: contains backend logic that supports most of the pages on the site. These tend to contain logic for selecting or updating data in the MySQL database, and processing it for display or output. More detail below.
  - Even within here, there is a lot of dead code. Classes you will likely work with include **TaxaManager**, **IdentManager**, **ExploreManager**, **OccurrenceManager** / **OccurrenceListManager** / **OccurrenceMapManager**, and possibly others.
- **config**: contains sitewide configuration options, feature flags, and database connection information. **symbini.php** is an important file here and is gitignored as it contains instance-specific settings.
- **css**: contains CSS files used across the site. Again, lots of dead code here. Important files include **css/symbiota/main.css** and **css/symbiota/customizations.css**, as well as the whole **css/compiled** directory (which holds the transpiled CSS styles for React pages — more on that later).
- **garden, ident, rare, taxa, projects checklists**: contain pages and API endpoints for some of OF's main tools
- **collections**: contains code and pages for herbarium-collection-related tools, including: herbarium search, mapping, and collection administration
- **js**: a monolith of its own, comprising of current & old JS dependencies used across the site, along with a smattering of extremely important files (particularly in the **symb** and **leaflet.OregonFlora** directories) and the entire React frontend of the site, buried in the **react** directory.
- **pages**: OF-specific static pages that render plain HTML and are not React-based (more on this later)
- **header.php** and **footer.php**: OF's React-based header and footer components, inserted on almost every page. These are distinct from Symbiota's **includes/head.php** and **includes/footer.php**, which mainly contain dependency links.

Below I'll discuss the classes and the frontend of the website in more detail. The frontend can be roughly divided into three different categories of pages: (1) pages that stem from Symbiota & their supporting files, (2) OregonFlora's interactive React-based pages, and (3) OregonFlora's other static pages.

Classes

These live inside the classes/ directory and are the heart of most of the backend functionality. Many of them are Symbiota-defined and may have some amount of OF customizations. Others are entirely created by OregonFlora.

- OregonFlora-created classes include TaxaManager (used for React-based taxon profile pages) and IdentManager (use for Identify, Grow Natives, and Rare Plant Guide tools)
- These classes tend to use Doctrine ORM as an additional abstraction layer around the MySQL database. The Doctrine query language is very similar to MySQL and is fairly straightforward to understand/write by following examples in the code. Note our Doctrine version (2.7) is very old so documentation online may not always be reliable.
- Most Symbiota-created classes just communicate with the database directly, using PHP's native mysqli interface.
- Be careful when instantiating Symbiota-created classes, as some of them make database calls or have other side effects in the constructor(!) and as a result can be very slow to instantiate.

React-based pages (e.g. Identify, Explore, Grow Natives, Taxon Profile Pages)

The frontend React code for these pages lives in js/react. Built output (from npm run build or npm run devstart) goes into js/react/dist and css/compiled. For the most part, each page, along with the common header and footer, is its own separate React module & entrypoint; you can see this in js/react/webpack.config.js where all the entrypoints are defined. These individual modules are combined together and rendered into full pages by PHP files — e.g. garden/index.php.

Most React pages rely on API endpoints in <root-level folder>/rpc/api.php. These files are all OregonFlora-created, but may rely on Symbiota classes to interface with the database. We have generally tried to keep database calls out of the API files themselves and use classes as an abstraction layer.

Some specific notes:
- The header and footer (seen across the site) are their own individual React modules.
- Many CSS rules (js/react/src/less) are scoped to specific pages, by root div id. If you are trying to making a change and it doesn't seem to stick, make sure you aren't inside the scope of a specific page (e.g. #react-taxa-garden-app). Note also that the css files must be compiled by npm run build or npm run devstart to have an effect.
- Hot module reloading does not work at this time. If you have npm run devstart running, the js and css modules will be watched & rebuilt when you make changes, but you'll need to reload pages in your browser to see the changes.
- As a result of OF having had multiple different developers on this project over time, you will notice differing levels of React idiomaticity/idiosyncrasy in the various modules. Some are written in a more modern functional style with hooks, and some are still in an older class-based style. Data processing and state control are also all over the place, and some pages are massive monoliths. I recommend trying to stick to best React practices in new pages moving forward, and working with existing modules as best you can.
- The garden (Grow Natives) and rare pages are sort of a special case of Identify, with a predefined (rather than dynamic) set of characteristics available to filter. They have their own PHP index files and API endpoints (though those are largely the same) and share a single React root component (checklist-special) that switches between the two dynamically depending on the root div id. This React component is fully separate from the Identify tool, descended from it, but with some differing functionality and moved to a somewhat more modern React style.
- Taxon profile pages (core, garden, and rare) are different enough that they each have their own PHP index file and React module, but they share a single API endpoint (with type=rare query param used to indicate a slightly different set of data being requested).

Symbiota pages

The majority of the pages on the site stem directly from Symbiota. These pages are generally written directly in PHP and tend to use jQuery or vanilla JavaScript for any interactivity. Most also have some amount of customization to fit into the OF site.

This includes all the functional pages for adding/editing data, user accounts/login, herbarium search & mapping tools, and virtually everything accessed through the Site Map.

Some important notes about OF's customizations:

- Majorly, we have inserted our header and footer on most pages. Since these are React apps and use the React CSS styles, this has resulted in a complicated CSS dependency chain for a lot of pages (further complicated by a dependency on bootstrap). Unfortunately, there are a lot of global styles defined in (a) the React CSS, (b) the Symbiota CSS, and (c) bootstrap, and not much scoping. So this means it can be very difficult and brittle to change global styles, because many pages depend on existing styles in unexpected and convoluted ways.
  - Being able to remove or scope the bootstrap CSS dependency to specific pages or elements would go a long way towards resolving this, but is a big job.
- Various tools have the minor customization added of being able to restrict their functionality (e.g. search) to taxa on the Oregon Vascular Plant Checklist.
- Certain pages (e.g. Symbiota's taxon profile pages) have been entirely replaced by our React versions. In these cases, .bak.php files contain the Symbiota versions, but there is no real need for them as far as I can tell.
- The mapping tool has some major customizations, largely around map layers and styles, clustering of data points, and being able to import or drag-and-drop KML files to define search polygons. Much of the customized functionality lives in separate files, such as js/leaflet.OregonFlora/leaflet.OregonFlora.js and js/symb/collections.map.index.OregonFlora.js .
- The mapping tool also has a major backend customization: instead of selecting data from MySQL, we use Apache SOLR. SOLR is faster for broad searches of large datasets, and crucially, is significantly faster for geospatial searches (e.g. searches within a complicated polygonal boundary).
  - All the changes to use SOLR are gated behind the $MAP_SOLR_SEARCH_FLAG, which may be helpful to search in order to better understand how this customization works. The intention was to build, as much as possible, a drop-in replacement on the JS side.
  - SOLR is running as a separate service on the live and dev servers. A nightly cronjob updates its data using live data from MySQL.
  - SOLRConnector.php is the interface used for connecting PHP with SOLR. It mainly just executes local curl commands and lightly processes the output. SOLRManager.php is a legacy class that I don't believe is used anywhere relevant.

Other static pages

There are a few other pages, mostly with static content, that are linked from the main header menus. These live in the pages/ dir and are simple PHP files that mostly render static HTML content. They are noteworthy for using yet a different set of CSS dependencies; some of them depend on older versions of Symbiota base CSS styles (css/base.css and css/main.css) and we did not remove this dependency when we went through the rest of the Symbiota and React pages in Jan 2025.