LIFERAY®

CONTROLLING PAGE STRUCTURE

THE OUTLINE OF A THEME

- One of the first areas we'll deal with is the page structure.
- Just about every page in Liferay is generated from a template in the theme.
- If we want to control the HTML for:
    - The head
    - The body
    - The navigation
    - The footer
- Then we'll want to modify the templates in our theme.

WHAT WE NEED

- The S.P.A.C.E. design team has already outlined some changes we'll need to implement to apply our custom branding to Liferay.
- In addition to the main styling changes, we want to:
  - Customize the nav bar on the screen
  - Add a custom footer to the page
- As we get in there, we may find additional areas we want to customize.

FREEMARKER TEMPLATING LANGUAGE

- Chances are, you've worked with some templating system before.
- Templates have their own syntax, and generate HTML pages when processed.
- Your theme will use FreeMarker as the main templating language.
- As we go, keep an eye out for these features:
    - **Interpolations:** these are basic variable expressions that will be evaluated, and the result put on the page:

      ```
      ${default_url}
      ```
    - **Directives:** special tags that tell FreeMarker to do something that doesn't always result in a visual difference:

      ```
      <#assign default_url = "http://spaceprogram.liferay.com/" />
      ```
- There are some other neat features, but we'll just focus on the basics for now.

WHAT MAKES UP A THEME?

- Let's take a look at what's in the theme and find our place:
  - **fonts** will contain any necessary custom fonts for the theme.
  - **images** will contain images that will be used in the theme.
  - **js** will contain JavaScript referenced in the themes templates.
  - **layouttpl** will contain any custom Liferay Layout Templates for the theme.
  - **templates** will contain the FreeMarker templates that provide the HTML structure of the theme.
- Let's walk through modifying some of the main files.
- We'll start by bringing in our theme files and modifying each section.

EXERCISE: SETTING UP BRACKETS SNIPPETS

- We've provided our theme's src files and snippets to walk through development.
- Let's start by setting up Brackets.

1. **Open** Brackets.
2. **Click** on the *Getting Started* dropdown in the top left menu.
3. **Click** *Open Folder...*
4. **Go to** *exercises/front-end-developer-exercises/03-theme-development/01-generating-a-theme/*.
5. **Choose** the snippets folder.

✓ Now our snippets are ready! Let's get started.

TEMPLATES AT OUR DISPOSAL

- When an HTML page in Liferay is generated and sent to the browser, a few templates are used to create the source HTML:
  - `portal_normal.ftl`: This is the main HTML document for every page, containing the `<html>`, `<head>`, and `<body>` tags.
  - `portlet.ftl`: This contains the HTML that surrounds each application on the page.
    - This will be used when displaying applications with borders on the page, and will be inside the Layout Template. There is more on this in the Appendix.
- The main HTML document for all pages in Liferay can be pretty complex.
- We can include templates in other templates to modularize our page structure.
- For instance, we can separate our page navigation, body, and footer into different files.
- Let’s set up our basic template files in the theme.

EXERCISE: MODIFYING OUR TEMPLATES

1. **Copy** the contents of the *exercises/front-end-developer-exercises/03-theme-development/01-generating-a-theme/exercise-src/templates* folder.
2. **Paste** the contents into the Space Program Theme's *src/templates* folder.

- We've added a few template files for our theme:
  - **footer.ftl**: A custom ftl file we can use to separate the footer code from the main src file
  - **init_custom.ftl**: Includes our custom variables for use in the theme
  - **navigation.ftl**: Includes the HTML structure of the navigation in our theme
  - **portal_normal.ftl**: As mentioned before, this is our main source file where everything comes together.
  - **portlet.ftl**: This includes the HTML structure of application containers.

PAGE COMPOSITION

- We'll build the main page using two additional templates.
- This will make the structure easier to maintain and modify.
- Our main page will be laid out with `portal_normal.ftl`:
  - *HTML Head*
  - *JavaScript*
  - *Stylesheets*
  - *Body*
    - *Header*
    - *Navigation* - we'll include `navigation.ftl`
    - *Page layout* - controlled by a Layout Template
    - *Footer* - we'll include `footer.ftl`

EXERCISE: MODIFYING THE HTML SOURCE

- Let's walk through modifying our `portal_normal.ftl`.
- We'll add the header, body, and footer sections inside our wrapper `<div>`.

1. **Drop** the `portal_normal.ftl` file from your theme's `src/templates` folder into the Brackets editor.
2. **Open** the *templates* section under *snippets*.
3. **Click** on the `01-portal-normal-header` snippet.
4. **Copy** the contents of the snippet.
5. **Paste** the snippet contents over the `<#-- Insert snippet 01-portal-normal-header here -->` comment in the `portal_normal.ftl` file.

THE HEADER

- If you’ve worked with Bootstrap, the class names will look familiar.
- Our header is a `navbar`, with some other UI components included.
- In the Header section of the `portal_normal.ftl`, we’ve added a fluid class for responsive design.
- This section includes our navigation, site name, and logo.
- Our full navigation structure is in the `navigation.ftl` and is being included separately.

MAKING A STATEMENT

- You'll see all of our basic FreeMarker syntax in this template.
- Text and URLs that Liferay gives us are in variables:

  ```
  ${site_name}
  ```

- We check to see if navigation is enabled using the `<#if />` directive:

  ```
  <#if has_navigation>
  ...
  </#if>
  ```

- We also check if we're supposed to show the Site name:

  ```
  <#if show_site_name>
  ...
  </#if>
  ```

- We'll see this kind of simple logic in most templates.

INCLUDING EXTERNAL TEMPLATES

- A useful directive for us is `<#include />`.
- This takes a template file name, and includes it in your template.
- It's the same as copying and pasting the contents:
  - FreeMarker looks in the same location as the current template.
  - If it finds the file, it adds the content of the template where the `<#include />` directive is.
  - The whole template file is processed, with this new template code in place.
- This makes separating our navigation into another file easy:

```
<#include "${full_templates_path}/navigation.ftl" />
```

- This also uses the `full_templates_path` variable Liferay gives to us.

USEFUL VARIABLES

- We're already seeing some helpful tools available in our theme templates.
- Some of the useful default variables include:
  - `site_name`: Returns the name of the Site, usually set by the Site Administrator
  - `site_default_url`: Returns the URL to the current Site
  - `site_logo`: Returns the URL to the Site logo image, usually set by the Site Administrator
  - `the_title`: The title of the application being displayed
  - `full_templates_path`: An absolute path on the file system to the folder containing the templates in the theme, useful for including external files.
- We'll see some more as we go.

EXERCISE: ADDING THE BODY SECTION

1. **Click** on the `02-portal-normal-main` snippet.
2. **Copy** the contents of the snippet.
3. **Paste** the snippet contents over the `<#-- Insert snippet 02-portal-normal-main here -->` comment.

- This section includes some accessibility classes.
- This part of the template also includes an advanced feature: user-defined tags, or *macros*.
- In the section, we have removed the breadcrumbs macro that's included as part of the navigation in the default `portal_normal.ftl`.

ADDING TEMPLATE FUNCTIONALITY WITH MACROS

- *Macros* allow you to assign template fragments to a variable, which makes them helpful for creating reusable pieces of template code.
- They will look very similar to normal tags:

  ```
  <@liferay.language_format />
  ```

  - Instead of starting with a #, they begin with a @.
- Like directives, they're used to perform different functions.
- When the page is rendered, the macro is replaced with the template fragment.

DEFAULT LIFERAY MACROS

- Liferay provides a few default macros for integrating the most commonly used applications in your theme (e.g., search, breadcrumbs, user personal bar).
- There is also a set of default macros for users seeking to make their site more friendly to foreign languages.
- You'll find default Liferay macros used all around our templates.

LOOKING AT LIFERAY MACROS

- Let's take a look at some of the default macros:

Default Liferay Macros		
Macro	Parameters	Description
breadcrumbs	default preferences	Adds the Breadcrumbs portlet with optional preferences
control_menu	N/A	Adds the Control Menu portlet
css	filename	Adds an external stylesheet with the specified file name location
date	format	Prints the date in the current locale with the given format
js	filename	Adds an external JavaScript file with the specified file name source
language	key	Prints the specified language key in the current locale

ADDITIONAL LIFERAY MACROS

Default Liferay Macros		
Macro	Parameters	Description
language_format	arguments key	Formats the given language key with the specified arguments
languages	default preferences	Adds the Languages portlet with optional preferences
navigation_menu	default preferences instance ID	Adds the Navigation Menu portlet with optional preferences and instance ID
search	default preferences	Adds the Search portlet with optional preferences
user_personal_bar	N/A	Adds the User Personal Bar portlet

EXERCISE: FINISHING WITH THE FOOTER

- Let's finish up with the footer section.

1. **Click** on the `03-portal-normal-footer` snippet.
2. **Copy** the contents of the snippet.
3. **Paste** the snippet contents over the `<#-- Insert snippet 03-portal-normal-footer here -->` comment right under the `</main>` tag.
4. **Save** the file.

- In this case, instead of adding our footer code in `portal_normal.ftl`, we've created `footer.ftl` file that we're adding.
  - You'll recognize the `<#include />` directive here.
- The benefit of this approach is keeping the `portal_normal.ftl` file uncluttered.

EXERCISE: ADDING THE FOOTER.FTL

- Now that our `portal_normal.ftl` is referencing the `footer.ftl` file, we need to add our footer code.

1. **Drop** the `footer.ftl` file from your theme's `src/templates` folder into the Brackets editor.
2. **Click** on the `04-footer` snippet.
3. **Copy** the contents of the snippet.
4. **Paste** the snippet contents over the `<#-- Insert snippet 04-footer here -->` comment in the `footer.ftl` file.
5. **Save** the file.

- Here we've added responsive fluid classes and a footer navigation menu.
- You'll notice this is similar to the header, but with different content.

FOOTER DETAILS

- A few new directives show up in the footer:
- `<#assign />` lets us assign a value to a variable.
- Here, we're using it to call methods on objects, and store the result:

```
<#assign VOID = freeMarkerPortletPreferences.setValue(...) />
```

- When the `setValue()` method returns, that value is stored in `VOID`.
- We don't use `VOID` anywhere on the page, so that value never gets displayed.
  - This is a nifty way to keep return values from being added to the page.
- `<@liferay.navigation_menu />` is a macro that adds the Navigation application to the page.

FILLING IN THE FRAMING

- With our basic templates in place, we’ve now provided the basic structure for styling.
- After modifying the core `portal_normal.ftl`:
  - We have a custom navigation that complements our branding
  - We have a custom footer section in place that we can easily modify later
  - We can add or remove some of the basic features using macros
- We can now start applying our custom styles to make the new page structure fit our vision.

Notes: