IMPORTING RESOURCES

IMPORTING PREDEFINED CONTENT

- The Resources Importer allows you to deploy your themes with predefined content.
- This can be useful for creating content and themes together to provide a wholesale style change.
- When deployed, the Resources Importer creates a site template, which can be used for creating new sites with a predefined look and feel.

RESOURCES IMPORTER'S FILE STRUCTURE

- Theme resources reside in the source of your theme.
- You can create files and folders in a new *[theme-folder]/docroot/WEB-INF/src/resources-importer* directory.
- Example file structure:

```
resources-importer/
    document_library/
        documents/
    journal/
        articles/
        structures/
        templates/
    assets.json
    sitemap.json
```

RESOURCES IMPORTER CREATES A SITE TEMPLATE

- The site template that will be generated is defined in the `sitemap.json` file.
- This file describes the contents and hierarchy of a site that Liferay can import as a site template.
- The `sitemap.json` file defines the following:
  - Pages of a to-be-generated site template
  - Layout templates
  - Applications
  - Application preferences (Portlet preferences)
  - Content to display

RESOURCES IMPORTER EXAMPLE

- Here is a `sitemap.json` example that will create one page named `Welcome` that has two columns, a Login application in one and a Hello World application in the other.

```
{
    "layoutTemplateId": "2_columns_ii",
    "publicPages": [
        {
            "columns": [
                [ { "portletId": "com_liferay_login_web_portlet_LoginPortlet"
                [ { "portletId": "com_liferay_hello_world_web_portlet_
                HelloWorldPortlet" } ]
            ],
            "friendlyURL": "/home",
            "name": "Welcome",
            "title": "Welcome"
        }
    ]
}
```

IMPORTING ASSETS WITH METADATA

- The `assets.json` file specifies details about the assets to be imported.
- Tags can be applied to any asset.
- Abstract summaries and small images can be applied to web content articles. As an example:

```
{
    "assets": [
        {
            "name": "company_logo.png",
            "tags": [ "logo", "company" ]
        },
        {
            "abstractSummary": "This is an abstract summary.",
            "name": "My Example.xml",
            "smallImage": "company_logo.png",
            "tags": [ "web content" ]
        }
    ]
}
```

IMPORTING DOCUMENTS AND MEDIA

- By default, all assets under the directory `document_library/documents/` get imported into the platform's global *Documents and Media*.
- Example file structure:

```
document_library/
    documents/
        image.png
        Custom Folder/
            image 2.png
```

- With this example file structure, `image.png` will be placed in the root folder of the *Documents and Media* application.
- `image 2.png` will be placed in a folder named `Custom Folder`.

IMPORTING WEB CONTENT

- The `journal` directory is used for importing various assets related to web content such as structures (JSON), templates (Velocity/FreeMarker), and web content articles (XML).

```
journal/
    articles/
        My Example Template A/ (matches Template name)
            My Example Article X.xml
            My Example Article Y.xml
    structures/
        My Example Structure H.json
    templates/
        My Example Structure H/ (matches Structure name)
            My Example Template A.flt
            My Example Template B.flt
```

EXAMPLE STRUCTURE

- Here is an example of a Structure. We'll call it `My Example Structure H.json`:

```
{
    "availableLanguageIds": [ "en_US" ],
    "defaultLanguageId": "en_US",
    "fields": [
        { "name": "Header", "type": "text", ... },
        { "name": "Body", "type": "textarea", ... }
    ]
}
```

- The source JSON of Web Content Structures can be found by clicking the `Source` tab while editing the Structure.

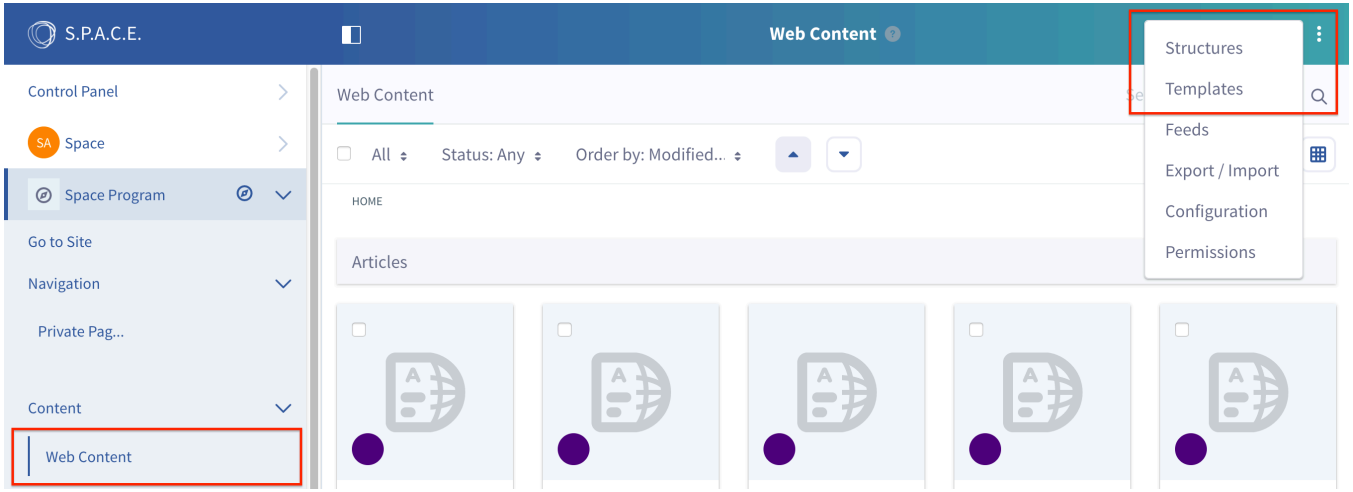
EXAMPLE TEMPLATE

- Here's an example of a Template. We'll call it `My Example Template A.ftl`:

```
<h1>${Header.getData()}</h1>

<p>${Body.getData()}</p>
```

- Both Structures and Templates can be created/edited by navigating to Web Content in *Site Administration* via the *Product Menu*.



EXAMPLE ARTICLE X.XML

- To access the source XML of an article, go to the edit page of the article and click the *View Source* option.
- Let's take a look at an example article. We'll call it `Example Article X.xml`:

```
<?xml version="1.0"?>

<root available-locales="en_US" default-locale="en_US">
    <dynamic-element name="Header" type="text" index-type="keyword"
    instance-id="mdyl">
        <dynamic-content language-id="en_US"><![CDATA[My Header]]>
        </dynamic-content>
    </dynamic-element>
    <dynamic-element name="Body" type="text_box" index-type="keyword"
    instance-id="opiq">
        <dynamic-content language-id="en_US"><![CDATA[My body <em>with</em>
        HTML.]]>
        </dynamic-content>
    </dynamic-element>
</root>
```

ADDING WEB CONTENT TO THE SITEMAP

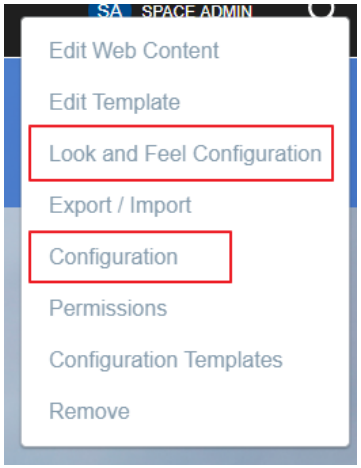
- To add a Web Content Display and choose a content article to display, we could do something like this example in the `sitemap.json`:

```
"columns": [
    [
        { "portletId": "com_liferay_hello_world_web_portlet_HelloWorldPortlet"
        {
            "portletId":
            "com_liferay_journal_content_web_portlet_JournalContentPortlet",
            "portletPreferences": {
                "articleId": "My Example Article X.xml",
                "groupId": "${groupId}",
                ...
            }
        },
...
```

- Here, we'll add the web content application right below the Hello World application.

PORTLET PREFERENCES

- You'll notice we use a `portletPreferences` property to select the article we will display in the Web Content Display application.
- `portletPreferences` are properties that store basic application configuration data.
- Just as you can apply some configurations for applications through the UI, *portletPreferences* properties allow you to set configurations for applications in the `sitemap.json` file.



COMMON PORTLET PREFERENCES

- Here are a few commonly used `portletPreferences` properties:

`articleId`	Selects an article
`groupId`	Selects a site
`portletSetupTitle_en_US`	Sets custom title
`portletSetupUseCustomTitle`	Allows use of custom title
`portletSetupPortletDecoratorId`	Sets Application Decorator

- Some `portletPreferences` properties are general, while others are application-specific.

EXAMPLE PORTLET PREFERENCES

- In the previous example, we are taking advantage of `portletPreferences` to configure the Web Content Display application.

```
"portletPreferences": {
    "articleId": "My Example Article X.xml",
    "groupId": "${groupId}",
    ...
}
```

- With the `articleId` property, we choose which article to display in the Web Content Display application.
  - The `Example Article X.xml` will be located in the `journal/articles` folder of our example Resource Importer module.
- The `groupId` property determines the site where the content will be displayed.

SETTING APPLICATION DECORATORS

- It is also possible to set the application decorator from a `sitemap.json` where `portletSetupPortletDecoratorId` is the id of the decorator to be used:

```
{
    "portletId": "com_liferay_journal_content_web_portlet_
    JournalContentPortlet",
    "portletPreferences": {
        "articleId": "My Content.xml",
        "groupId": "${groupId}",
        "portletSetupPortletDecoratorId": "barebone"
    }
}
```

- In this example, the application decorator is set to the *barebone* setting.
- The `portletSetupPortletDecoratorId` property can be set to any of the application decorators available (i.e., *barebone, borderless, decorate*).

SETTING CUSTOM DECORATORS

- The `portletSetupPortletDecoratorId` preference could be set to our custom decorator id as well. The preference can be set when embedding applications in a theme:

```
<#assign VOID = freeMarkerPortletPreferences.setValue
("portletSetupPortletDecoratorId", "barebone") />
<div aria-expanded="false" class="collapse navbar-collapse"
id="navigationCollapse">
    <#if has_navigation && is_setup_complete>
        <nav class="${nav_css_class} site-navigation" id="navigation"
        role="navigation">
            ...
        </nav>
    </#if>
</div>

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

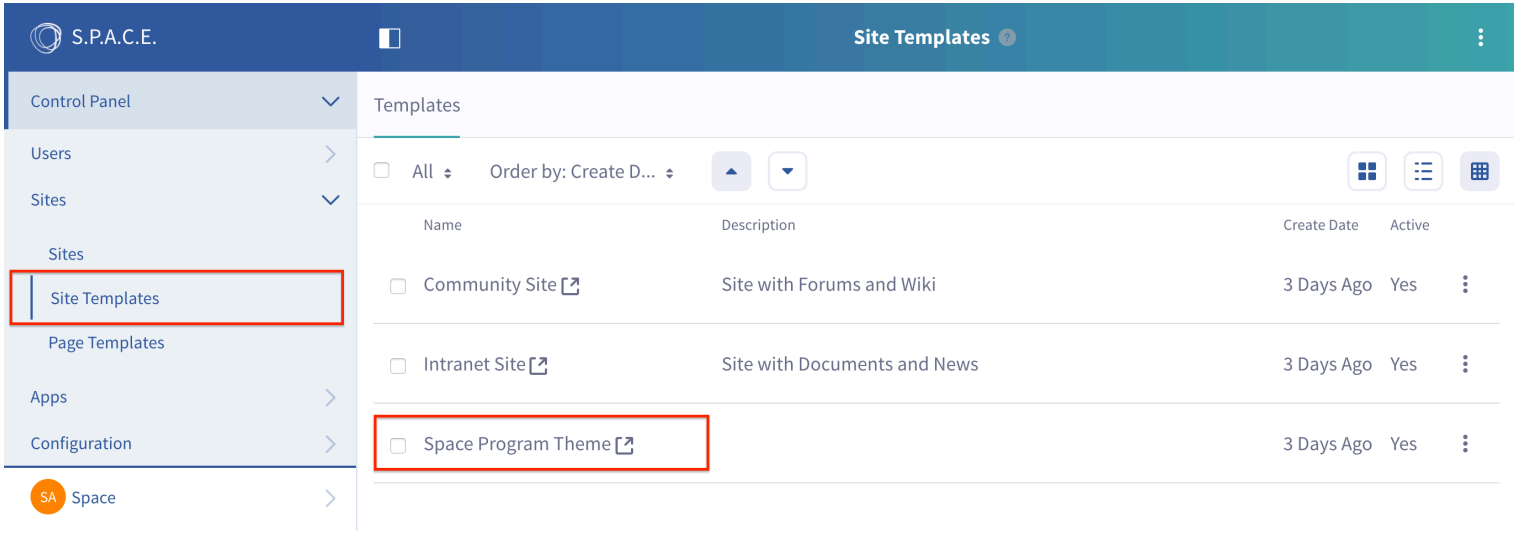
- Application decorators can also be updated from the application's *Look and Feel* menu.

EXERCISE: BUNDLING RESOURCES IN OUR THEME

1. **Go to** `space-program-theme/src/WEB-INF`.
2. **Create** a folder named `src`.
3. **Copy** the `resources-importer` folder from `exercises/front-end-developer-exercises/03-theme-development/09-importing-resources`.
4. **Paste** the entire folder into `space-program-theme/src/WEB-INF/src`.

DEPLOYING RESOURCES

- With the necessary `sitemap.json` and resources, you can deploy the theme.
- With the default configuration, the Resources Importer will create a Site Template that shares the name of the theme.
- We can also create a new site that includes the resources on the page.
- We'll do this to create a separate Space Program site that will display different assets from S.P.A.C.E.



IMPORTING RESOURCES TO EXISTING SITES

- To configure the Resources Importer to import resources into a site, rather than a site template, add the following properties to: `{theme-name}/src/WEB-INF/liferay-plugin-package.properties`.

```
...
resources-importer-target-class-name=com.liferay.portal.kernel.model.Group

resources-importer-target-value=[site-name]
...
```

- If the `site-name` is an existing site, the Resources Importer will import into that site.
- When the site doesn't exist, the Importer will create it for you.
  - **Note:** This can be a great way to show off your theme's feaures, or demonstrate how to arrange content nicely.

EXERCISE: UPDATING OUR PACKAGE PROPERTIES

1. **Go to** the `exercises/front-end-developer-exercises/03-theme-development/09-importing-resources` folder.
2. **Copy** the `liferay-plugin-package.properties` file.
3. **Go to** the `space-program-theme/src/WEB-INF`.
4. **Paste** the file to overwrite the existing `liferay-plugin-package.properties`.
5. **Run** `gulp deploy`.

SELECTING THE THEME

- The Resources Importer created a new site called *Space Program*, but the theme may not be automatically applied to it.
- You may need to manually apply the theme to the *Space Program* site to see the full styling:
  1. **Click** the *Site Selector* under *Site Administration* in the *Menu*.
  2. **Click** the *My Sites* tab.
  3. **Choose** the *Space Program* site.
  4. **Click** *Private* Pages under *Navigation*.
  5. **Click** *Options→Configure* to the right of *Private Pages*.
     - **Note:** if the *Options* menu is not visible, you may need to use the *Options* menu on *Private Pages*.
  6. **Click** *Change Current Theme*.
  7. **Choose** the *Space Program* theme.
  8. **Click** *Save*.

DEVELOPER MODE

- The Resources Importer has a developer mode, which deletes and re-creates the target site or site template on each deploy.
- To enable developer mode, add the following to: `liferay-plugin-package.properties`

```
...
resources-importer-developer-mode-enabled=true
...
```

- Themes created via the Liferay Theme Generator have developer mode enabled by default.
- Although this is useful for development, it should never be used in a production environment.

Notes: