Build React Apps for Liferay DXP

Learning Objectives

- Understand the various methods of connecting React apps to Liferay DXP
- Learn the best practices and things to avoid when building and adapting React apps for Liferay DXP

Tasks to Accomplish

- Use Headless APIs to Connect a React App to Liferay DXP
- Add a React Remote App to Liferay DXP as a widget

Exercise Prerequisites

- Java JDK installed to run Liferay
  - Download here: https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
  - Instructions on installation here: https://www.java.com/en/download/help/download_options.xml
- Unzipped module exercise files in the following folder structure:
  - Windows: `C:\liferay`
  - Unix Systems: `[user-home]/liferay`
- Node.js installed (v 16.13.2 used in exercise videos)
- Create React App installed
  - To install, run `npm install -g create-react-app` in your terminal
- A create-react-app project created with the appropriate files replaced with those located within the *prerequisites* folder

- The React-Bootstrap framework installed in your project
  - To install, first run `npm install react-bootstrap`, then `npm install bootstrap` in your terminal
- React Router v5.2 installed in your project
  - To install, run `npm install react-router-dom@5.2.0` in your terminal
- An IDE, such as Visual Studio Code, installed
-

An running instance of Liferay DXP 7.4 with the following contents:

- A picklist called *Account Types* with options for *Savings*, *Checking*, and *Individual Retirement*.
-

  A Liferay Object called *Bank Accounts* with the following fields:

Label	Type	Mandatory
Account Holder	Text	Yes
Account Number	Integer	Yes
Account Type	Picklist	Yes
Account Balance	Decimal	Yes

-

  At least two entries for the *Bank Accounts* Object with your Administrator as the *Account Holder* (three will be demonstrated)

Table of Contents

Use a Remote App with Liferay DXP

Exercise Goals

- Observe the ability of remote applications to connect to Liferay DXP
-

Observe the capabilities of Liferay DXP's Headless APIs

View the Faria Financial Manager App

1. **Open** the Faria Financial Manager App at `localhost:3000`.
2. **Click** the arrows in the carousel to view the different Accounts.
    - You should have at least two Accounts.

View the Bank Accounts Object in Liferay DXP

1. **Sign in** to your your instance of Liferay DXP at `localhost:8080`.
    - Your Sign In credentials should be the email `test@liferay.com` and the password `test`.
2. **Open** the *Site Menu*.
3. **Go to** *People → Bank Accounts* to view the added Bank Accounts.
    - The Bank Accounts Object should be scoped to the Site level and placed under *People* in the *Site Menu*.
    - Click the ID numbers to view or edit specific entries. To add entries, click the *Add* button.

View the Index.js File

1. **Go to** your React application `src` folder.
2. **Open** the `index.js` file.
    - Note that we are rendering the App in Strict Mode.

View the App.js File

1. **Open** the `App.js` file.
2. **View** the Router locations.
    - Although this App does not need to use Browser Router, doing so allows the application's functionality to be expanded. For example, you could add detailed pages for each account and add additional routes to the App.js file.

View the Account Files

1. **Open** the `Account1.js` file.
2. **View** the `componentDidMount()` section.
    - Note the line for fetching the Headless API. `/c` indicates that the Headless API was created from a custom Liferay Object.
3. **View** the Basic Authorization.
4. **View** the `render()` section.
    - Note the error handling lines.
5. **Open** the `Account2.js` file and view.
6. **Open** the `Account3.js` file and view.

Enable the CORS URL Pattern

1. **Open** the *Global Menu* in your Liferay instance.
2. **Go to** *Control Panel > Configuration > System Settings*.
3. **Click** *Security Tools*.
4. **Click** *Portal Cross-Origin Resource Sharing (CORS)*.
5. **Select** *Default Portal CORS Configuration*.
6. **Add** the `/o/c/*` URL Pattern if you have not already.
    - This enables CORS for all Liferay Objects, so that the Data can be used with Remote Apps.

View the Accounts.js File

1. **Open** the `Accounts.js` file.
2. **View** the Imports.
   - Note the Carousel import from the react-boostrap framework.
3. **View** each `Carousel.Item`.
   - Note how each `Carousel.Item` renders a different Account number class object.

View the Dashboard.js and style.css Files

1. **Open** the `Dashboard.js` file.
   - `Dashboard.js` renders the `Accounts.js` file.
2. **Open** the `style.css` file to view the styling.

Add a React Remote App to Your Platform

Exercise Goals

- Register a React Remote App with Liferay DXP
-

Add the App to a Page as a Remote App Widget

Register the Faria Financial Manager App as a Remote App

1. **Open** an instance of Liferay DXP at `localhost:8080`.
    - Make sure you have the Faria Financial Manager App running at `localhost:3000`.
2. **Open** the *Global Menu*.
3. **Select** *Remote Apps* under *Custom Apps* in the *Applications* tab.
4. **Click** the *Add* icon.
5. **Type** `Faria Financial Manager` as the name.
6. **Open** the *Type* drop down.
7. **Select** *IFrame*.
    - Note: Since the Faria Financial Manager is a fairly simple App, we are using IFrame. More complex Apps should use Custom Element.
8. **Type** `http://localhost:3000/` as the URL.
9. **Click** *Publish*.

Create a Financial Management Page

1. **Go to** the main site.
2. **Go to** *`Site Builder > Pages`* in the *Site Menu*.
3. **Click** the *Add* icon.
4. **Choose** *Public Page*.
5. **Choose** *Blank* for the Page Template.
6. **Type** `Financial Management` as the *Name*.
7. **Click** *Add*.

Add the Faria Financial Management Widget to the Financial Management Page

1. **Click** the *Fragments and Widgets* (gray plus sign) icon in the far right toolbar.
2. **Drag and Drop** a *Container* layout element onto the page.
3. **Click** the *Widgets* tab.
4. **Click** the *Remote Apps* category.
5. **Drag and Drop** the *Faria Financial Manager* widget into the *Container*.
6. **Click** *Publish*.
7. **Click** the *Financial Management* page to view the widget.

Bonus Exercise

1. Create a dashboard page for Mondego Group customers. Be sure to include the Faria Financial Manager App as well as areas for blog posts, comments, and a profile.

Build React Apps for Liferay DXP Module Quiz

1. Which of the following is *not* a project type that can be adapted to run on Liferay DXP using the Liferay JS Generator?

   A. React

   B. jQuery

   C. Angular

   D. Vue CLI

2. The frontend-js-react-web module provides a single, common version of React for all React consumers across Liferay DXP and includes common hooks, a React component tag, and a standard method for mounting components in Liferay DXP.

   A. True

   B. False

3. Which of the following methods can be used to connect remote applications to Liferay DXP?

   A. Headless REST APIs

   B. Headless SOAP APIs

   C. Headless RPC APIs

   D. Headless BATH APIs

4. Due to the nature of React, it is generally preferable to break up an application into its React components and create separate React widget projects for each one when adapting a React application to run on Liferay DXP.

   A. True

   B. False

5. Which of the following are valid methods of integrating a React application with Liferay DXP? (Choose all correct answers)

   A. Use Service Builder to create a remote connection

   B. Establish a remote connection via Headless APIs

   C. Use Liferay Objects to create a container for the app

   D. Use the adapt tool to create an addable widget

   E. Separate into components to create multiple widgets

Answer Key

1. B
2. True
3. A
4. False
5. B, D, E