PreviewerX (Apryse)

All about PreviewerX

- PreviewerX Overview - Architecture, Building, Running, Deploying
- Running WebViewer Locally

PreviewerX Overview - Architecture, Building, Running, Deploying

Base Project Architecture

General Info

apryse-webviewer - Repos (azure.com)

Main branch - previewerX/8.12-Main - this is our main branch because we can't use the actual Main. This is because we originally forked off of the Apryse Gihub project and we need to preserve their branches so we can more easily pull updates in the future if needed.

After cloning, you'll need to make sure you've checked out the previewerX/8.12-Main branch, since the default master branch won't have the Revver-specific changes and you won't be able to get very far through this installation guide.

There's 3 main pieces to PreviewerX and depending on the work you're doing it may change which ones you'll be interacting with the most.

Demo Project - /Demo

This project is not deployed anywhere, it is simply an internal project to speed up testing so you don't have to constantly move files into Utopia/Atlantis in order to see your changes. One nice thing about this project is that you can make changes in the PreviewerX project and with a quick reload, you'll see those changes come through.

If you want to run the demo project pointed at your local files, please see Previewer Setup for Local Environment. However, if you're only making changes to the PreviewerX code and not any interactions specifically with Utopia, you can also point the URLs in the form to staging which are provided as the default values.

PreviewerX - /PreviewerX

This is the main project where we try to put the majority of our code. It provided a wrapper for the Apryse Previewer objects and it provides the interface that Utopia (and eventually Atlantis) will see when you integrate your changes into Utopia.

ApryseUI - Root folder and /src

The other 2 projects are nested into this main project folder. This project is the UI code provided from Apryse and was originally forked from their repo in Github.

This is a dense project to initially dig into and we try to avoid making changes in this area, mostly because any changes we make in here will make it harder for us to stay up to date when Apryse releases new versions and will inevitably cause merge conflicts. The main reason you might find yourself making changes in here is because the internal state of the app is not visible or modifiable outside of their React app. Most of the time, that's okay, there's an interface to hook into the internal events, and that would be the preferable method. However, we've had to add a few modals directly into the previewer itself such as watermarks where we want to block interaction behind the scenes but need to provide a modal that's still within the bounds of the previewer itself.

Build and Run this bad boy

As mentioned there's 3 separate projects inside this repo, but we've added some convenience scripts to make building and launching as easy as possible.

We've all generally been using Node version 14+ and had no issues. If you have Node 11 (from the less cool and older previewer) it will cause issues during build time. You'll also need to have Python version 3.11 installed; if you have version 3.12 installed, you'll get lots of errors and you'll waste half your day troubleshooting and get super sad.

NOTE: All the following command and some more can be found in the root package.json file

1. Open a terminal in the root folder
2. `npm run install-all`
   - This will go into each project and run npm i for you
3. `npm run download-webviewer`
   - This will install the correct version of the "Core" webviewer files, these are the super secret libraries that power the UI
   - You MUST do this before running the next command, the build command takes these files and puts them where the demo project needs them in order to properly run
4. `npm run build-all`
5. Congrats, everything should be built now! You have a couple options from here:
   1. To launch the demo project and test things out use `npm run demo`
   2. To build the standalone file that is transferred to Utopia use `npm run build-standalone`

1. This will output a js file into the /PreviewerX/Standalone folder that can be copied over to the Utopia/wwwroot/scripts/previewer folder

Running WebViewer Locally

Local Webviewer Server with Local Utopia

Preface

This guide is assuming you already have previewerX building and running. If you leave the Web Server Url and 'Connecting to local Utopia and Webviewer' checkbox unchecked, PreviewerX will run client side only and will only preview PDF files properly. This is fine for most use cases, but if you need to work with annotations, redactions, or any page manipulation functions you'll need the Webviewer Server running.

Instructions

1. If you don't have it already, download and install Docker Desktop
2. Download the docker-compose.yml file, preferably somewhere you can find it
3. Open a command prompt in that folder and run: `docker-compose up -d --force-recreate`
    1. This will take a few seconds to spin up the first time, you can test if it's up by going to http://localhost:8090/demo?s
4. Navigate to the Utopia project (The main project, not the base repo) from the repo base
    1. Find the .vs/Utopia/config/applicationhost.config file and open in an editor like notepad++
    2. Find the <sites> section and the <site name="Utopia"> section inside of it

3. Add the additional binding
   `<binding protocol="https" bindingInformation="*:44334:host.docker.internal" />`
   the full Utopia Site section should look similar to this:

```
<site name="Utopia" id="2">
  <application path="/" applicationPool="Utopia AppPool">
    <virtualDirectory path="/" physicalPath="C:\src\Utopia\Utopia\Utopia" />
  </application>
  <bindings>
    <binding protocol="http" bindingInformation="*:57584:localhost" />
    <binding protocol="https" bindingInformation="*:44334:localhost" />
    <binding protocol="https" bindingInformation="*:44334:host.docker.internal" />
  </bindings>
</site>
```

5. You may need to restart Visual Studio if it was already running
6. If you're testing in the PreviewerX Demo project, enter the following values into the form on the side:
   Base Url - `https://localhost:44334`
   Web Server Url - `http://localhost:8090`
   Check the box for 'Connecting to local Utopia and Webviewer'
   (Make sure there isn't whitespace at the end of the URL, but that would never happen to you, right?)

Congrats! Everything should be working locally now!

Technical Details

To add technical details in case there's any issues later on.. under the hood we were having issues running locally be the webviewer running in docker couldn't make a call out to Localhost on the host machine, so it was impossible to load the file. Fortunately, docker desktop adds a host.docker.internal dns entry under the hood that allows docker containers to reach the host machine. I'll be honest, I'm not sure if it's mapped to the loopback on the host somehow, or if it's the local IP. I assume the local IP. To get everything running locally, we're simply overriding the base URL of the loadDocument call to use the host.docker.internal address. However, that means we also need to add that as a viable binding in IISExpress, so we modify the binding in the applicationhost.config file. I'm looking to see if there's a way to just add this by default to Utopia, but for now, hopefully this isn't too painful.

Problems and Solutions

Sometimes Docker gets confuse and you get a "Error: (HTTP code 500) server error - Ports are not available: exposing port TCP 0.0.0.0:60001 -> 0.0.0.0:0: listen tcp 0.0.0.0:60001: bind: An attempt

was made to access a socket in a way forbidden by its access permissions." from the loadbalancer docker image. Usually you'll notice this because previewerX can't connect to localhost:8090 even though everything is running. When this happens try this:

1. Stop the Apryse Docker images
2. Open an admin/elevated command prompt and run:
   `net stop winnat`
3. Restart the docker images or re-run `docker-compose up -d --force-recreate` from the folder with the docker-compose.yml file
4. Once everything is green, go back to the command prompt and run:
   `net start winnat`