Getting started with service workers

This document explains how to enable Angular service worker support in projects that you created with the Angular CLI. It then uses an example to show you a service worker in action, demonstrating loading and basic caching.

Prerequisites A basic understanding of the information in Introduction to Angular service workers.

Adding a service worker to your project

To set up the Angular service worker in your project, use the CLI command `ng add @angular/pwa`. It takes care of configuring your application to use service workers by adding the `@angular/service-worker` package along with setting up the necessary support files.

```
ng add @angular/pwa --project *project-name*
```

The preceding command completes the following actions:

1. Adds the `@angular/service-worker` package to your project.
2. Enables service worker build support in the CLI.
3. Imports and registers the service worker in the application module.
4. Updates the `index.html` file:
   - Includes a link to add the `manifest.webmanifest` file.
   - Adds a meta tag for `theme-color`.
5. Installs icon files to support the installed Progressive Web App (PWA).
6. Creates the service worker configuration file called `ngsw-config.json`, which specifies the caching behaviors and other settings.

Now, build the project:

```
ng build
```

The CLI project is now set up to use the Angular service worker.

Service worker in action: a tour

This section demonstrates a service worker in action, using an example application.

Serving with `http-server`

Because `ng serve` does not work with service workers, you must use a separate HTTP server to test your project locally. Use any HTTP server. The following example uses the http-server package from npm. To reduce the possibility of conflicts and avoid serving stale content, test on a dedicated port and disable caching.

To serve the directory containing your web files with `http-server`, run the following command:

```
http-server -p 8080 -c-1 dist/<project-name>
```

Initial load

With the server running, point your browser at http://localhost:8080/. Your application should load normally.

Tip: When testing Angular service workers, it's a good idea to use an incognito or private window in your browser to ensure the service worker doesn't end up reading from a previous leftover state, which can cause unexpected behavior.

Note: If you are not using HTTPS, the service worker will only be registered when accessing the application on `localhost`.

Simulating a network issue

To simulate a network issue, disable network interaction for your application.

In Chrome:

1. Select **Tools** > **Developer Tools** (from the Chrome menu located at the top right corner).
2. Go to the **Network tab**.
3. Select **Offline** in the **Throttling** dropdown menu.

Now the application has no access to network interaction.

For applications that do not use the Angular service worker, refreshing now would display Chrome's Internet disconnected page that says "There is no Internet connection".

With the addition of an Angular service worker, the application behavior changes. On a refresh, the page loads normally.

Look at the Network tab to verify that the service worker is active.

Notice that under the "Size" column, the requests state is `(ServiceWorker)`. This means that the resources are not being loaded from the network. Instead, they are being loaded from the service worker's cache.

What's being cached?

Notice that all of the files the browser needs to render this application are cached. The `ngsw-config.json` boilerplate configuration is set up to cache the specific resources used by the CLI:

- `index.html`.
- `favicon.ico`.
- Build artifacts (JS and CSS bundles).

- Anything under `assets`.
- Images and fonts directly under the configured `outputPath` (by default `./dist/<project-name>/`) or `resourcesOutputPath`. See `ng build` for more information about these options.

Pay attention to two key points:

1. The generated `ngsw-config.json` includes a limited list of cacheable fonts and images extensions. In some cases, you might want to modify the glob pattern to suit your needs.
2. If `resourcesOutputPath` or `assets` paths are modified after the generation of configuration file, you need to change the paths manually in `ngsw-config.json`.

Making changes to your application

Now that you've seen how service workers cache your application, the next step is understanding how updates work. Make a change to the application, and watch the service worker install the update:

1. If you're testing in an incognito window, open a second blank tab. This keeps the incognito and the cache state alive during your test.
2. Close the application tab, but not the window. This should also close the Developer Tools.
3. Shut down `http-server`.
4. Open `src/app/app.component.html` for editing.
5. Change the text `Welcome to {{title}}!` to `Bienvenue à {{title}}!`.
6. Build and run the server again:

```
ng build
http-server -p 8080 -c-1 dist/<project-name>
```

Updating your application in the browser

Now look at how the browser and service worker handle the updated application.

1. Open http://localhost:8080 again in the same window. What happens?

```
<img src="generated/images/guide/service-worker/welcome-msg-en.png" alt="It still says Welco
```

What went wrong? Nothing, actually. The Angular service worker is doing its job and serving the version of the application that it has **installed**, even though there is an update available. In the interest of speed, the service worker doesn't wait to check for updates before it serves the application that it has cached.

Look at the `http-server` logs to see the service worker requesting `/ngsw.json`. This is how the service worker checks for updates.

1. Refresh the page.

The service worker installed the updated version of your application *in the background*, and the next time the page is loaded or reloaded, the service worker switches to the latest version.

More on Angular service workers

You might also be interested in the following: * App Shell. * Communicating with service workers.