Angular: App Startup (pre-V19)

During application startup, you may have activities that need to occur before the first page opens. One example is a splash screen.

> NOTE: This page applies to Angular v18 and earlier, as APP_INITIALIZER is deprecated in V19.
>
> See this page for how to do the same in v19: Angular: App Startup (V19 and later)

To make one occur, do these things:

1. Add an import of `APP_INITIALIZER` to your app.module.ts, like this:

   ```
   // This import added to hook into the initialization so we can delay opening pages for the splash
   container to display.
   import { APP_INITIALIZER } from '@angular/core';
   ```

2. The App Initalizer can act as a DI provider that will call any method you like.
   We will add a startup method call to the providers block, like this:

   ```
   providers:
   [
    {
     provide: APP_INITIALIZER,
     useFactory: initializeApp,
     multi: true,
     deps: [AppconfigService]
    },
   ```

   The above provider calls a method, named, `initializeApp`. This method must return a Promise<boolean>.

   > NOTE: The above call has a DI dependency. So, we add a deps list that includes it.

3. Still inside app.module.ts, define the `initializeApp` method, like this:

```
// This method is called when the app first starts up.
// It will load any initial config, and setup anything required.
export function initializeApp(_appsvcconfig: AppconfigService): () => Promise<boolean>
{
  console.log("AppModule-" + "_" + ":initializeApp - triggered.");

  // To ensure that dependencies are stood up and ready, we must wrap our logic in a lambda, instead of naked execution...
  let lambda = async (): Promise<boolean> => {

    console.log("AppModule-" + "_" + ":initializeApp - inside startup promise...");

    if(_appsvcconfig == null)
    {
      console.error("AppModule-" + "_" + ":initializeApp - appconfig service instance is null.");

      // Cannot startup app.
      return false;
    }

    console.log("AppModule-" + "_" + ":initializeApp - Sleeping...");

    await Runtime_Sleep(1000);

    console.log("AppModule-" + "_" + ":initializeApp - Sleep done. Calling LoadConfig...");

    let val = await _appsvcconfig.LoadConfig();

    console.log("AppModule-" + "_" + ":initializeApp - LoadConfig returned.");

    // Register any singletons...

    // Stand up any local config...

    // Report success...
    return true;
  };

  return lambda;
```

```
}
```

The above method returns a Promise (to the provider factory) that is actually a lambda method with an async body.

4. Inside the initializer’s lambda, we call a sleep method to slow down presentation, so a splash page can linger.
5. We can add any startup activities to this method as needed for initialization.

Revision #4
Created 16 March 2025 01:57:38 by glwhite
Updated 16 March 2025 02:04:33 by glwhite