Embedding Additional Files

You may need to include additional files in your application bundle that aren't part of your frontend (your `distDir`) directly or which are too big to be inlined into the binary. We call these files `resources`.

To bundle the files of your choice, you can add the `resources` property to the `tauri > bundle` object in your `tauri.conf.json` file.

See more about tauri.conf.json configuration here.

`resources` expects a list of strings targeting files either with absolute or relative paths. It supports glob patterns in case you need to include multiple files from a directory.

Here is a sample to illustrate the configuration. This is not a complete `tauri.conf.json` file:

tauri.conf.json

```
{
  "tauri": {
    "bundle": {
      "resources": [
        "/absolute/path/to/textfile.txt",
        "relative/path/to/jsonfile.json",
        "resources/*"
      ]
    },
    "allowlist": {
      "fs": {
        "scope": ["$RESOURCE/*"]
      }
    }
  }
}
```

NOTE

Absolute paths and paths containing parent components (`../`) can only be allowed via `"$RESOURCE/*"`. Relative paths like `"path/to/file.txt"` can be allowed explicitly via `"$RESOURCE/path/to/file.txt"`.

Accessing files in JavaScript

In this example we want to bundle additional i18n json files that look like this:

de.json

```
{
  "hello": "Guten Tag!",
  "bye": "Auf Wiedersehen!"
}
```

In this case, we store these files in a `lang` directory next to the `tauri.conf.json`. For this we add `"lang/*"` to `resources` and `$RESOURCE/lang/*` to the fs scope as shown above.

Note that you must configure the allowlist to enable `path > all` and the `fs` APIs you need, in this example `fs > readTextFile`.

```
import { resolveResource } from '@tauri-apps/api/path'
// alternatively, use `window.__TAURI__.path.resolveResource`
import { readTextFile } from '@tauri-apps/api/fs'
// alternatively, use `window.__TAURI__.fs.readTextFile`

// `lang/de.json` is the value specified on `tauri.conf.json > tauri > bundle > resources`
const resourcePath = await resolveResource('lang/de.json')
const langDe = JSON.parse(await readTextFile(resourcePath))

console.log(langDe.hello) // This will print 'Guten Tag!' to the devtools console
```

Accessing files in Rust

This is based on the example above. On the Rust side, you need an instance of the `PathResolver` which you can get from `App` and `AppHandle`:

```
tauri::Builder::default()
  .setup(|app| {
    let resource_path = app.path_resolver()
      .resolve_resource("lang/de.json")
      .expect("failed to resolve resource");

    let file = std::fs::File::open(&resource_path).unwrap();
    let lang_de: serde_json::Value = serde_json::from_reader(file).unwrap();
```

```
    println!("{}", lang_de.get("hello").unwrap()); // This will print 'Guten Tag!' to the
terminal

    Ok(())
  })
```

```
#[tauri::command]
fn hello(handle: tauri::AppHandle) -> String {
   let resource_path = handle.path_resolver()
      .resolve_resource("lang/de.json")
      .expect("failed to resolve resource");

    let file = std::fs::File::open(&resource_path).unwrap();
    let lang_de: serde_json::Value = serde_json::from_reader(file).unwrap();

    lang_de.get("hello").unwrap()
}
```

Edit this page

*Last updated on **Mar 25, 2023***