Hot Reload (Webpack)

The highest impact on your application's bootstrapping process has a **TypeScript compilation**. But the question is, do we have to recompile a whole project each time when change occurs? Not at all. That's why **webpack** HMR (Hot-Module Replacement) significantly decreases an amount of time necessary to instantiate your application.

Installation

Firstly, let's install required packages:

```
$ npm i --save-dev webpack webpack-cli webpack-node-externals ts-loader
```

Configuration

Then, we need to create a `webpack.config.js` which is a webpack's configuration file, and put it in the root directory.

```
const webpack = require('webpack');
const path = require('path');
const nodeExternals = require('webpack-node-externals');

module.exports = {
  entry: ['webpack/hot/poll?100', './src/main.ts'],
  watch: true,
  target: 'node',
  externals: [
    nodeExternals({
      whitelist: ['webpack/hot/poll?100'],
    }),
  ],
  module: {
    rules: [
      {
        test: /.tsx?$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
    ],
  },
  mode: 'development',
  resolve: {
    extensions: ['.tsx', '.ts', '.js'],
```

```
  },
  plugins: [new webpack.HotModuleReplacementPlugin()],
  output: {
    path: path.join(__dirname, 'dist'),
    filename: 'server.js',
  },
};
```

This configuration tells webpack few essential things about our application. Where sits an entry file, which directory should be used to hold **compiled** files, and also, what kind of loader we want to use in order to compile source files. Basically, you shouldn't worry to much, you don't need to understand the content of this file at all.

Hot-Module Replacement

In order to enable **HMR**, we have to open Nest application entry file (which is `main.ts`) and add few critical things.

```
declare const module: any;

async function bootstrap() {
  const app = await NestFactory.create(ApplicationModule);
  await app.listen(3000);

  if (module.hot) {
    module.hot.accept();
    module.hot.dispose(() => app.close());
  }
}
bootstrap();
```

And that's all. To simplify execution process, add those two lines into your `scripts` inside `package.json` file.

```
"start": "node dist/server",
"webpack": "webpack --config webpack.config.js"
```

Now simply open your command line and run below command:

```
$ npm run webpack
```

Once webpack started to **watch files**, run another command in the another command line window:

```
$ npm run start
```

A working example is available here.

Support us

Nest is an MIT-licensed open source project. It can grow thanks to the support by these awesome people. If you'd like to join them, please read more here.

Principal Sponsor

Sponsors / Partners