Building a Web-based IDE using Eclipse Theia

Paul Maréchal <paul.marechal@ericsson.com>
Agenda / Chapters

- What is Theia?
- Architecture
- Implementation
- Extensions
- Applications
- Demonstration
What is Theia?

- Theia is a framework for making web-based IDE-like applications.
- The framework is distributed as npm packages.
- Most packages add functionalities to your applications, others add tooling.
What does Theia address?

- Create remote or local IDE-like applications.
  - Use in the browser.
  - Use as a desktop application.
  - (Almost) same source.
What is being solved?

- How to run an IDE in/from the cloud?
- If running in a browser, then how to consume native tools?
- How to build an application: for web and/or desktop?
- How to make the IDE extensible/configurable?
What is NOT being solved?

- How to create development environments.
- How to serve different users (multi-tenancy).
Agenda / Chapters

- What is Theia?
- Architecture
- Implementation
- Extensions
- Applications
- Demonstration
Frontend / Backend

- **Frontend** runs the IDE as such in the browser.
- **Backend** is a Node.js server:
  - Serves the frontend **assets** and JavaScript **bundles**.
  - **Bridges** different processes (LS, DA, etc...)
  - Runs **services** for the frontend to use:
    - Plugins (theia, vscode, other namespaces...)
    - Filesystem (read/write, watch, etc)
    - Terminals
    - ...

(JSON-RPC: Notifications + Requests)

Initial Application Serving

Frontend

Backend

(Services)
Distributed Architecture

- **Frontend** might spawn web-workers (isolated threads).
- **Backend** can forward connections from frontend to various services.
- **PluginHost** process interfaces with the **Backend** for **plugins**.
- Plugins can run in said web-workers.
Connection Scope

- **One set of services per tab.**

![Diagram showing connection scope with tabs and backend, including LS, DA, etc plugins]
Alternative Electron setup

- Application entry point is the **Electron Main** process.
- Electron main process spawns the backend in a Node.js process.
- Browser is replaced by an **Electron Renderer** process.
Agenda / Chapters

- What is Theia?
- Architecture
- Implementation
- Extensions
- Applications
- Demonstration
Everything is an extension

- Extensions as npm packages.
- Extra meta-data for tooling to create applications.

"theiaExtensions": [
    {
        "frontend": "lib/browser/menu/browser-menu-module",
        "frontendElectron": "lib/electron-browser/menu/electron-menu-module"
    },
    {
        "frontend": "lib/browser/window/browser-window-module",
        "frontendElectron": "lib/electron-browser/window/electron-window-module"
    },
    {
        "frontend": "lib/browser/keyboard/browser-keyboard-module",
        "frontendElectron": "lib/electron-browser/keyboard/electron-keyboard-module",
        "backendElectron": "lib/electron-node/keyboard/electron-backend-keyboard-module"
    },
    {
        "frontendElectron": "lib/electron-browser/shutdown-hook/electron-shutdown-hook-module"
    }
],
Many runtimes/environments

- Same programming language, but very different runtimes:
  - Browser runtime cannot access the filesystem.
  - Node.js doesn't handle graphical components.

Browser JavaScript:

```javascript
const input = document.getElementById('game-input') as HTMLInputElement;

input.addEventListener('keydown', event => {
  if (event.code === 'Enter') {
    alert(input.value);
  }
});
```

Node.js Javascript:

```javascript
const readline = require('readline');

const prompt = readline.createInterface({
    input: process.stdin,
    output: process.stdout,
});

prompt.question('value?', input => {
    console.log(input);
});
```
Many runtimes/environments

- Theia's code is separated by target/runtime:
  - browser
  - common
  - node
  - electron-browser
  - electron-node (same as node, but for Electron apps)
  - electron-main later?
Modularity via D.I.

- Dependency Injection container modules using Inversify.js.
- Bind components to "contribution points".
- One container module per runtime.
- Modules are mounted together to create applications.
Agenda / Chapters

- What is Theia?
- Architecture
- Implementation
- Extensions
- Applications
- Demonstration
Extension build system

- Regular yarn install.
- Build using tsc.
Writing an extension

- Depend on "@theia/core" and other extensions.
- Write a DI module container:
  - Bind, unbind and rebind components.

@theia/languages, @theia/editor, @theia/core, @theia/navigator, @theia/process, @theia/output, @theia/filesystem, @theia/userstorage, @theia/debug, @theia/variable-resolver, @theia/console, @theia/callhierarchy, @theia/json, @theia/textmate-grammars, @theia/workspace, @theia/task, @theia/messages, @theia/typescript, @theia/preferences, @theia/python, @theia/plugin-ext, @theia/outline-view, @theia/keymaps, @theia/plugin, @theia/java, @theia/plugin-ext-vscode, @theia/git, @theia/tslint, @theia/editorconfig, @theia/cpp, @theia/bunyan, @theia/getting-started, @theia/metrics, @theia/java-debug, @theia/plugin-dev, @theia/mini-browser, @theia/cli, @theia/application-package, @theia/debug-nodejs, @theia/extension-manager, @theia/application-manager, @theia/electron, @theia/node-pty, @theia/plantuml, @theia/rust, @theia/generator-plugin, @theia/monaco, @theia/markers, @theia/file-search, @theia/editor-preview, @theia/merge-conflicts, @theia/preview, @theia/scm, @theia/yeoman-plugin, @theia/vscode-coverage, ...
Extension API documentation

- Some but not much.
- Read the sources.
Agenda / Chapters

- What is Theia?
- Architecture
- Implementation
- Extensions
- Applications
- Demonstration
Application build system

A bit more involved...

- Create a web application composed of some set of extensions.
  - Defined in package.json.
- Extensions are installed following yarn's dependency resolution.
  - Hoisting and deduplication happens.
- Bundle JavaScript files using webpack.
  - Using an auto-generated webpack config.
- Create a Node.js application to serve bundles and assets.
- Server should also run the various backend services.

- Most of the setup is automated...
- ...but it is important to understand what happens because it can be customized.
Setting up an application

- Simple package.json.
- Run "theia build" which will trigger generators.
- Generator sources are located under "dev-packages" in main repository.
Distributing a Web application

- Simply host the backend on a server
- Add your own wrapping/auth around it.
- Provide your users with a link.
Distributing an Electron application

- Bundle the Electron application build output
- Native dependencies → Need to build/bundle an application for each platform.
  - native-keymaps
  - node-pty
  - nsfw
Agenda / Chapters

- What is Theia?
- Architecture
- Implementation
- Extensions
- Applications
- Demonstration
Theia 3D Viewer
Theia 3D Viewer

- Basic prototype.
- Allows you to view .obj files within a Theia application.
- Demonstrates opportunities of the web and Theia:
  - WebAssembly support.
  - JavaScript language evolving fast.
  - WebGL/XR support (Vulkan soon?)