Building a Web-IDE based on Eclipse Theia for Smart Home

Jonas Helming
Building a **Web-IDE** based on Eclipse Theia for Smart Home

Jonas Helming
35 minutes on building a Web-IDE based on Eclipse Theia, which is capable of creating an App, which can be deployed on a Smart Home system, which is finally capable of turning a light switch on

Jonas Helming
35 minutes on building a Web-IDE based on Eclipse Theia, which is capable of creating an App, which can be deployed on a Smart Home system, which is finally capable of turning a light switch on AND OFF AGAIN

Jonas Helming
Idea of an AppStore

Tool to develop Smart Home Apps
Why web-based?

- Apps are small, expected time of usage is very short
  => No installation time provides a huge benefit

- We target at “hobby” developers
  => We want a slim stand-alone tool, not a plugin to an existing IDE

- There is a gateway connected to the network
  => We can pre-deploy the Tool on this gateway
Why Eclipse Theia?

- Extensible
- Well-architected
  - Extensible via DI
  - Extensive use of existing components
- Includes core features
  - Code Editing
  - Console
  - Workspace
  - Windowing
- Deployment
  - Browser-based
  - Desktop-based via Electron
- Open Source and Community
Theia architecture and deployment

- Local (Browser / Electron) - Eclipse Theia Frontend - Frontend extension
- Local (Electron and Node Server)
- Remote (Node Server) - Eclipse Theia Backend - Backend extension
Demo Time
Adding a custom Menu

```typescript
@injectable()
export class SmartHomeEditorCommandContribution implements CommandContribution {

    registerCommands(registry: CommandRegistry): void {
        registry.registerCommand(ScaffoldingCommand, this.scaffoldingHandler);
    }

    protected scaffoldingHandler() {
        return {
            execute: () => {
                // Do sth.
            }
        }
    }
}

@injectable()
export class SmartHomeEditorMenuContribution implements MenuContribution {

    registerMenus(menus: MenuModelRegistry): void {
        menus.registerMenuAction(SmartHomeMenus.SMART_HOME, {
            commandId: ScaffoldingCommand.id,
            label: ScaffoldingCommand.label
        });
    }
}
Create Project Wizard

Local (Browser / Electron)
- Handler
- Custom Dialog

Remote (Node Server)
- YeoMan
- YeoServer

Parameters
Dependency Editor

Local (Browser / Electron) → OpenHandler → Custom Editor (based on JSONForms)
Code Generator

Local (Browser / Electron)

Remote (Node Server)

Eclipse Instance

OnSaveHandler

Code Generator (REST Service)
Code Editor

Local (Browser / Electron) → Monaco Code Editor → JDT Language Server → Remote (Node Server)
Run a JUnit Test

Local (Browser / Electron) -> Custom Task -> Remote (Node Server) -> Console
Build and Deploy

Local (Browser / Electron) → Deploy Handler → DeployService → AppStore
Conclusion

Any questions?

Join us at our booth!

Jonas Helming
jhelming@eclipsesource.com
Evaluate this Session

Sign in and vote at eclipsecon.org

WITH

-1 0 +1