Building Web-based Modeling Tools with Theia and Che
Why web-based tools?

- **Accessibility**
  - No client installation
  - Access through a web link
  - Simple client updates
  - Physical resource sharing

- **Usability**
  - Modern UI look and feel
  - SWT vs. HTML5
  - GEF 3 vs. SVG

- **Maintainability**
  - Room for evolution
  - Availability of developers
A prototypical (modeling) tool
Key enablers for building domain-specific, web-based tools

- Eclipse Theia
  - Extensible cloud IDE
  - Default frontend for Eclipse Che

- Eclipse Che
  - Kubernetes-native IDE platform
  - Management of workspaces and dev environments

- Monaco & Language server protocol (LSP)
  - Protocol enabling the separation of editor (front-end) and language implementation (back-end)
  - Feature-rich and broadly adopted (VS Code)

⇒ Related talk: “Eclipse Theia and Che, explained and explored!”, Today 17:00, Theater Stage
Reinventing the wheel?

- Which components can be reused?
- What needs to be reimplemented?
- How do we separate frontend and backend functionality?
Typical reuse example: Code Generation

- **Browser**: JS/TS
  - **Eclipse Theia frontend**
  - **Frontend extension**

- **Node.js**: JS/TS
  - **Eclipse Theia backend**
  - **Backend extension**

- **JVM**: Java
  - **Code Generator**
Overall tool architecture
Demo overview

- One connected model for coffee makers
  - Structural model
  - Behavioral model
- Example IDE with the following features:
  - Tree-Editor for structural modeling with forms
  - Graphical editor for behavioral model
  - Code generation
  - Working with source code
  - Textual Modeling
  - Model Analysis
  - Multi-User Support
Demo: Form-based editing
Form-based editing

- **Tree**
  - Based on Theia Tree Widget
  - LabelProvider and ContentProvider

- **Detail-Form**
  - Based on JSON Forms
  - Declarative approach: JSON + UI Schema

- **Synchronization**
  - Based on EMF.cloud Model Server
  - Typescript-based Model Server client API
    - Push changes as commands
    - Subscribe to updates

⇒ Related talk: “Property editors in space”, Thursday 1pm, Theater Stage
Demo: Graphical Modeling
Graphical Modeling

- Graphical Language Server Platform (GLSP)
  - LSP for Graphical Editors
  - GLSP client:
    - generic
    - renders graphical visualization
  - GLSP server:
    - specific to DSL
    - maps model to graphical visualization
    - synchronization with model server
  - Based on Eclipse Sprotty

⇒ Related talk: “Diagrams in web and space with GLSP”, Thursday 15:10pm, Bürgersaal 2
Demo: Model Server
Model Server

![Diagram of Model Server with components and connections]

- **Model Server**
- **JSON**
- **Forms**
- **GLSP**
- **EMF.cloud Model Server**
- **LSP**

Model Server

- Component of EMF.cloud Eclipse project
- Features
  - Command-based change interface
  - Notification mechanism via sockets
  - Convenient model access with client APIs

⇒ Related talk: “Lifting the greatness of EMF into the cloud with EMF.cloud”, Wednesday 12:35pm
Demo: Generators - Model to Text
Generators: Model to Text

- Based on Eclipse Xtend
- Generator jar build with Maven
- Launched on demand via CLI
- Generates into selected Theia workspace folder
Demo: Working with source code
Working with source code

- **Enablers:**
  - Language Server Protocol (LSP)
  - Debug Adapter Protocol (DAP)
- **Theia Code Editor**
  - Monaco-based (VS Code)
  - Uses LSP to “understand” language
- **Theia Debug Extension (DAP)**
  - Uses DAP to support language debug
Demo: Textual Modeling
Textual Modeling

- Frontend: Theia Editor (Monaco)
- Backend:
  - DSL modeled as XText grammar
  - XText LSP Server for DSL
Demo: Model Analysis

50.0% of executions take this path

Push → Check Water → Refill Water → Drip Tray → Preheat
Model Analysis

- **Frontend:**
  - Uses D3 to visualize analysis results

- **Backend:**
  - Fetches data from model server
  - Calculates analysis result from data
Multi-User Support
Running Theia on Che

Che Server

Workspace 1

Workspace 2

Workspace 3
Excursion: Running applications on Kubernetes

Diagram showing a Master Node connected to multiple Node pods.
Excursion: Running Che and Theia on Kubernetes

- Browser
- Master Node
- Che Server Node
- Node

Che
Theia
WS
Config
Summary

● Web-based modeling tools are feasible today
● Reuse and migration easier than expected
● Web technology can leverage unique advantages
  ○ Modern UI and styling
  ○ Zero installation for users
  ○ Enables “cloud” business models

● There is open-source components
  ○ Eclipse Theia and Eclipse Che
  ○ EMF.cloud, LSP, GLSP, JSON Forms, XText, Sprotty and D3
  ○ Existing business logic can often be reused

● Demo code available: https://github.com/eclipsesource/coffee-editor

→ Important now: Define strategy and timeplan, build POC
Please evaluate the sessions.

Sign in and vote using the conference app or eclipsecon.org.

Thank you!
Towards a migration strategy

- **Now**: Define a strategy and timeplan, build POC
- **Short-term**: Consider for architectural decisions
- **Mid-term**:
  - Prepare architecture for migration **iteratively**
  - Migrate high-value use cases **iteratively**
  - Single-source components, enable reuse
- **Long-term**:
  - Migrate use-case by use-case **iteratively**
  - Deprecate desktop-based solution
Excursion: Language Server Protocol

- **Separation of concerns**
  - Tooling for editing code and textual DSLs
  - Language smarts: auto-completion, refactoring support

- **Advantages**
  - LSP-Client is language-agnostic
  - LSP-Server is tool-agnostic

---

Visual Studio Code

Language Server Protocol (JSON-RPC)

Language Server

User opens document

Notification: textDocument/didOpen; Params: document

User edits document

Notification: textDocument/didChange; Params: [documentURI, changes]

Notification: textDocument/publishDiagnostics; Params: Diagnostic[]

User executes "Code definition"

Request: textDocument/definition; Params: [documentURI, position]

Response: textDocument/definition; Result: Location

User closes document

Notification: textDocument/didClose; Params: documentURI

Server publishes errors and warnings
Excursion: Separation of Concerns with GLSP

Modeling Backend

Diagram Rendering
- Editing Tools
- Visual Feedback
- Editing Rules
- Commands
- Edit Transactions
- Live Validation
- Model Management

Client

Protocol

Server