Building Cloud-native (modeling) tools

Maximilian Koegel
What is EMF Cloud?

- Reusable components for (modeling) tools
  - Data model
  - Data model synchronization between editors
  - Editor support: Tree, Form, Graphical, Textual
  - Additional functionality: Validation, Diffing/Merging
- Reduce maintenance effort for products
- Industrial hardening
Where is EMF Cloud now?

- **Components**
  - Model Server based on Java+EMF
  - Editor support:
    - Tree: Tree-Editor (EMF in the backend)
    - Graphical: GLSP integration (EMF in the backend)
    - Form-based: JSON Forms integration (migrate from EMF Forms)
    - Textual DSLs: XText integration
  - Additional features: EMF Validation integration, EMF Compare integration
- **Demo products based on components:** Coffee Editor, Ecore Editor
What was missing in EMF Cloud?

- Focus on reuse is awesome for migration
- But reuse leads to compromise (for greenfield projects)
  - Mixed stack dev experience
  - Inflexible deployment
  - **Fat** backend
What is in the making for EMF Cloud?

- **Second line of components for greenfield projects**
  - Uniform TypeScript-based stack
  - Flexible deployment options
  - Slim to no backend

- **Status and timeline**
  - Based on components in production
  - Significant resources are and will be put into this
  - Q4 2023: legal release process
  - Q1 2024: official OS release
Demo

Model Management

Tree Views

Diagram Editors

Form Editors

Textual DSLs
Architecture: Centralized editor access via Model Hub

Model Hub

<table>
<thead>
<tr>
<th>Load/Get/Save Model</th>
<th>Read/Write Model</th>
<th>Undo/redo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolve references</td>
<td>Validate</td>
<td>Your Custom API</td>
</tr>
</tbody>
</table>
Architecture: Modularization by capabilities

Model Hub

modelService Contributions

Persistence<M>
- load(uri): M
- save(uri)

EditingDomain<M>
- get(uri): M
- undo() / redo(), patch()

CrossRefs<M>
- resolve(Ref<N ⊂ M>): N

Validator<M>
- validate(M): Diagnostics[]

CustomAPI<M>
- *(): *

M
Architecture: Capabilities facilitate reuse
Coffee Editor: Capability implementation

Diagram:

- Persistence\(<M>\)
- CrossRefs\(<M>\)
- Validator\(<M>\)
- EditingDomain\(<M>\)
- CustomAPI\(<M>\)

Flow:

- Langium
- JSON
- CommandStack
- Custom Implementation
Coffee Editor: Architecture

- Tree Editor
- GLSP Editor
  - Frontend
  - Backend
- Monaco Editor
- ModelHub
- Coffee Model Contribution
- Langium
- Workspace

Additional tools:
- CommandStack
- JSON
- Workspace
Java-based vs. TypeScript-based

- Value of existing code vs. ease of development
- Migration vs. deployment options
- More features vs. slim footprint

→ Very tool specific tradeoff
Summary

● EMF Cloud is enhanced by a second line of components
  ○ Uniform TypeScript-based stack
  ○ Flexible deployment options
  ○ Slim to no backend
● Based on components in production
● Release in Q1 2024
  ○ Interested now? => we can explore early access options

→ More info: [https://eclipse.dev/emfcloud](https://eclipse.dev/emfcloud) (Update: Christmas)
→ Contact: Maximilian, mkoege@eclipsesource.com