Diagram Editors with GLSP

Why flexibility is key

Philip Langer
planger@eclipsesource.com
Eclipse Graphical Language Server Platform (GLSP)
Eclipse Graphical Language Server Platform (GLSP)

Applying the architectural pattern of LSP to graphical modeling

- Development of browser-based diagram clients
- Frontend focused on rendering & user interaction
- Encapsulate language smarts on the diagram server
Eclipse Graphical Language Server Platform (GLSP)

- Initialization with parameters
  - URI
  - Diagram type
  - ...

![Diagram of GLSP client and server with init connection](image)
Eclipse Graphical Language Server Platform (GLSP)

- Server has responsibility to obtain source model
Eclipse Graphical Language Server Platform (GLSP)

→ Source model can be anything
  - EMF
  - JSON
  - Xtext
  - emf.cloud
  - Databases, REST, …
Eclipse Graphical Language Server Platform (GLSP)

- Server maps source model into *graph model*
Eclipse Graphical Language Server Platform (GLSP)

- Client translates *graph model* into SVG with Eclipse Sprotty 🍀
Eclipse Graphical Language Server Platform (GLSP)

- Editing tools on the client
  - Communicate with server via actions (defined in protocol)

- Extensible with custom tools and actions
  - Add domain-specific functionality
  - Adjust default behavior
Eclipse Graphical Language Server Platform (GLSP)
What's new?

- **API Stabilization and Server API Refactoring**
  - Flexibilization of source model technology
  - Ready-to-use modules for EMF, JSON, EMF.cloud

- **Node-based Server Framework**
  - Typescript for client and server
  - Alleviates runtime requirements

- **Documentation and Project Templates**
  - Getting started quickly for your architecture

and more
Why Flexibility is Key

● Not just about getting onto web stack / cloud
  ○ Architectural paradigm shift
  ○ Modern tech stack: fluid and diverse
  ○ Modularity and combinability

● Diagram editors: Specific to language by nature
  ○ Different domains
  ○ Different data sources
  ○ Different workflows
  ○ Different integrations with other tool components
Flexibility in GLSP’s Architecture
Full Flexibility and Customizability on the Client

- Direct access to excellent base technologies
  - Customizable editing tools with Typescript
  - Custom UI controls with HTML
  - Dynamic SVG views and CSS

- No limiting abstraction layers in the middle
  - Well known technologies
  - Benefit from full power
  - Great debugging experience
Maximum Reuse Across Tool Platforms

1. Editor registration, global menus, platform styling, etc.
2. Platform Integration

- @glsp/client
- my-diagram-client

Functionality expected from the platform
Maximum Reuse Across Tool Platforms

1. Platform Integration
   - Editor registration, global menus, platform styling, etc.

2. Client
   - my-diagram-client
   - @glsp/client
   - my-theia-integration
   - @glsp/theia-integration
   - @theia/*
Maximum Reuse Across Tool Platforms

1. Platform Integration
   Editor registration, global menus, platform styling, etc.

2. Client

- my-diagram-client
  - @glsp/client
  - my-theia-integration
    - @glsp/theia-integration
    - @theia/*
  - my-vscode-integration
    - @glsp/vscode-integration
    - vscode.*

Demo
Flexibility on the Server

1. Platform Integration
2. Client
3. Model Management
4. Server
Model Management is Your Choice

- Model Management
  - Format, structure, framework
  - Local filesystem or remote
  - Shared across users or isolated

- Base modules available

- Facilitates migration

- Enables reuse across multiple deployments

See Project Templates
A new Server Framework alongside Java

- GLSP servers can use any language
  - Protocol and some IO (e.g. socket)
  - More efficient to use existing base framework

- TypeScript GLSP Server Framework (1.0)
  - Homogeneous dev env for client & server
  - No need for a JVM on the user machine
  - Opens door to new deployment options
Deployment Options

Tool  Server  Source
Java  emf

Tool  Server  Source
TS  emf

Tool  Server  Source
TS  REST

Tool  Server  Source
Java  XML
Deployment Options

Platform Integration

Client

Server

Model Management

TS
Deployment Options
Deployment Options

1. Platform Integration

2. Client

3. Model Management

4. Server

Theia.cloud - Running Theia-based products in the cloud
Tomorrow, 17:00 CET, Bürgersaal 2
Deployment Options

1. Platform Integration
2. Client
3. Model Management
4. Server

No server at all
Conclusion

- Flexibility across all levels is key
  - Modern technology stack: more fluid and diverse
  - Multiple deployment / distribution channels

- Getting started is easier than ever
  - API stability with 1.0
  - Documentation
  - Project templates

- Try it out and get in contact with us!
  https://www.eclipse.org/glsp

Related Talks & Events

- Getting started with Theia
  The nextGen Eclipse Platform
  Today, 10:45 CET

- Introducing Eclipse CDT.cloud
  C/C++ tooling in the web
  Today, 11:45 CET

- BoF: Building (web-based) tools with Eclipse
  Today, 19:30 CET, Silchersaal

- Theia.cloud - Running Theia-based products in the cloud
  Tomorrow, 17:00 CET, Bürgersaal 2
Evaluate the Sessions

- Please help by leaving feedback on the sessions you attend!
- To rate a session, you must be registered for it in Swapcard BEFORE the talk starts.
- Swapcard will prompt you to leave feedback after the end of each session.
- You may also rate a talk by locating the session from the “Agenda” or “My Event” buttons on the Event Home page. Click on the session and look for the “Give your feedback” box.