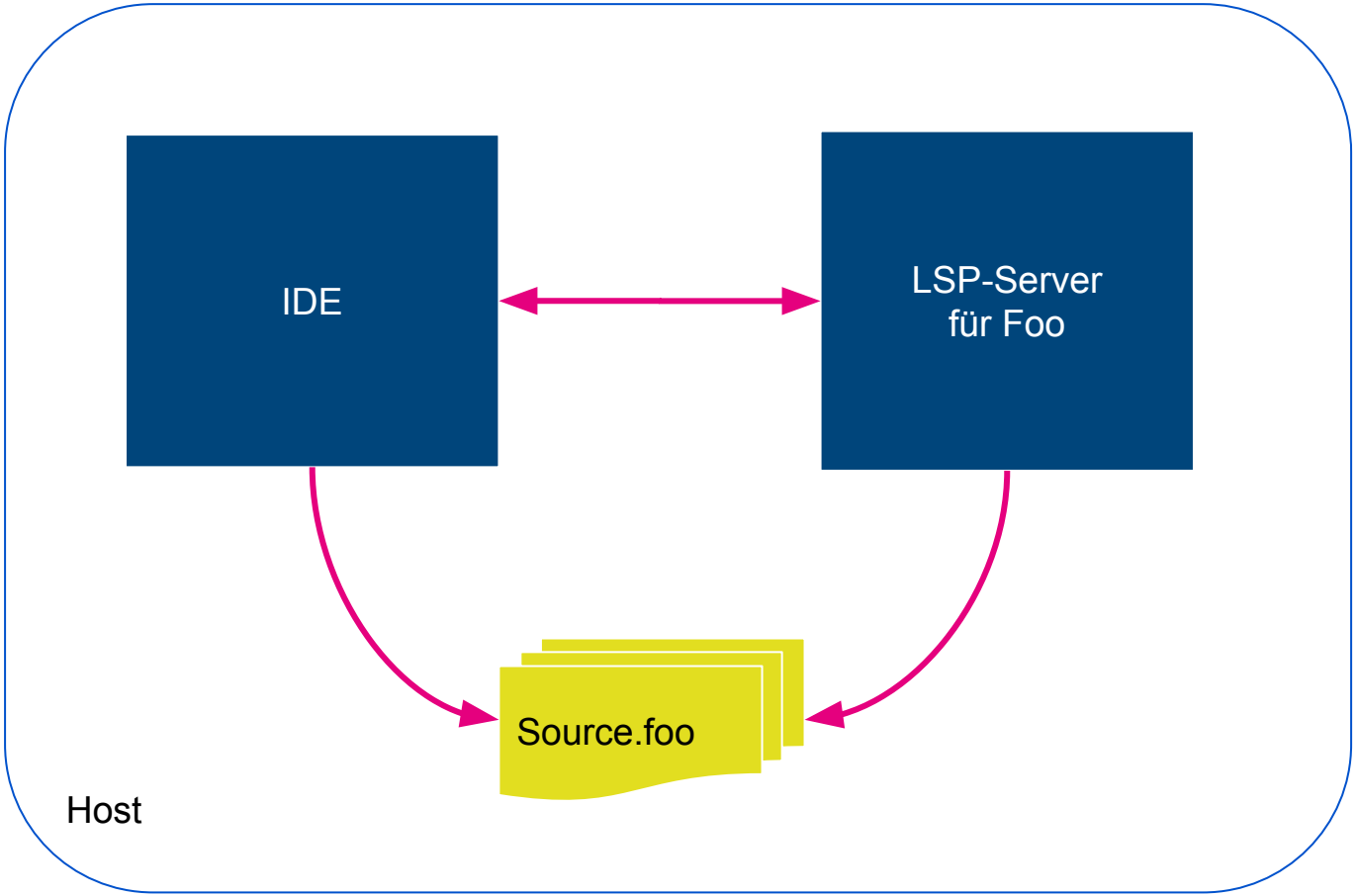
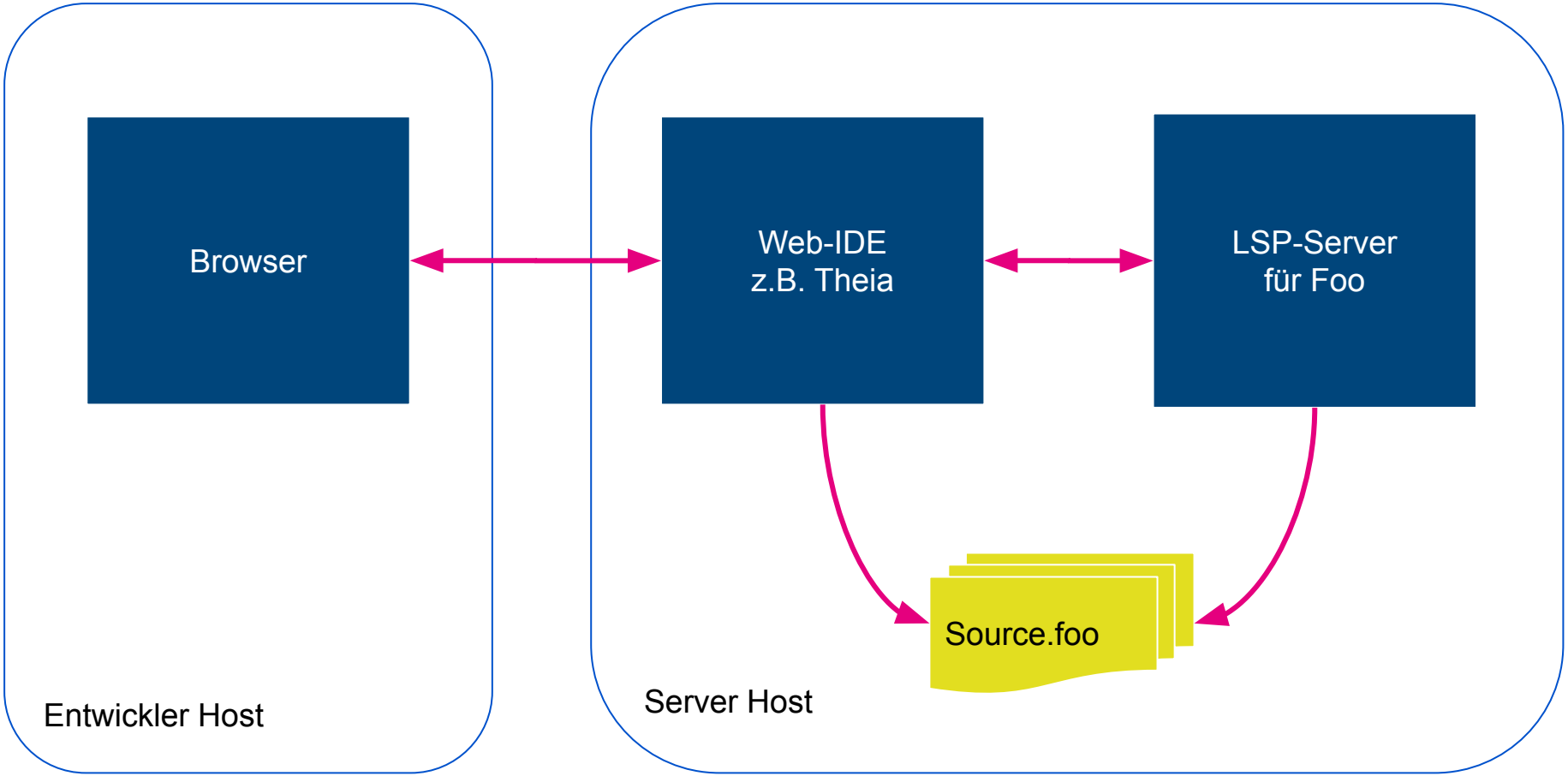
Language Server Protocol mit LSP4J und Xtext

Roland Meuel,Consultant

Language Server Protocol

- Von Microsoft für VS Code entwickelt.
- Seit 2016 ein offenes Protokoll.
- Basiert auf JSON-RPC und Messages
  - Request + Response
  - Notifications
- Entkoppelt IDE und Sourcecode-Editoren von konkreten Programmiersprachen
  - LSP-Server kennt $Programmiersprache
  - IDE kennt $Programmiersprache nicht, aber allgemeingültige Sprach- und Editor-Konzepte
  - LSP-Server liefert sprachspezifische Ausprägungen dieser Konzepte an die IDE
  - Für jedes konkrete Sourcecode File


IDE
LSP-Server
für Foo
Source.foo
Host


Browser
Web-IDE
z.B. Theia
LSP-Server
für Foo
Source.foo
Entwickler Host
Server Host

Beispiel für Notification

- DidOpenTextDocument

```
interface DidOpenTextDocumentParams {
	/**
	 * The document that was opened.
	 */
	textDocument: TextDocumentItem;
}
```

```
interface TextDocumentItem {
	/* The text document's URI. */
	uri: DocumentUri;

	/* The text document's language identifier. */
	languageId: string;

	/* The version number of this document (it will
	 *increase after each change, including undo/redo).
	 */
	version: number;

	/* The content of the opened text document. */
	text: string;
}
```

Ein paar LSP-Requests zu Language Features

- Completion
- Hover
- Signature Help
- Goto Declaration
- Goto Definition
- Goto Type Definition
- Goto Implementation
- Find References
- Document Highlight
  - aka “Mark Ocurrences”
- Document Link
- Document Formatting
- Range Formatting
- Prepare Rename
- Rename
- Folding Range

Syntax Highlighting

egamma commented on 12 Jul 2016 Member ...

> To make my question clearer: is syntax highlighting even supposed to be supported through the protocol at all (in the current version)?

@vladdu No syntax highlighting is intentionally not defined by LSP. Syntax highlighting is best done by the editor host (sublime, vscode, eclipse, atom) and not the server.

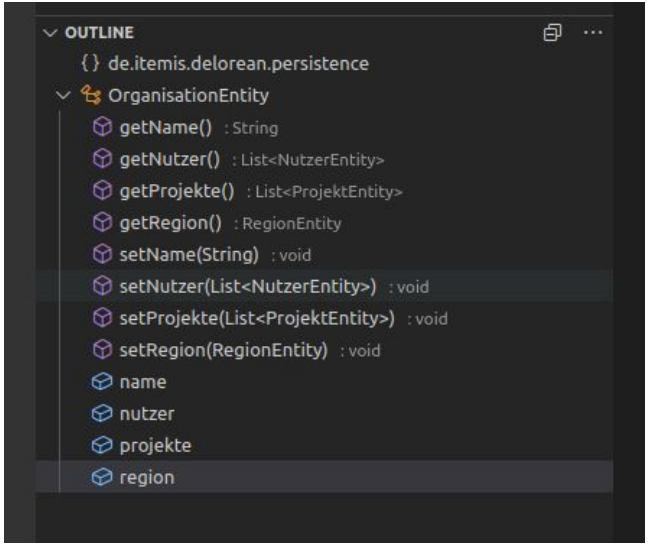
What is on the backlog for the language server protocol is to add support for semantic coloring/classification of tokens.

- Semantic Highlighting seit 2018 in Entwicklung und für das nächste LSP-Release 3.16 geplant.
- Highlighting durch die IDE oder den Editor setzt wieder irgendeine Form von Tokenizing (und Parsen) voraus
  - z.B. TextMate Grammatik in VS Code

Outline View

- Document Symbols Request

```
/**
 * Represents programming constructs like
 * variables, classes, interfaces etc. that
 * appear in a document. Document symbols can be
 * hierarchical and they have two ranges: one
 * that encloses its definition and one that
 * points to its most interesting range,
 * e.g. the range of an identifier.
 */
export interface DocumentSymbol {
     name: string;
     kind: SymbolKind;
     range: Range;
     children?: DocumentSymbol[];
     ...
}
```




```
export namespace SymbolKind {
     export const File = 1;
     export const Module = 2;
     export const Namespace = 3;
     export const Package = 4;
     export const Class = 5;
     export const Method = 6;
     export const Property = 7;
     export const Field = 8;
     ...
}
```

LSP4J

- Eclipse Projekt
- Framework für das Implementieren eigener LSP-Server
  - Für existierende Programmiersprachen
  - Für eigene Programmier- oder Modellierungssprachen (aka Domain Specific Languages, DSL)
- Für Ziel-IDE sprach-spezifisches Plugin bzw. Extension benötigt
  - Macht die Sprache in der IDE bekannt (z.B. Datei-Endung)
  - Startet die LSP-Instanz im Hintergrund
  - Kümmert sich um das Syntax Highlighting
-

Xtext für LSP-Server

- Xtext-Grammatik zum Beschreiben der eigenen DSL
- Xtext generiert Parser und Eclipse-Plugins
- Zusätzlich wird auch ein LSP-Fat-Jar erzeugt (~16MB)
  - Nutzt LSP4J, is ja klar
- VSCode Language Extension für DSL
  - Sobald Datei mit Suffix der DSL geöffnet wird und die Extension anspringt, wird der LSP-Server im Hintergrund hochgefahren

```
13  function getServerOptionsForEmbeddedServer(context: ExtensionContext): ServerOptions {
14      // Connect to the language server via a io channel
15      console.log("Starting Embedded Server");
16      let launcher = os.platform() === 'win32' ? 'mydsl-ls.bat' : 'mydsl-ls';
17      let script = context.asAbsolutePath(path.join('languageserver', 'bin', launcher));
18
19      console.log("Starting server: " + script);
20      let serverOptions: ServerOptions = {
21          run: { command: script, args: [context.asAbsolutePath(".")], options: { env: createDe
22          debug: { command: script, args: [context.asAbsolutePath(".")], options: { env: create
23      };
24      return serverOptions;
25  }
26
27  function getServerOptionsForRemoteServer(context: ExtensionContext): ServerOptions {
28      console.log("Connecting to server!");
29      let connectionInfo: net.TcpNetConnectOpts = {
30          port: 3000,
31          host: "localhost"
32      }
33
34      let serverOptions = () => {
35          let socket = net.connect(connectionInfo);
36          let streamInfo: StreamInfo = {
37              writer: socket,
38              reader: socket
39          }
40          return Promise.resolve(streamInfo);
41      };
42
43      return serverOptions;
44  }
```

LSP 3.16

- Semantic Token Support für LSP-basiertes Syntax Highlighting
- Type Hierarchies
- Funktionsumfang nähert sich klassischen IDEs weiter an
- Release-Datum ?

Fazit

- LSP für jüngere IDEs und Editoren interessant
  - Unterstützung von etablierten Programmiersprachen schnell anzubieten, falls ein LSP-Server bereits existiert (wovon auszugehen ist)
- LSP für eigene Sprachen (DSLs) interessant
  - Unterstützung kann schnell in vielen IDEs und Editoren angeboten werden, sofern sie einen LSP-Client anbieten (wovon auszugehen ist)

Fragen?

Roland Meuel
roland.meuel@itemis.com