Microsoft

使用TypeSpec创建MCP工具服务

石辰杰

微软高级开发工程师 Azure SDK团队

目录

- TypeSpec介绍
- TypeSpec MCP扩展
- 用TypeSpec实现一个MCP工具服务

TypeSpec介绍

- 为什么会有TypeSpec
  - Azure SDK团队
    - 管理来自不同Azure服务团队的API
    - 6种主流语言的SDK
    - Azure服务的API规范，以及相关工具链
  - 解决Azure SDK团队遇到的挑战
    - OpenAPI的局限性
    - 难以将规范固化，Azure服务团队不容易遵守
    - 支持更多协议，支持更多场景
    - 支持Azure服务不断增长的规模

TypeSpec介绍

- TypeSpec是什么 

  - 一种可扩展的API定义语言
    - 更容易的定义API
  - 一套完整的工具链
    - 更好的通过扩展实现围绕API的各类产品开发
  - 核心组件在5月初GA：[TypeSpec 1.0 GA: API First, Made Practical]

- 面向3P提供更好的API开发体验

TypeSpec介绍

- TypeSpec基本语法
  - 类TypeScript和C#的语法，完整的语言工具支持（LSP Server，VSCode/VS插件），很容易上手

```
namespace DemoService;

model Widget {
  id: string;
  weight?: int32;
  color: "red" | "blue";
}

op getWidget(id: string): Widget;
```

TypeSpec介绍

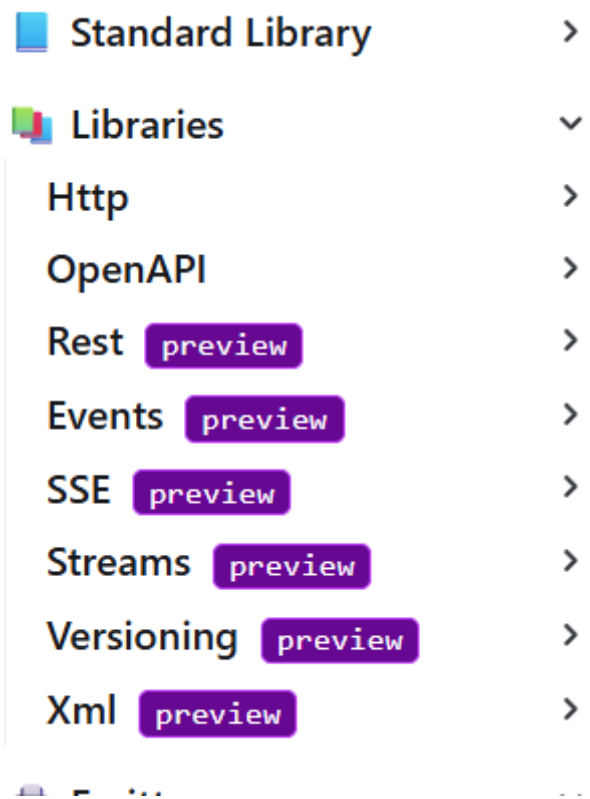
- TypeSpec优势在于它的扩展性
  - Template：可以方便的定义和复用各种API模式
    - REST
    - Azure Service
  - Decorator：扩展任何内置类型，实现不同的协议、序列化方式、客户端接口定义等
    - HTTP
    - MCP

```
model Widget {
  @key id: string;
  weight: int32;
  color: "red" | "blue";
}

@error
model Error {
  code: int32;
  message: string;
}

interface WidgetService extends
Resource.ResourceOperations<Widget,
Error> {
  @get @route("customGet")
customGet(): Widget;
}
```

TypeSpec介绍

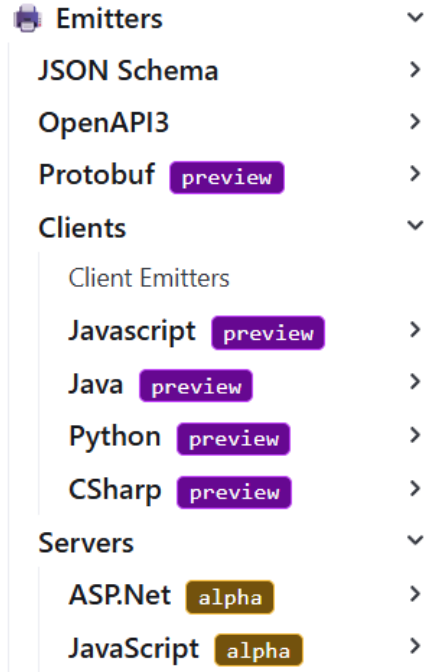
- 通过NPM包进行分发
  - Library：封装不同功能以供复用

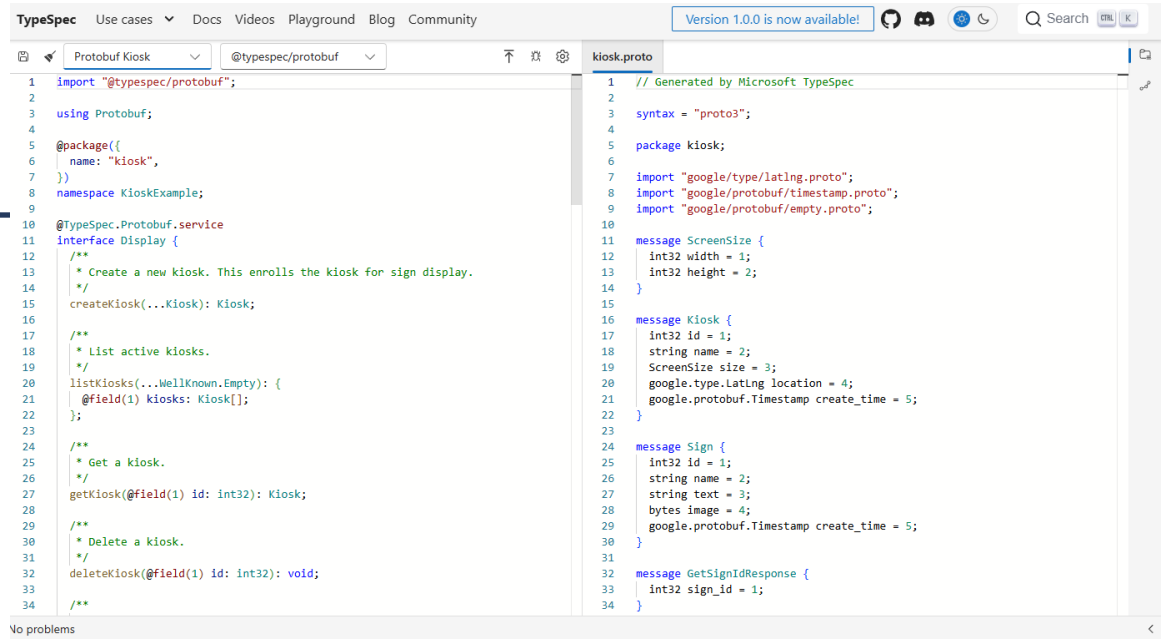
```
import "@typespec/http";

using Http;
@service(#{ title: "Widget Service" })
@useAuth(BasicAuth)
namespace DemoService;
```



TypeSpec介绍

- TypeSpec工具链
  - 检查工具
    - Lint
  - 生成工具
    - 现有的API规范
    - 客户端SDK
    - 服务端骨架
  - 开发者工具
    - Playground
    - 组件化生成框架






```
/**
 * Main function to handle the emission process.
 * @param context - The context for the emission process.
 */
export async function $onEmit(context: EmitContext<JsClientEmitterOptions>) {
  const packageName = context.options["package-name"] ?? "test-package";
  const output = (
    <Output program={context.program}>
      <ts.PackageDirectory
        name={packageName}
        version="1.0.0"
        path="."
        scripts={{ build: "tsc" }}
        devDependencies={{ "@types/node": "~18.19.75" }}
      >
        <ay.SourceDirectory path="src">
          <ts.BarrelFile export="." />
          <Client />
          <ay.SourceDirectory path="models">
            <ts.BarrelFile export="models" />
            <Models />
            <ay.SourceDirectory path="internal">
              <ModelSerializers />
            </ay.SourceDirectory>
          </ay.SourceDirectory>
          <ay.SourceDirectory path="api">
            <OperationsDirectory />
```

TypeSpec MCP扩展

- 为什么要用TypeSpec来定义MCP
  - 快速开发MCP服务，生成确定性的骨架代码
  - 更好的复用已有的API，将其暴露给MCP

TypeSpec MCP扩展

- 用TypeSpec定义和生成MCP服务
  - Library
    - typespec-mcp
      - 扩展语法
        - @mcpServer，@tool等
        - MCPError，TextResource等
    - typespec-mcp-server-js
      - 根据MCP服务定义生成JS服务骨架代码
    - typespec-mcp-server-http-js
      - 生成一个MCP JS服务代码暴露Http服务功能

```
import "typespec-mcp";

using MCP;

@mcpServer(#{ name: "My Todo app" })
namespace Todos;

/** A todo item */
model Todo {
  /** ID of the todo */
  id: int32;
  /** Text of the todo */
  text: string;
  /**
   * Status of the todo
   * - `todo` - The todo is not done
   * - `done` - The todo is done
   */
  status: "todo" | "done";
}
```

TypeSpec MCP扩展

- 如何使用
  - Library
    - typespec-mcp
      - Decorator和Type用于表示MCP服务
        - @mcpServer，@tool，@resource等
        - MCPError，TextResource等
    - typespec-mcp-server-js
      - 根据MCP服务定义生成JS服务骨架代码
    - typespec-mcp-server-http-js
      - 生成一个MCP JS服务代码暴露Http服务功能

```
   * - `done` - The todo is done
   */
  status: "todo" | "done";
}

/**
 * Add a new todo
 * @param text - The text of the todo
 */
@tool op addTodo(text: string): Todo;

/**
 * List todos.
 * Group the result by status. Todo first and do
last.
 * @param filter - filter by status. List all if
provided
 */
@tool op listTodos(filter?: "done" | "todo"): To

/**
 * Delete a todo
 * @param id - The ID of the todo
 */
@tool op deleteTodo(id: Todo.id): string;
```

TypeSpec MCP扩展

- 如何使用
  - Library
    - typespec-mcp
      - Decorator和Type用于表示MCP服务
        - @mcpServer，@tool，@resource等
        - MCPError，TextResource等
    - typespec-mcp-server-js
      - 根据MCP服务定义生成JS服务骨架代码
    - typespec-mcp-server-http-js
      - 生成一个MCP JS服务代码暴露Http服务功能

```
import "@typespec/http";
import "typespec-mcp";
import "./main.tsp";

using MCP;

/**
 * MCP server definition for NewsData.io API
 */
@@mcpServer(NewsDataioAPI, #{
  name: "NewsData.io Service",
  version: "1.0.0"
});

/**
 * Export the getLatestNews operation as an MCP to
 */
@@tool(NewsDataioAPI.getLatestNews);
```

用TypeSpec实现一个MCP工具服务

- Demo 1：用TypeSpec实现快速开发MCP Tool
- Demo 2：用TypeSpec将已有Http服务实现成MCP Tool

附录

- TypeSpec官网
  - typespec.io
- Playground
  - typespec.io
- TypeSpec MCP Repo
  - microsoft/typespec-mcp: Create an MCP Server for your API using the TypeSpec MCP Server

Microsoft

谢谢