The modern CLI-based development experience to develop, run and test UI5 applications and libraries

UI5 TOOLING & ECOSYSTEM

PETER MUESSIG, MARGOT WOLLNY
APR 01, 2020
“Development Efficiency is Key!”
The UI5 Tooling is developed as an open source project based on Node.js. It offers a modular, configurable, and extensible command line interface tailored for efficient development of applications, libraries, and reuse components with the UI5 framework.
Motivation

Increase the developer experience and thus the developer efficiency.

- Reuse of open source tooling
- Flexible and extensible
- Easy to integrate
- Enable modern language features
- Support for any IDE of choice
Architectural Overview

node.js script

Command Line

Web Browser

Local

File System

Project

Builder

Logger

Server

SAP Web IDE

SAP Business Application Studio

M Server Middleware

T Builder Task
Project Types

Types define how a project can be configured and how it is built. The UI5 Tooling distinguishes between the following types:

**Application**
Defines the main or root project. In a project’s dependency tree, there should only be one project of type application.

**Library**
Defines a project with the aim to share code across multiple projects in UI5 (e.g. for controls, reuse components).

**Theme-library**
Defines a project with theming resources for the controls of one or multiple libraries.

**Module**
Defines a non/UI5 project like a third party library. During build, it is copied without modification.
Prerequisites & Installation

# Global installation of the UI5 CLI

```bash
npm install --global @ui5/cli
```

# Verify installation

```bash
ui5 --help
```

Or other Node.js package manager that relies on `package.json` files
UI5 CLI usage

Commands

Usage: ui5 <command> [options]

Options:
  --help, -h    Show help [boolean]
  --version, -v Show version number [boolean]
  --config      Path to config file [string]
  --translator, --t manualTranslator to use. Including optional colon separator
  --verbose     Enable verbose logging. [boolean]
  --loglevel, --log-level Set the logging level {error|warn|info|verbose|sill

Examples:
ui5 <command> --manualTranslator static:/path/to/projectDependencies.yaml Execute:
ui5 <command> --config /path/to/ui5.yaml Execute comm

The CLI automatically checks for updates using update-notifier. While this is skipped in CI
environments, you might also opt-out manually by following the steps described here.

See: [UI5 CLI](https://ui5.sap.com).
Configuration

ui5.yaml

The UI5 Tooling configuration of a project is located in a YAML file named ui5.yaml.

- Must contain a project name, a project type, and a specification version.
- Allows the definition of a path mapping (in case project structure differs from expected structure), the file encoding of properties files, a copyright string, and more…
- Defines build, server, and extension configuration information.
- The build configuration can be extended with definition for custom bundling.

See: [UI5 Tooling: Configuration](#)
# Download the OpenUI5 Sample App
`git clone https://github.com/SAP/openui5-sample-app.git`

# Start the development server
`ui5 serve`

# Build the OpenUI5 Sample App
`ui5 build`
Extensibility

The capabilities of the UI5 Tooling can be extended in three ways:

- **Custom Task**: Enables the extension of the build process by a custom build task.
- **Custom Server Middleware**: Enables the plugging of custom middleware implementations into the internal express server of the UI5 server module.
- **Project Shim**: Enables the definition or extension of a project configuration of a module. (expert feature)

Extensions can be part of the configuration of a project or defined as modules to be shared across multiple projects.
"We can shape the UI5 development experience together!"
# UI5 Ecosystem Showcase

## Available Tasks and Middlewares

Below you can find the list of available tasks and middlewares. Please share your own tasks and middlewares here by extending the list of available entries here.

### Available Tasks

The following tasks (below this page) have been created and are available in the public NPM registry:

<table>
<thead>
<tr>
<th>NPM package</th>
<th>Description</th>
<th>Badge</th>
</tr>
</thead>
<tbody>
<tr>
<td>ui5-task-convert</td>
<td>template file to webcode</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
<tr>
<td>ui5-task-convert</td>
<td>bundle the entire webapp in a zip archive</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
<tr>
<td>ui5-task-rebuild</td>
<td>reify HTML/LESS/JSX files</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
</tbody>
</table>

There might be more tasks in the public NPM registry. You can find all tasks which applied the suggested naming convention here.

### Available Middlewares

The following middlewares (below this page) have been created and are available in the public NPM registry:

<table>
<thead>
<tr>
<th>NPM package</th>
<th>Description</th>
<th>Badge</th>
</tr>
</thead>
<tbody>
<tr>
<td>ui5-middleware-client-dev</td>
<td>use the application as proxy</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
<tr>
<td>ui5-middleware-client-dev</td>
<td>sync UI transactions across connected browsers (signal.conf)</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
<tr>
<td>ui5-middleware-client-dev</td>
<td>usage of full data for development</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
<tr>
<td>ui5-middleware-compiler</td>
<td>on-demand build to build template when requesting</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
<tr>
<td>ui5-middleware-sysproxy</td>
<td>simple express proxy</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
<tr>
<td>ui5-middleware-serve</td>
<td>serve static resources</td>
<td><img src="download" alt="Download" /> <img src="license" alt="License" /></td>
</tr>
</tbody>
</table>

There might be more middlewares in the public NPM registry. You can find all middlewares which applied the suggested naming convention here.
UI5 Tooling – a modern CLI-based development experience!

Just a few steps ahead:
- Create a UI5 project in SAP Web IDE
- Download and enable the project to run locally
- Include custom tasks and middlewares
- Feel the difference of the development experience

# Start the development server
ui5 serve

Server started
URL: http://localhost:8080
UI5 Tooling Family & Friends

- Yeoman
- Karma
- TypeScript
- Terser
-parcel
- expressjs
- {less}
- Babel
- JS Doc
- Migration
- TS
- S
OpenUI5

> 2.0

npm

SAPUI5

© 2020 SAP SE or an SAP affiliate company. All rights reserved. | PUBLIC
The UI5 Tooling 2.0 brings several enhancements:

- Advanced **frontend package manager** to avoid version inconsistencies with a shared local dependency cache (not per project, like npm)
- UI5 **framework dependencies** are managed in the `ui5.yaml` and can be OpenUI5 or SAPUI5:
  - Specify one version for all UI5 libraries of the framework
  - Declare the required UI5 libraries
- New CLI commands:
  - `ui5 use`: select a framework and the framework version
  - `ui5 add`: add one or more library dependencies
- Introduces validation of the `ui5.yaml` file to improve the error detection
- **Properties files** encoding now **defaults to UTF-8**

* New UI5 Tooling 2.0 features require `specVersion: '2.0'` in `ui5.yaml`!

---

```yaml
# UI5 dependencies
framework:
  name: SAPUI5
  version: 1.76.0
libraries:
  - name: sap.ui.core
  - name: sap.m
  - name: themelib_sap_fiori_3
  - name: sap.suite.ui.microchart

~
```

---

See: [UI5 Tooling 2.0](#)
What else

- Project creation via a primitive SAPUI5 Yeoman generator
- Provides 3 templates: Worklist, Master Detail, Worklist OData V4

```bash
# Install the SAPUI5 Yeoman generator
npm install --global @sapui5/generator-sapui5-templates

# Run the generator (select the template from list in question step)
yo @sapui5/sapui5-templates
```

- Continuously deliver the TypeScript definitions for OpenUI5/SAPUI5
- Enables basic code-completion for UI5 development

```bash
# Install the DTS files for OpenUI5 (available for 1.60)
npm install @openui5/ts-types

# Install the DTS files for SAPUI5 (will be available for 1.71, >1.75)
npm install @sapui5/ts-types
```
What’s next

- Enable a modern development experience for UI5 by using a Babel toolchain (focusing on app development)
- Open-source solutions already exist, it is all about using them properly (*)
- Using TypeScript to benefit from code-completion and extended checks in any editor
- Utilize Babel to transpile ES6+ features back to UI5 APIs (such as import/export to sap.ui.define/require)

(* Independent from UI5 Tooling (integrated via extensibility)

**DISCLAIMER:** no warranties / no timelines yet!
Further resources

- [UI5 Tooling](https://github.com/SAP/ui5-tooling) Project on GitHub

- [UI5 Tooling Documentation](https://github.com/SAP/ui5-tooling)

- [UI5 Tooling Feedback/Issues](https://github.com/SAP/ui5-tooling)

- [Introducing the UI5 Tooling for SAPUI5 projects](https://simplenews.sap.com/2019/09/introducing-the-ui5-tooling-for-sapui5-projects) Blog


- [Utilize UI5 tooling's extension capabilities for an improved development experience](https://simplenews.sap.com/2019/09/utilize-ui5-toolings-extension-capabilities-for-an-improved-development-experience) Blog


- [UI5 Ecosystem Showcase](https://github.com/SAP/ui5-tooling) on GitHub
THANK YOU

Margot Wollny
Product Expert, SAPUI5
margot.wollny@sap.com
SAP Community: @margot.wollny

Peter Muessig
Chief Architect, SAPUI5
peter.mueissig@sap.com
Twitter: @pmuessig
SAP Community: @peter.mueissig