Software Requirements Specification

Tweezers

26/12/18

Nissan Cohen, Alon Boxiner

Contents

[Introduction 3](#_Toc533624016)

[Scope 3](#_Toc533624017)

[Glossary 3](#_Toc533624018)

[Overall Description 3](#_Toc533624019)

[Product Features 3](#_Toc533624020)

[Tweezers.Core 3](#_Toc533624021)

[Tweezers.UI 4](#_Toc533624022)

[User Characteristics 4](#_Toc533624023)

[System Architecture 4](#_Toc533624024)

[Software Requirements 5](#_Toc533624025)

[Software General Requirements 5](#_Toc533624026)

[Tweezers.Core 5](#_Toc533624027)

[Tweezers.UI 5](#_Toc533624028)

[Internal Requirements 5](#_Toc533624029)

[Tweezers.Core 5](#_Toc533624030)

[Tweezers.UI 5](#_Toc533624031)

[Database 5](#_Toc533624032)

# Introduction

## Scope

The Tweezers project is aiming to change the way developers create and maintain the Frontend and Backend of their form-based web applications.  
Tweezers is a unified platform to connect organization’s databases into modular components that can generate both CRUD Restful server or Frontend with minimal development skillset.

With Tweezers you can generate your Frontend and Backend components according to your needs.

## Glossary

|  |  |
| --- | --- |
| **FE** | Frontend. |
| **BE** | Backend. |
| **Tweezers.Core** | Tweezers discovery helper component. |
| **Tweezers.UI** | Tweezers Frontend generator. |
| **API** | Application Programming Interface. |
| **CRUD** | [Create, Read, Update and Delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete). A generic term for the four basic functions of persistent storage. |
| **RESTful** | Representational state transfer. A popular software architectural style in designing and creating web services. |
| **Reflection** | Provides objects that describe assemblies, modules and types in C#. |
| **SPA** | Single Page Application |

# Overall Description

## Product Features

The Tweezers project consists of 2 main components. Each is responsible to deliver a different part of the overall generated web application. We encapsulated the following components to make them isolated and swappable by the developer.

### Tweezers.Core

This component is responsible to generate API and metadata that helps the developer in building his web server. This component is built using the .NetCore framework.

A developer needs to reference Tweezers.Core in his backend .NetCore project. The Tweezers.Core component is scanning the project’s classes using Reflection and generates both the needed metadata and resources for Tweezers.UI and potentially the desired RESTful API endpoints.

Tweezers.Core is using Reflection and instrumentation to both show the developer the injection points he can use to fill his own logic and to fill in the gaps in which the developer decided to let Tweezers do the work for him.

The project is maintaining an active [WIKI page](https://github.com/tweezersCi/Tweezers.Core/wiki) that contains all the main components and their explanations. Including installation[[1]](#footnote-1) steps and sample code.

Tweezers.Core is built with every push to Git using a custom CI script using [AppVeyor](https://ci.appveyor.com/project/tweezersCi/tweezers-core/branch/master). The latest binaries can be fetched from there.

During the build process, Tweezers.Core code will be statically analyzed with the SonarCloud static code analysis tool.

### Tweezers.UI

This component is responsible in generating the UI needed for the form-based web applications that are either using Tweezers.Core or implementing needed interfaces in the Tweezers standard.

Tweezers.UI exploits the implemented Tweezers discovery mechanism to generate the SPA.

Tweezers.UI uses specialized “discover” endpoints that come from the BE that contains the needed metadata that in turn will be translated to the UI.  
There’s a single endpoint that Tweezers.UI calls to that forks into the entire discoverable server-side entities.   
Each entity will expose a route that returns the [Discoverable Metadata](https://github.com/tweezersCi/Tweezers.Core/wiki/DiscoverableMetadata) needed to generate the page.  
Once Tweezers.UI has all the information needed to generate the pages, it will start loading the application.

Tweezers.UI will maintain an active [WIKI page](https://github.com/tweezersCi/Tweezers.UI/wiki) that will contain all the main components, their explanations and installation instructions.

Tweezers.UI will be written in [Vue.js](https://vuejs.org/). We aim to start writing Tweezers.UI once the Core component is mature enough to serve it.

## User Characteristics

The application is intended for developers with at least a basic knowledge in C#. Intermediate developers can use Tweezers’ advanced capabilities to enhance their projects.

Tweezers demands the developer to implement either a login component.

## System Architecture

Can be found in the [Tweezers Flow](https://github.com/tweezersCi/Tweezers.Core/blob/master/Docs/Tweezers%20Flow.png) chart.

# Software Requirements

## Software General Requirements

The following is a list of capabilities that Tweezers is expected to support

### Tweezers.Core

* Find discoverable classes.
* Create the needed metadata for Tweezers.UI.
* Implement a basic CRUD RESTful API, with the option to add the following actions:
  + Create entities.
  + Read entities (single and many).
  + Update entities.
  + Delete entities.
* Enforce basic RBAC.

### Tweezers.UI

* Get discoverable entities and metadata.
* Generate the following pages:
  + Login page.
  + Main menu component with all discoverable entities.
  + Grid item for each entity.
  + Single item page for each entity.
* Support the following UI components in the forms:
  + TextBox – String, Int.
  + Dropdown – Enums.
  + Slider – Boolean.
  + Mini entity window – Nested discoverable entities.
  + [Tagbox](https://jbt.github.io/tagbox/) (or equivalent) – for numeric/string arrays.

## Internal Requirements

### Tweezers.Core

Tweezers.Core will be written in the latest version of .NetCore (currently 2.1).

### Tweezers.UI

Tweezers.UI will be written in Vue.js.

### Database

Tweezers will implement two basic database connectors for PostgresQL 10 and MongoDB 4.  
The connectors will be written in .NetCore.

1. Under construction [↑](#footnote-ref-1)