Small Application for inserting and consulting trouble tickets (web + REST service with a DB)

In this demo there is a web site, which is the client for a Trouble Ticket service. The web application allows an employee to submit a problem as a trouble ticket, and also to consult previous submissions.

The web application uses a REST WCFService to do most of its operations (in this case the creation of a new trouble ticket and the retrieval of previous trouble tickets submitted by an employee, as well as getting the list of employees). For these simple operations the service simply uses a database. The service is hosted separately from the web site, and is available for the web application and other external applications.

The VS 2019 solution (TTs) of this demo has three projects. One of them (TTService) implements the REST WCFService as a library. Another is the host of the web service as a console application (TTServer), and finnaly the WebSite (TTSite) with the web application (an ASP.NET form). The site has a reference to the other project (TTService) that puts automatically the service .dll in the Bin subdirectory. In order to invoke the service operations from the web application code we have written manually a simple proxy to the service, inside the TTSite code.

The TTService project was created from the template 'WCF Service Library' and the interface and implementation written according to the pretended mapping to HTTP requests.

The TTServer project is a simple .NET console application that uses the class WebServiceHost as a REST web server. The app.config file should be configured accordingly. Also because of some security measures of Windows, to run a server on the HTTP protocol we must run it from VS or a console window in administrator mode (or register the application permantely as an administrator).

The TTSite is an ASP.NET web forms application. I started with an 'ASP.NET Empty Web Site' template and then added a new 'Web Form' item. A Web form has two files: one with the extension .aspx where you could write HTML and CSS and add visually interface controls from the file design window in VS and the toolbox (at left); there is also a code file (code behind) where you can write c# code mainly to be executed when there are 'postbacks' in consequence of user interactions with the page. The code file can contain initializations, handlers to events (e.g. button clicks), and other auxiliary code. It is executed on the server side and can modify the page interface is response to 'postbacks'.

In this demo there is a database as a SQL Server LocalDB in the TTServer project, but can be any other. This one is free and has full support from VS 2019. It only needs to be installed in the same system as VS for development and in the server system for deployment. This kind of DB Manager uses a file for each DB (in this case the TTs.mdf file), which can be put in any place in the file system, facilitating its deployment. But it is also a server system in the sense that the DB engine runs on a separated process. Nevertheless that process is started automatically when its client starts (or makes the first connection) as a client's child process. This can have some advantages in performance (and also some disadvantages) over a simple embedded DB.

The supplied DB has already some data and two tables. One is for the employees, whose data is displayed in a combo box (in the ASP.NET web page). The other has the Trouble Tickets information and is managed by the REST WCFService.

The VS Server Explorer can be used to create, visualize and edit the database file, inside a project. The app.config file contains the string (except the directory) to connect in run time to the DB (the connection string section). The constructor of the service builds the full connection string.