How to use Lasernet as a Web Service

- 2022-11-04 - Comments (0) - Lasernet Developer FAQs

Lasernet can be configured both as a client for a web service or a web service server. The purpose of this article is to demonstrate how Lasernet can be configured as a custom web service server. When setting up Lasernet as a web service, it can be thought of as defining an API for the configuration.

The API is what is exposed to clients via a runtime generated WSDL file (Web Services Description Language).

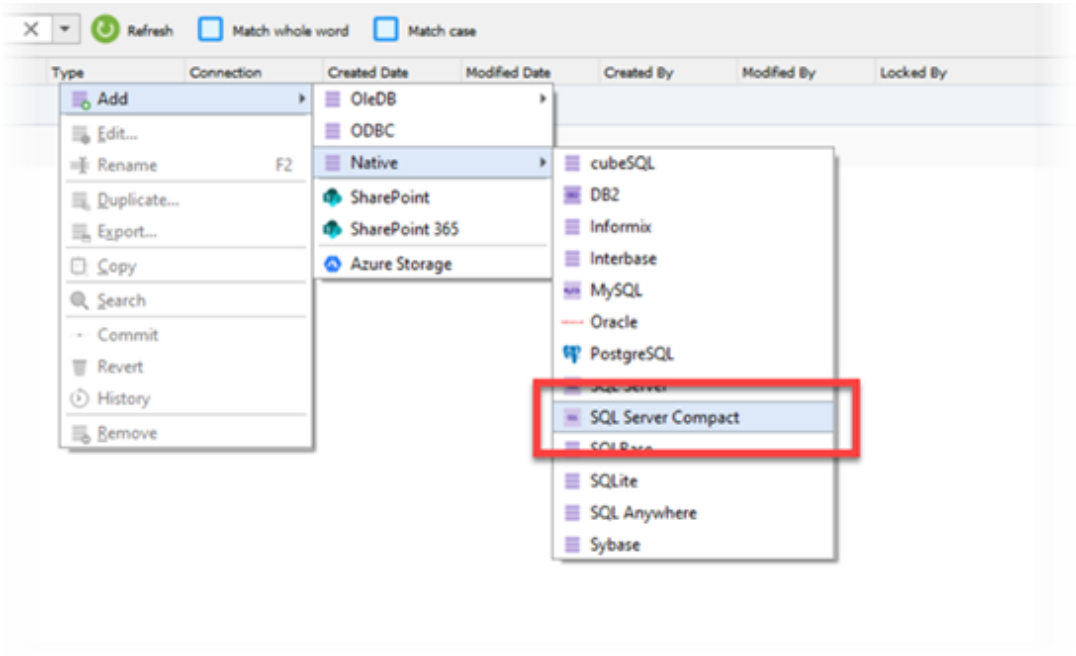
This example shows a range of the features of the Web Service server feature, however, the example does not cover all features but demonstrates a good starting point.

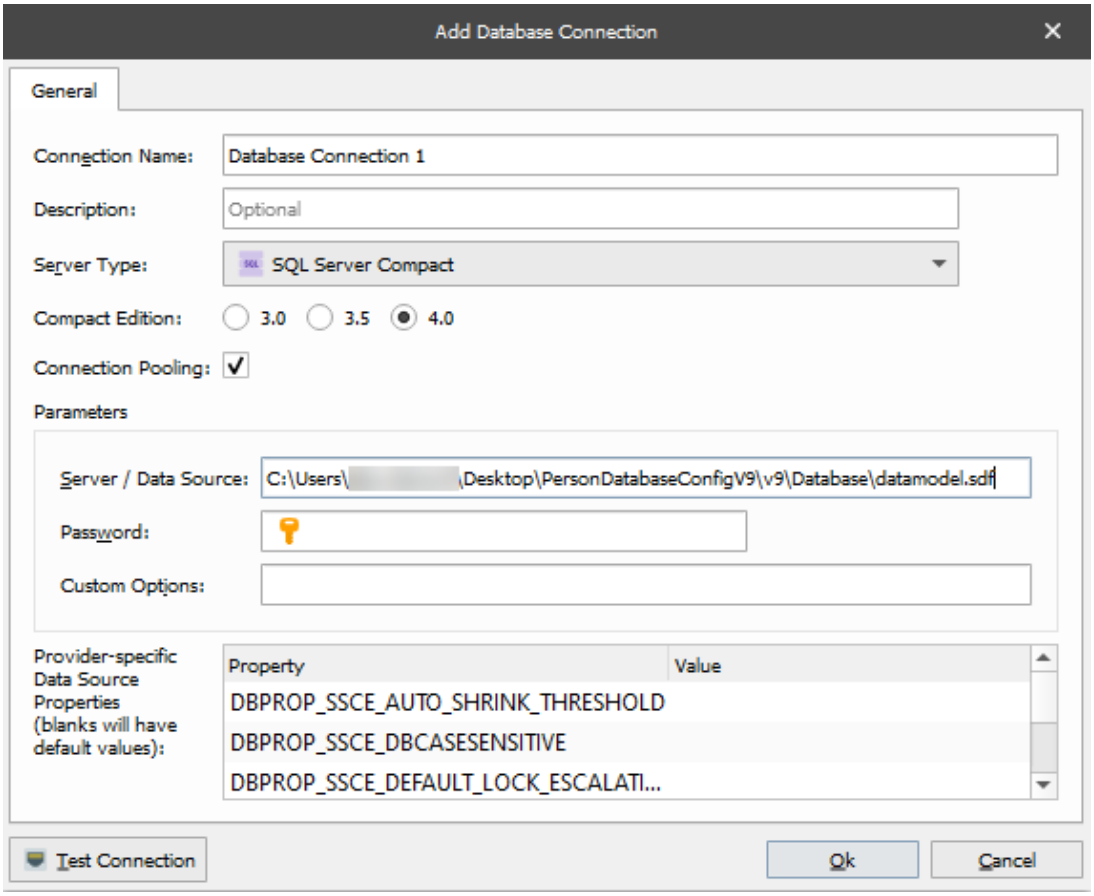
Data model

To have regular data we need a database. Since this is a simple example, our data model will contain just one table. Follow these steps:

1. With Lasernet Developer open, click the **Commands** option, right-click **Connections** in the workspace and click **Add**.

2. In the drop-down menu, select **Native** and then choose **SQL Server Compact** (included with the Lasernet package).





3. Click the **Test Connection** button and an empty database will be created in the specified folder if it exists.

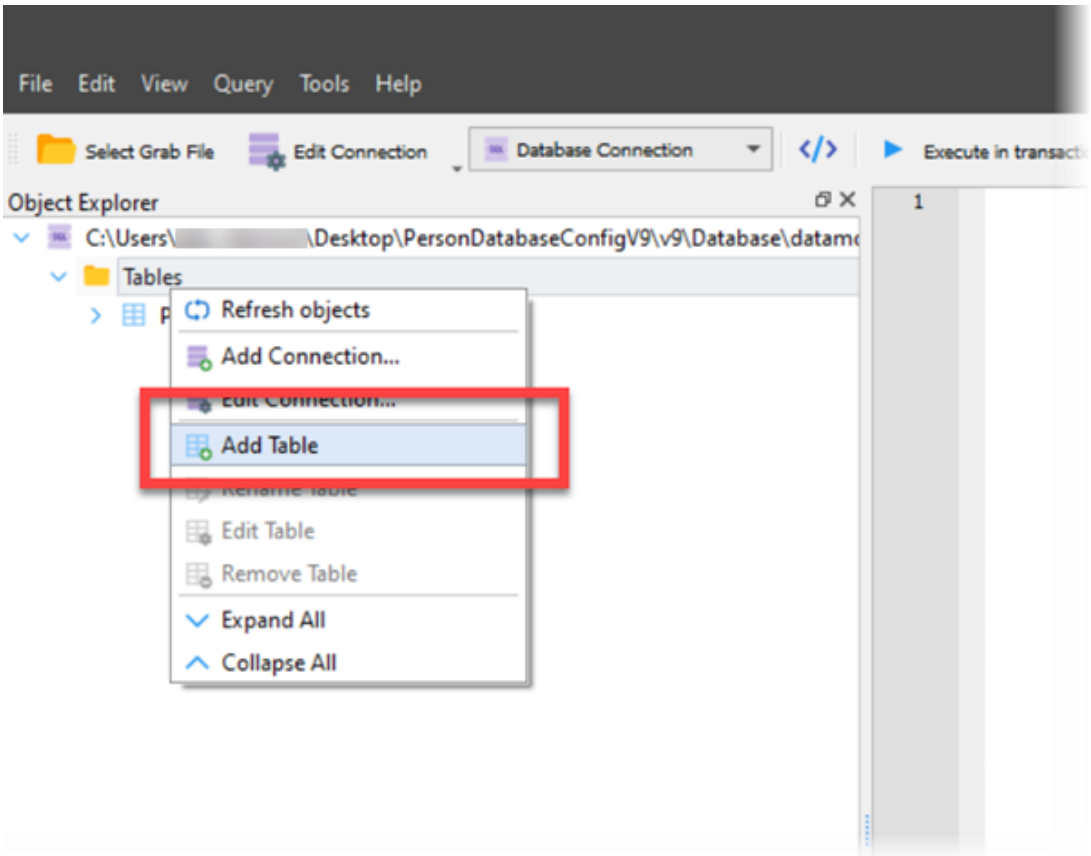
We need to create database commands to read from, write to and delete from our

database, but first, we need to define our Person table. This can be done via the built-in table editor.

4. Right-click **Commands** on the Workspace, select **Add** and then choose **Command Text** from the drop-down menu.

5. Ensure **Database Connection** is selected in the Toolbar and create a database command for the first command ‘PersonAdd’,

6. Choose our previously created database connection and in the Object Explorer panel, right-click and select **Add Table**.



7. Give the table a name and add columns to it.

We have added normal metadata for a person and a primary key which will automatically be increased when a new person is added to the database (identify flag).

The server

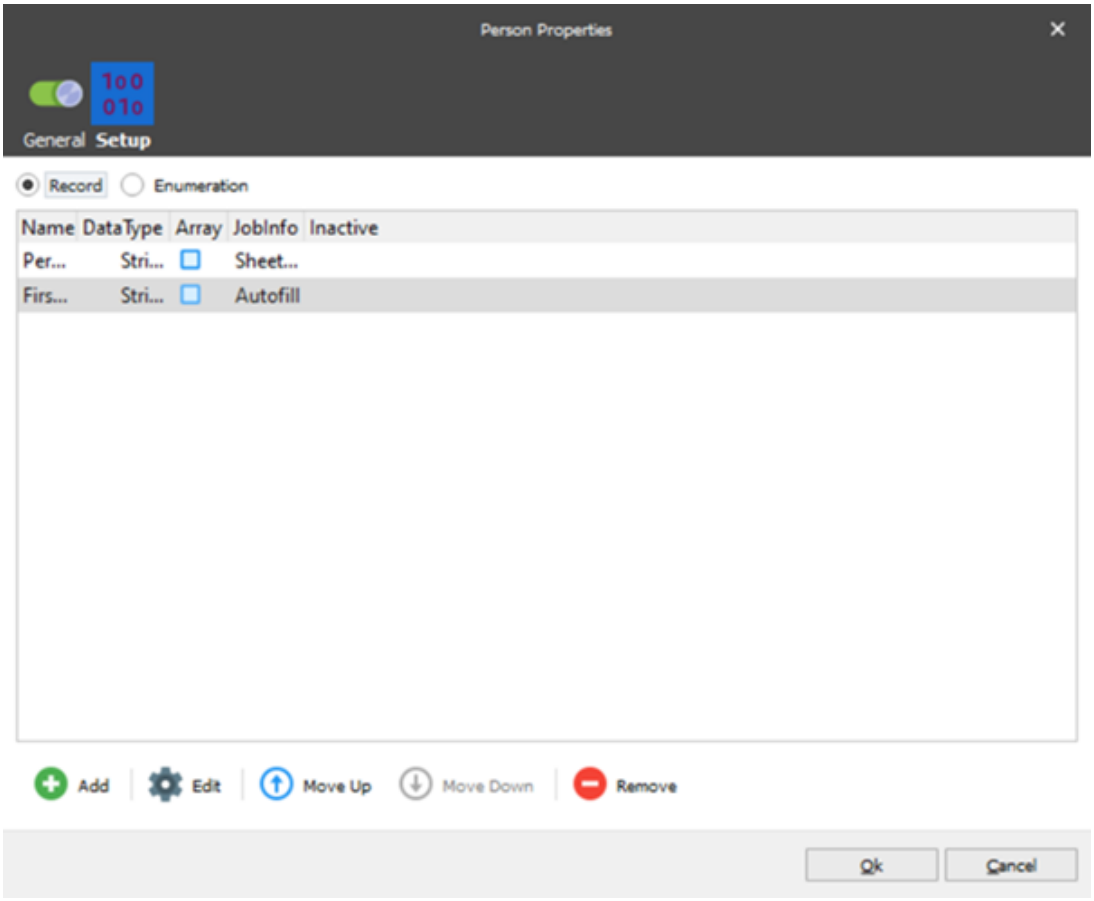
Our example will be an API encapsulating a person database. We will need a database and a web server input port.

Add a Web Server module to a configuration and give it a meaningful name. In this example, we have used ‘PersonDatabase’.

When updating configuration, the Lasernet web server (Microsoft IIS) will be booted and an URL will be added. A log a message will be shown when service starts;

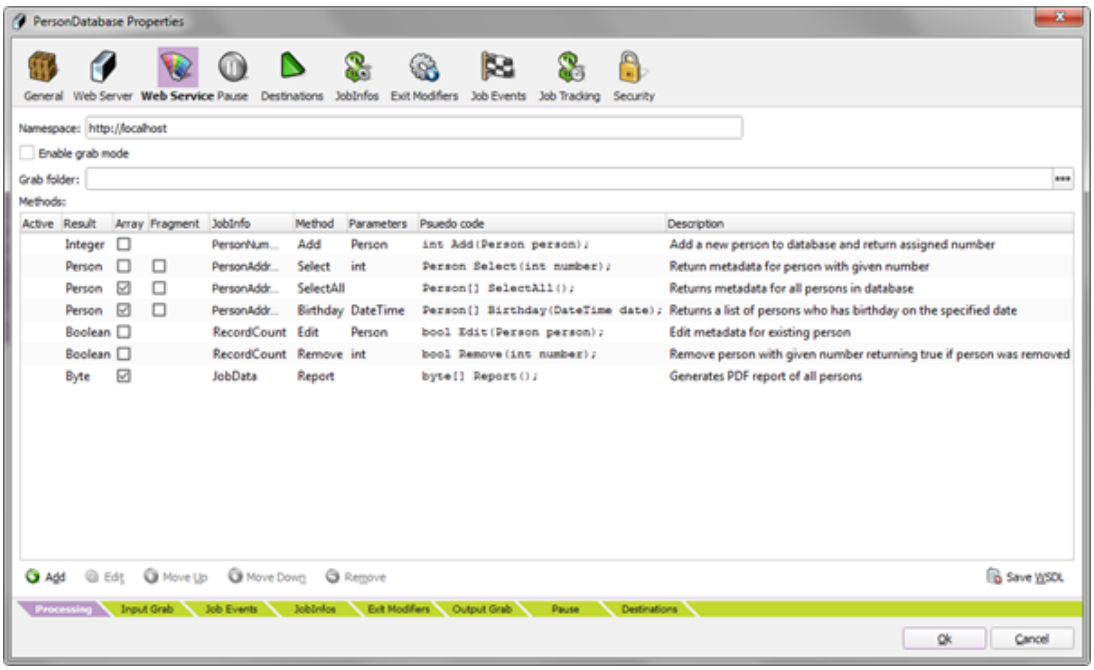
URL added: http://+:8080/webinputport/PersonDatabase/webservice/

This is the URL where the web service can be reached. The web service will only be booted if at least one method has been defined.
Defining the APITo be able to communicate with our database we need to add some methods to our web service. Methods such as *Add, Edit, Remove* which directly manipulate the data model. But also methods like *Report* which works like a preview function generating a PDF report of the content of the data model.



Methods

Methods are added to the Web Service tab of the Web Server input port. The methods in this example can be seen in the following screenshot and investigated more in detail in the attached configuration.



Response

The Lasernet web service is blocking. This means when a client calls a method, he is blocked until Lasernet responds. How and when Lasernet responds is up to the developer of the configuration.

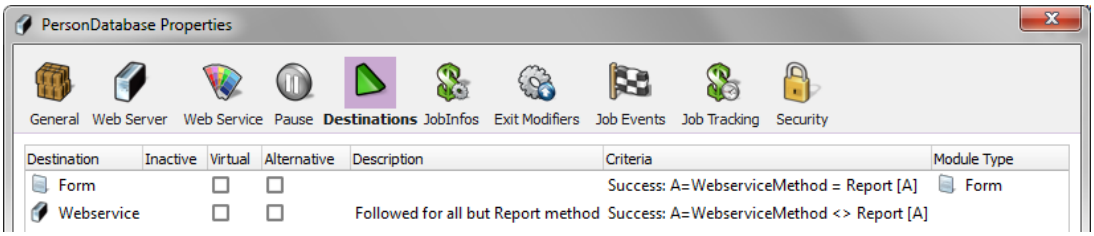
Lasernet responds by passing the Job to a special destination called ‘Webservice’. This Destination is only visible in the GUI when a Web Server has been added to the configuration.

The Job passed to ‘Webservice’ destination is used as a base for the response.

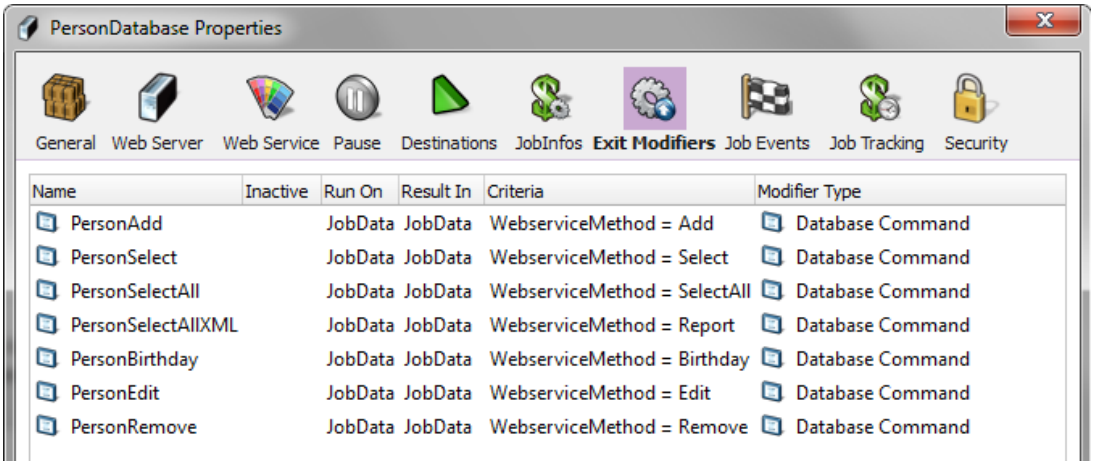
Logic behind calls to methods

When a client calls a method, a request is received in Lasernet and a Job is generated. Method parameters will be mapped into the specified JobInfos in the Job.

A JobInfo called ‘WebserviceMethod’ is set with the method name. This can be used for filtering where the Job is passed (see Report method in the example) depending on which modifier or script is called.

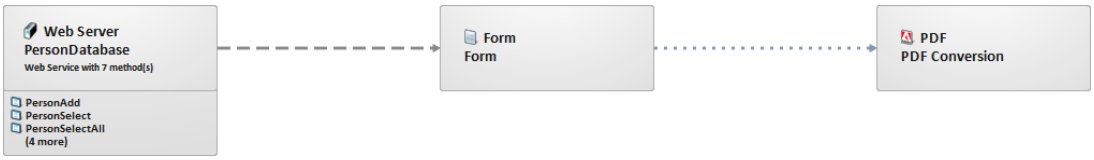


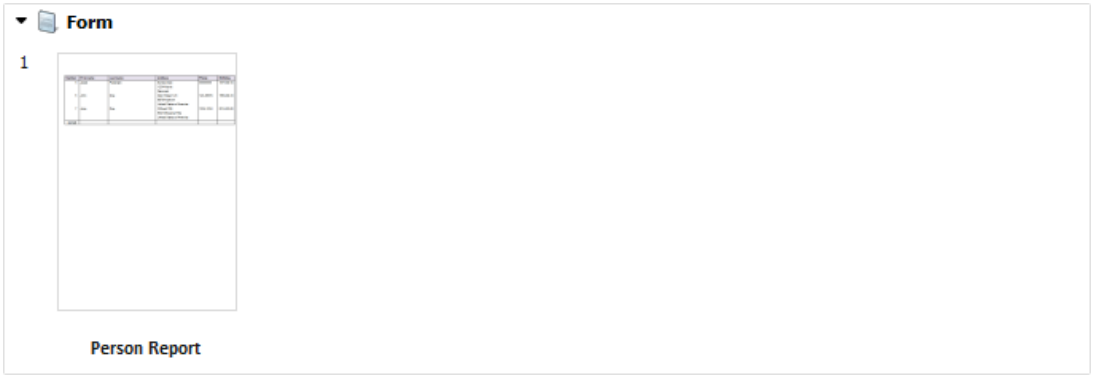
For the Report method, the Job is passed to the Form Engine. For all other methods, the Job is passed to ‘Webservice’ and Lasernet responds after running Exit modifiers.

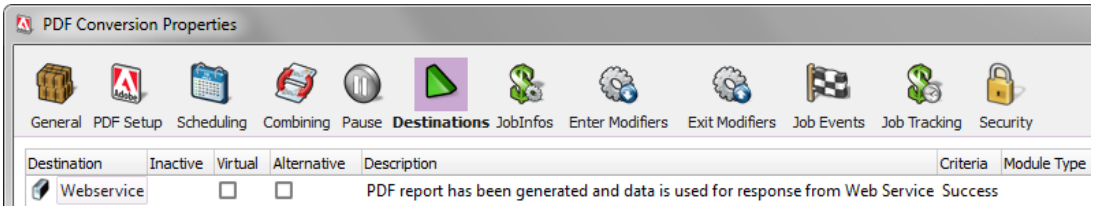


All methods except Report is mapped to a specific Database Command which uses the parameters passed to the method to read from and write to database. Database reads will set JobInfos which are used to build the responses from the methods.

After the PDF report has been generated, the Job is passed to ‘Webservice’ and JobData is returned as an array of bytes.

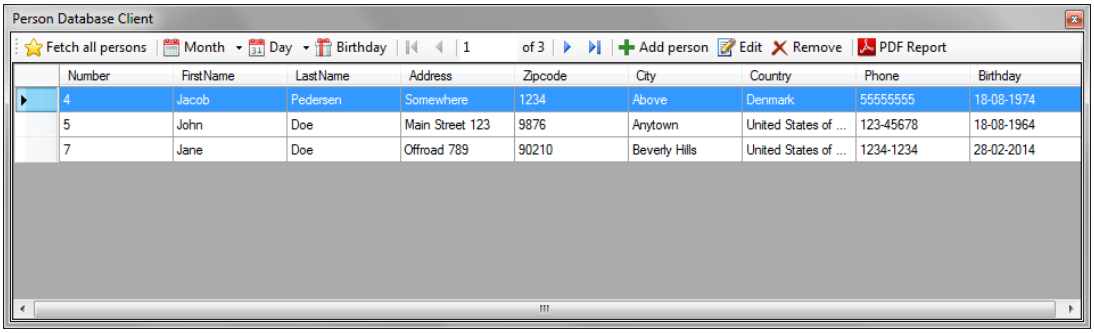






A Client for the Web Service

A sample project is attached for testing the Person database.



Different methods in the Web Service can be tested, described below:

Fetch all persons: Calls SelectAll which returns all Persons in the database.

Birthday: Set a given day of the year via Month and Day buttons. Only persons who have their birthday on this given day are fetched. The Birthday method is called.

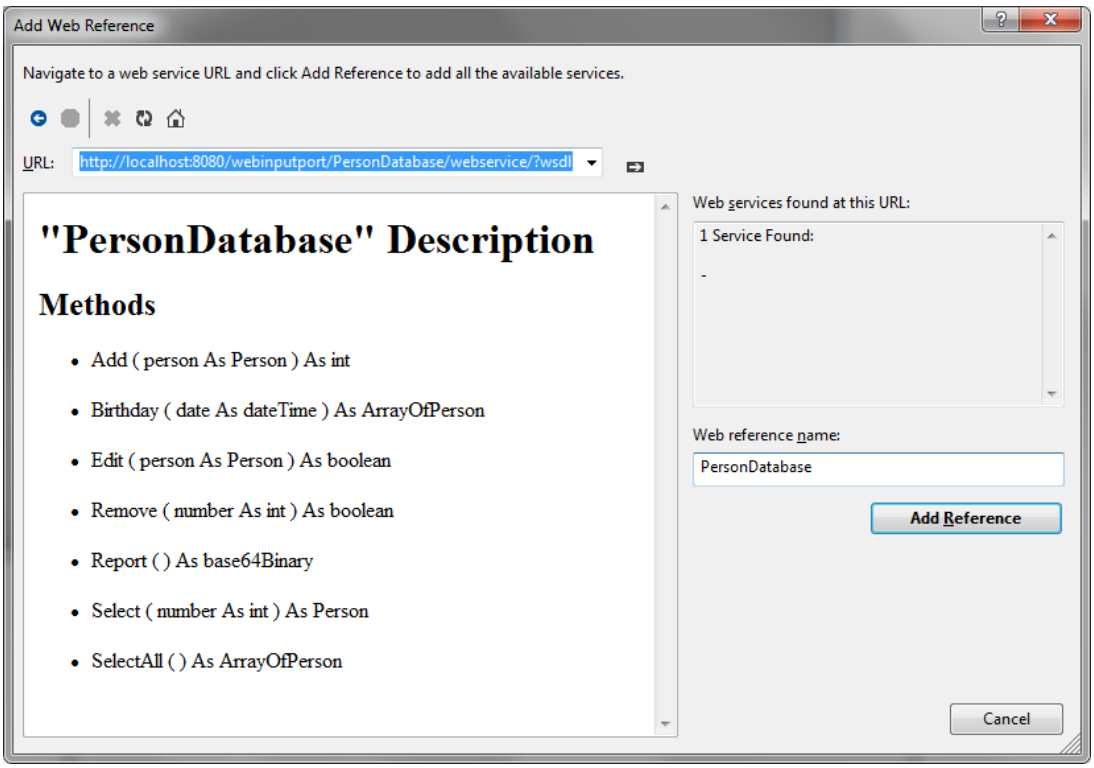
Add: Function to add persons to the database. Add method is called.

Edit: Function to edit the selected person. This function calls the Select method to get the last metadata for the Person. Shows the dialog and saves the changes via the Edit method.

Remove: Function to remove the selected person. Remove method is called.

PDF Report: Function to show a report of all persons in the database. A PDF will be generated by Lasernet and shown in the associated PDF viewer. The Report method is called.

A sample project was made by adding a web reference to the WSDL file (the screenshot is an example from Visual Studio);



This will auto-generate code to communicate with the web service. Afterward, set the URL (and network credentials if used);

var pd = new PersonDatabase.PersonDatabase();

pd.Url = @"http://localhost:8080/webinputport/PersonDatabase/webservice/";

Then the seven methods can be called.

Sample projects

There are two zip files attached which can be downloaded and used as a reference and for testing purposes.

PersonDatabaseConfigV7.zip and PersonDatabaseConfigV

The zip file contains a demo Lasernet configuration which can be opened in Lasernet Developer 7 or 9. Follow these steps:

1. Unzip the configuration to any folder on your computer and import it in Lasernet Developer.

2. Browse the configuration to view the settings described in this article.

3. Open the Database Connection included in the configuration.

4. Change the folder for the Data Source to the root directory of the configuration plus database name. The name of the included Microsoft SQL Server Compact database file is "datamodel.sdf".

5. Change the server/instance name in the configuration to your server/instance running

the Lasernet Service.

6. Update/Commit and Deploy the configuration to the server.

PersonDatabaseClient.zip

The zip file contains a .NET sample project with source code for testing the Person database.

1. Unzip the sample project to any folder on the computer running the Lasernet Service with the uploaded configuration.

2. Run the PersonDatabaseClient.exe in the *\PersonDatabaseClient\PersonDatabaseClient\obj\Debug* folder.

3. Click the buttons in the Person Database Client to test the different methods described in this article.

4. Start Lasernet Monitor to view log messages for the various methods.

Modules required:

For testing purposes, a valid license is required for the following applications and modules:

- Lasernet Developer 7 or 9
- Lasernet Server 7 or 9
- Database Module
- Conversion Module
- Form Engine (XML In)

PersonDatabaseConfigV7.zip

PersonDatabaseConfigV9.zip

PersonDatabaseClient.zip

Related Content

- How to run Lasernet in debug mode