Consuming RESTful Services using jQuery and Ajax

Overview

In this lab, you'll implement a Web application that allows users to make suggestions for products that a company should sell. Users will be able to enter a description of the product and a recommended retail price. Users will also be able to update and delete product suggestions, as well as query existing product suggestions.

Here's a quick overview of the server-side and client-side parts of the Web application:

- The server-side part is already completely implemented. This Web service defines URLs that represent CRUD actions for product suggestions – i.e. it has URLs that a client can ping to create, read, update, and delete product suggestions.
- The client-side part is an HTML page that includes JavaScript code to interact with the server. Some of the basic JavaScript code is already complete. You will add some additional code to consume the CRUD operations provided by the server.

Roadmap

There are 6 exercises in this lab, of which the last exercise is "if time permits". Here's a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

1. Familiarization with the solution
2. Familiarization with the client-side code
3. Inserting a product
4. Updating a product
5. Deleting a product
6. Additional suggestions

Exercise 1: Familiarization with the solution

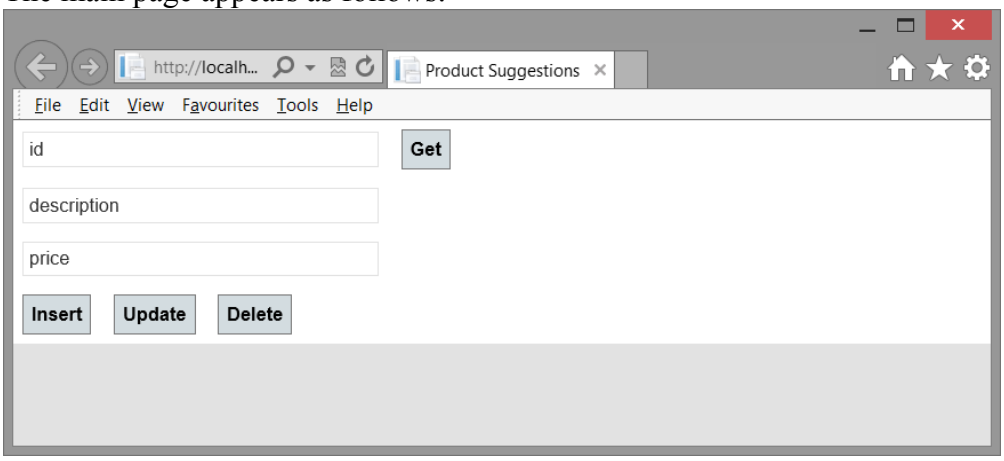
Start your editor and open the *Solution* project for this lab, to get an idea of what you're aiming for. Here are the details of the project:

- *Folder:*
  `\solution\product-suggestions`

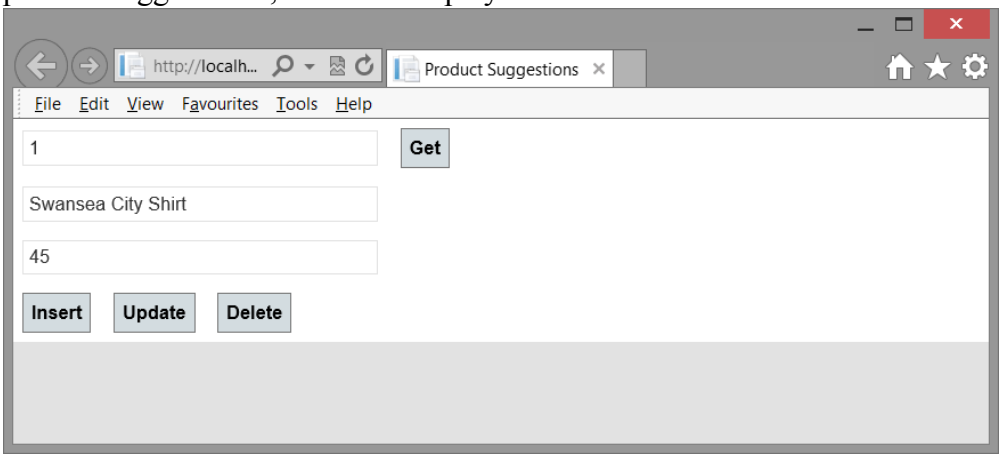
Build and run the application:

- Open command line in folder `\solution\product-suggestions`
- To install the dependencies, type: "npm install"
- To run the project, type: "npm start"
- Open the browser at the location that the server wrote on the command line (http://localhost:3000)

The main page appears as follows:



Type a number between 1 and 8 in the id text box, and then click the Get button. This causes the Web page to send an HTTP GET request to the server, to get a product suggestion object from the server (the server holds 8 hard-coded product suggestion objects in memory initially). The web page displays the gotten product suggestion in the text boxes – for example, if you get product suggestion 1, it will be displayed as follows:



Modify the description and/or price and then click the *Update* button (you can't modify the id, because it's like a primary key). The Web page sends an HTTP *PUT* request to the server, to update the existing product suggestion. All being well, the web page will display a confirmation dialog box indicating the item has been updated successfully.

Now enter a description and price for a new product suggestion, and then click the *Insert* button (you don't specify the id, because it will be generated by the server). The Web page sends an HTTP *POST* request to the server, to insert a new product suggestion. All being well, the server will generate an id for the new product suggestion object, and then return the object back to the web page – the web page should display a dialog box indicating the id of the new item.

Finally, enter an id of an existing product suggestion, and then click the *Delete* button. The Web page sends an HTTP *DELETE* request to the server, to delete the product suggestion. All being well, the web page will display a dialog box indicating the item has been deleted successfully.

Close the browser window and return to your editor.

Exercise 2: Familiarization with the client-side code

In your editor, open the *Start* project for this lab as follows:

- *Folder:*
  `\start\product-suggestions`

Build the application:

- Open command line in folder `\start\product-suggestions`
- To install the dependencies, type: “npm install”

If you want to run the project later in the other exercises:

- To run the project, type: “npm start”
- Open the browser at the location that the server wrote on the command line (http://localhost:3000)

If you want to stop running the project later in the other exercises:

- Press ctrl+c and answer with Y(es).

Expand the `public` folder and take a look at `index.html`. This page contains the mark-up for the web page – there are no great surprises here! All the JavaScript code is tucked away in a separate JavaScript file named `ProductSuggestionsManage.js`, so open this file and take a look at the code. Here's a quick summary of the functions in the file:

- `getProductSuggestion()`
  This function is invoked when the user clicks the *Get* button on the web page. This function is already complete, to get the ball rolling. The function uses jQuery to send an Ajax *GET* request to a URL such as `/productsuggestions/1`, where the final part of the path is the product id that the user entered in the web page. The server will return a `productSuggestion` object in JSON format. The "success" function receives this object as its `data` parameter, and passes the object into the `displayProductSuggesion()` helper function (more on this function in a moment).
- `insertProductSuggestion()`
  This function is invoked when the user clicks the *Insert* button on the web page. You will complete this function in Exercise 3, to send an Ajax *POST* request to insert a new product suggestion.
- `updateProductSuggestion()`
  This function is invoked when the user clicks the *Update* button on the web page. You will complete this function in Exercise 4, to send an Ajax *PUT* request to update an existing product suggestion.
- `deleteProductSuggestion()`
  This function is invoked when the user clicks the *Delete* button on the web page. You will complete this function in Exercise 5, to send an Ajax *DELETE* request to delete an existing product suggestion.

- `clearProductSuggestion()`
  This is a helper function, to clear all the text boxes on the web page.
- `displayProductSuggestion()`
  This is a helper function, to display a product suggestion object in the text boxes.
- `displayError()`
  This is a helper function, to display an Ajax error message in a popup dialog box.

Exercise 3: Inserting a product

In `ProductSuggestionsManage.js`, locate the `insertProductSuggestion()` function.

This function has some existing code to create a JavaScript object named `productSuggestion`, based on the values entered in text boxes by the user. This JavaScript object is a client-side equivalent of the `ProductSuggestion` class that we saw earlier at the server – i.e. it has properties named `Description` and `Price` (the `Id` property will be returned by the server after successful insertion).

Complete the function as indicated by the TODO comments, to send an Ajax *POST* request to insert the product suggestion at the server. Note that your Ajax call will need to set the `data` property to a stringified version of the `productSuggestion` object as follows:

```
data: JSON.stringify(productSuggestion)
```

When you're ready, build and run the project. In the web page, try to insert a new product suggestion. If you get any problems, press F12 in the browser window and debug your client code to see what's wrong.

Exercise 4: Updating a product

Back in Visual Studio, locate the `updateProductSuggestion()` function. Complete this function as indicated by the TODO comments, to send an Ajax *PUT* request to update an existing product suggestion at the server.

Build and run the project. In the web page, first get an existing product, then change its description and/or price and click *Update*. Verify that the product is updated successfully.

Exercise 5: Deleting a product

In Visual Studio, locate the `deleteProductSuggestion()` function. Complete this function as indicated by the TODO comments, to send an Ajax *DELETE* request to delete an existing product suggestion at the server.

Build and run the project. In the web page, enter the id of an existing product suggestion, and then click *Delete*. Verify that the product is deleted successfully.

Exercise 6 (if time permits): Additional suggestions

Add new functionality to the web page, to get all product suggestions from the server. Display the collection of items in an HTML table.