Hemran Akhtari

Mrs. Winnie van Schilt & Mr. Rob Smit

Data Processing

17 January 2020

Documentation

**1. Introduction**

Starting the project, I picked the World database from [dev.mysql.com](http://dev.mysql.com) as my dataset with MySQL (using XAMPP) as my database language.

My API language of choice is node.JS and for the visualisation part I used C# Windows Form Application. My goal was to create an API with node.JS that accesses the MySQL database to provide a dataset in JSON and XML to my C# Application without connecting directly to the MySQL database which gives you serious benefits such as high security and low latency applications.

**2. API (Node.JS)**

**2.1 Prerequisites**

The first step for creating the API is to download Node.JS from [nodejs.org](http://nodejs.org). After successful installation you need to install the following packages with the “npm install” command in your console of choice (I used Git Bash):

* Express
* Mysql
* Express-xml-bodyparser

**2.2 Execution**

The next step is to import those installed packages with the “require” method in the index.js. After initiating the packages I defined routes to my JSON / XML node.JS files and defined a port the node.JS server is running. Furthermore, I created a connection to my database.

After successfully creating the index.js file, I went on creating the route files which in my case are country.js and countryXML.js. Inside those files I created routes for either (C)reating, (R)eading, (U)pdating or (D)eleting rows in the database. I fetched the data form the database and send it to the specified route. For the XML part I parsed the json to XML with the “express-xml-bodyparser”.

**3. Visualisation (C#)**

**3.1 Prerequisites**

The first step for creating the Windows Form Application is to import all important namespaces:

using Newtonsoft.Json.Linq;

using System;

using System.Linq;

using System.Windows.Forms;

using System.Xml;

using System.Xml.Linq;

using System.Xml.Schema;

using Newtonsoft.Json.Schema;

using Newtonsoft.Json;

using System.IO;

using System.Collections;

**3.2 Execution**

Then I created the Windows Form with multiple tabs for general information about the country, population counts per city, language distribution and a comparison table for population. The webClient library helped me to grab all the data from the API. Taking this data I created an Entity Framework and mapped to to the classes I created.

The user can change the country via a comboBox which then changes all the information in the mentioned tabs.

For the last step I validated the received API data with draft-07 for JSON and XSD for the XML.

**4. Why did I choose those languages?**

I chose Node.JS for its known high performance (uses Googles JSv8 Engine which compiles JS directly into machine code), easy scalability (horizontal scaling: adding nodes to existing system, vertical scaling: adding extra resources to existing nodes) and caching (gets cached in the application memory, so developers don’t have to re-execute the code). C# on the other hand I chose because I wanted to unveil the true potential of API’s by having a non web-application on the consumer side and its easy to use Windows Form creator. Either way I would not recommend using C# for building fast and easy consumer applications for the reason that you have to write a lot of code to make it work.