Generating Test Scripts for a Single Page Web Application Based On Database Schema

Kantinan Sangwatthanarat[1], Taratip Suwannasart[2]

[1,2] Department of Computer Engineering, Chulalongkorn University, Bangkok, Thailand

Abstract.

Using automatic testing methods for software testing helps reduce mistakes, time and number of manual testing by software testers. However, the testing process still required software testers who have expertise in creating test scripts. The test data is a main component of test scripts. The software testers generate huge volumes of test data to ensure their quality booming contribution in the delivery of the product for real-world use. Manually generate test data from the database will also be a time-consuming option but manually inserting test data into test scripts is not an affordable option by effort as well. This research aims to establish an approach to simplify test scripts generation for web applications using URL and model file from asp.net as inputs. Our approach elicits input elements from a web page using a provided URL and then analyzes their values using model file to extract the data from database to create test data. This produces test scripts that run under Robot framework.

Keywords: Software Testing, Automation Testing, Test Scripts, Test Automation Framework, Robot Framework

1. Introduction

Software testing is an important process that helps verify that the software functions correctly as it unveils faults, verifies correctness, assures quality, and ensures reliability of the software system.[1] Software testing process requires an intensive labour work, timing consumption, and a high budget. Due to the fact of creating the test cases, it has to be created to cover all the function of the software for further development. The software tester uses manual testing to create this test case to eliminate the errors in the software. This is a repetitive and time-consuming task. Thus, the main idea of this project is to use test scripts to decrease those repetitive works, which leads to decrease all the intensive labour work, budget, and reduce time-consuming in the software testing process.

Nowadays, test automation has been implemented to reduce software errors, time-consuming,[2] and labour work of the manual testing of software testers. Automated testing is the use of test scripts as the tools to assist in the software testing process. The software will run the testing process according to the steps that the programmer has an outline in the test script, it will follow every step and activities that have been created in the software. However, in order to create a test script requires an advance knowledge ability in software testing such as expertise in programming languages because creating test scripts use programming language skill to create test scripts, which could be difficult for regular or beginner programmers [3].

Moreover, the test automation framework has been introduced to solve problems and errors [4], such as a robot framework, and more. The robot framework is an automated testing framework that is used to create a test scripts. The robot framework is created based on keyword-driven, which makes it easier to create a test script by eliminating the keywords as a demanding language. In addition, the robot framework can also be connected to external test libraries, by using the external test library. It allows software testing to access a variety of keywords, this is the reason why it becomes popular and commonly used among software testers today. The key component of the test scripts is the test data. It represents data that affects or affected by software execution while testing. The preparation of data for testing is a very time-consuming for software tester and manually inserting test data into test scripts is not an affordable option by effort as well.

This research is to describe a method of generating test scripts for web applications using an automation framework and generate test data from database. Our method uses URL, and model file from asp.net as input information to generate test scripts. First, HTML structure is analyzed from URL to extract input elements which used to generate test scripts with no test data. Next, model file from asp.net is analyzed to extract data from database to inputs of the web applications to generate test data. Finally, the test scripts are combined with test data to generate complete test scripts for further use by software testers.

2. BACKGROUND AND RELATED WORK

2.1 Test Automation [5]

To be able to have a full understanding of this test scripts project, there is requires background knowledge and related work. So software testers could use this information and adapting to the project. The requirement processes are listed in this article. Automated testing is a test tool that creates a series of scripts that mimic the actions of the user. It is testing on the application under test that runs repeatedly with minimum or no human intervention. The application under test can be using any kind of technology or domain. The key goal of the automated testing is to increase the reuse of a set of scripts and comprehensive the tests, so it can be done faster so they can do the test more often and more efficiently. Therefore, this allows the application under test is more reliable and effective.

2.2 Automation Testing Framework [5]

The automation testing framework is a conceptual format that supports the creative process of test scripts to meet the goals of automated testing. The automated testing framework is a layered structure that provides the mechanism for establishing relationships and interacting with each other between layers. The automated testing framework design the test scripts that it is easy to maintain and reuse the test scripts. Automated testing frameworks can be classified by using a type of technique and structure, such as data-driven, keyword driven, and hybrid.

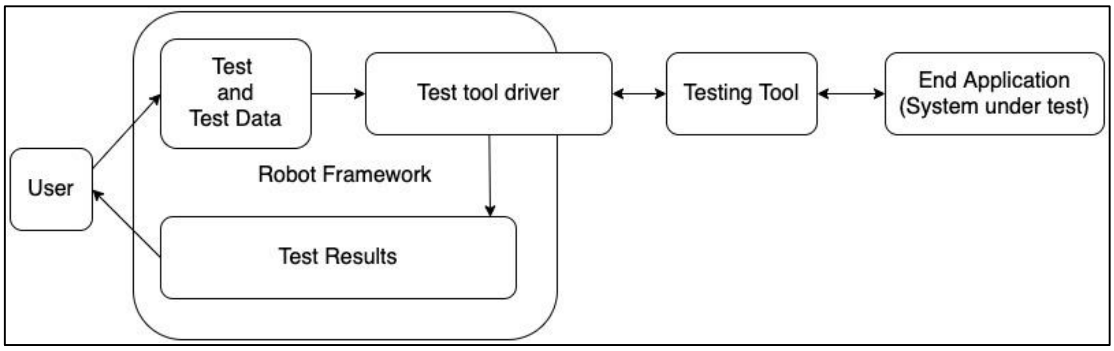
2.3 Keyword-driven Testing [6]

The keyword-driven testing is also known as table driven testing or action based testing. It is a software testing method that uses spreadsheets to identify test cases in a table format. There is a function designed for each keyword which the keywords are stored into a columns table. It is an automated testing technique that separates the program from the test script. The scripts used in the test can be developed based on knowledge of programming. The results are easy to maintain, even though the testing program has been changed a lot. This technique requires the developer to develop the keywords independently from the testing tool in the data testing process. A script contains codes that drive the tests through data and programs, which is similar to the test case of manual testing. The function of the program will be tested and recorded in a table format, as well as the step-by-step instructions for each testing.

2.4 Robot Framework [7]

The robot framework is used to create a framework for the test scripts, which is easy to use the syntax based on the principle of the keyword driven. Since the robot framework is able to connect to the external test libraries, makes it possible to execute keywords from the test libraries that robot frameworks connected. The overall concept of the robot framework is shown in Figure 1 and details are as follows.

Figure 1: Concept overview of the robot framework [7]



2.4.1 Tests and test data: is the default test configuration, which contains the test files and test data, as well as the contents of the files used for testing.

2.4.2 Robot framework: is the main framework for successing the completing of the software testing process.

2.4.3 Test tool drivers: is the tool that use as a communicator between the framework and the testing tool.

2.4.4 Testing tool: is a tool that uses for software testing.

2.4.5 End application (system under test): is the software that has been tested.

2.4.6 Test results: are the final results of the testing. They will be used to determine the results of the testing, as well as to record the results of the testing that can be used to evaluate the differences of software testing.

2.5 SeleniumLibrary [9]

Selenium libraries are libraries that use for testing web applications, which are robot framework external libraries. The selenium web driver is a built-in module used to control the web browser. Selenium libraries are working related along with the HTML elements. Selenium libraries are using the locators that are defined in the test scripts.

Table 1: Input fields under element groups

Strategy	Match based on	Example
id	Element id	id=example
name	Name attribute	name=example
identifier	Either id or name	identifier=example
class	Element class	class=example
tag	Tag name	tag=div
xpath	XPath expression	xpath=//div[@id="example"]
css	CSS selector	css=div#example
dom	DOM expression	dom=document.images
link	Exact text a link has	link=The example
partial link	Partial link text	partial link=he ex

In addition, from Table I can be found that there are several methods for identifying the locator, for instance:

- Specifying the locator by id is a method that used with the id assigned element. If the position of this element is changed; it will not affect the test script, such as id = txtSearch.

- Specifying a locator with name is a method that is applied to the element whose name is defined, such as name = btSubmit.

2.6 Model-View-Controller [10]

Model-view-controller is a type of software architecture. The software structure is divided into 3 main parts including Model, View, and controller.

2.6.1 Model is the step that interfaces with the database. Managing in and out data use it in analyzing processes. The model contains classes that are connected to the database. When the data is loaded in from different places into the model part. The model manages and prepares the data in a suitable format that is waiting for the data to be executed from the controller.

2.6.2 View is the display of the web browser in HTML format, which receives the commands from data controller and delivers the data from the model to display in the system.

2.6.3 Controller is the first part that starts running when starting up the Web browser program, which is the main part of this process. The Controller interacts the database from the Model and displays the data through View.

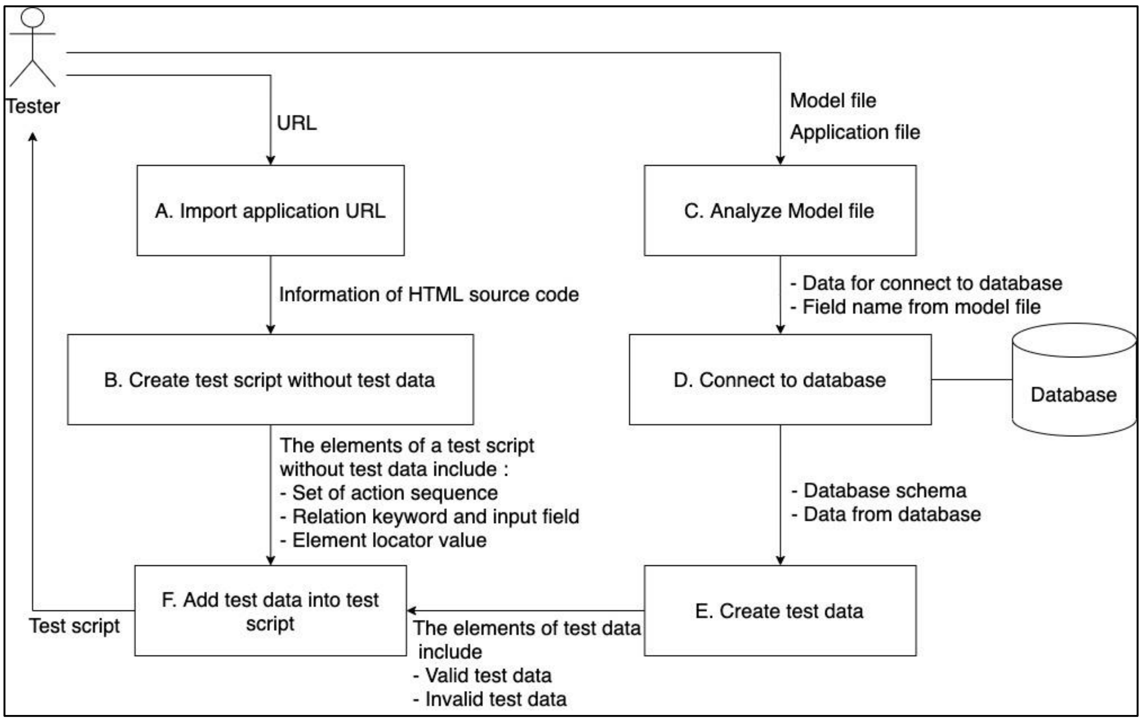
2.7 ASP.NET MVC [12]

ASP.NET MVC is a web application development framework designed to support the traditional software architecture model-view-controller. In which separating the functions of the elements in the application into 3 main folders. First, models folder, it stores model files that are interfaces with the database. Second, views folder, it stores html files as part of the display. Lastly, controller folder, it will store the controller file as part of the main processing of the program.

3. PROPOSED APPROACH

The research has presented the way to create test scripts for web applications based on a database schema. By reading the URL name and importing HTML source code to create a test script without test data based on the structure of the robot framework, the model file eliminates both database schema and the data. The data will be stored in the database ready to be analyzed and used to create test data and add test data to test scripts for software testers to use. An overall ideal image of this research is shown in Figure 2. According to the diagram, the overview of the research shown 6 main steps, such as Import Application URL, create a test script without test data, analyze model files, connect to database (the program database to extract the database schema and data from the program database based on the model file), lastly, create test data and add the test data to the test scripts with details of each step.

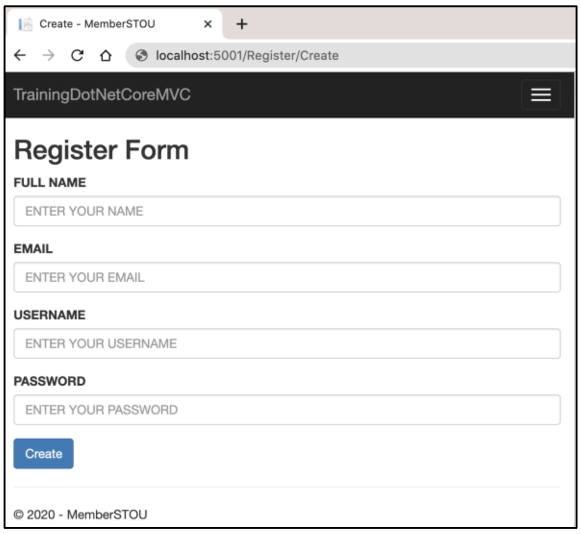
Figure 2: A conceptual process diagram


Tester
URL
Model file
Application file
A. Import application URL
C. Analyze Model file
Information of HTML source code
- Data for connect to database
- Field name from model file
B. Create test script without test data
D. Connect to database
Database
The elements of a test script without test data include :
- Set of action sequence
- Relation keyword and input field
- Element locator value
- Database schema
- Data from database
F. Add test data into test script
E. Create test data
Test script
The elements of test data include
- Valid test data
- Invalid test data

3.1 Import application URL

First of all, software testers need to import the URL of the web application they want to the software testing. For example, a software tester imports a URL of a web application as http://localhost:5001/Register/Create, as shown in Figure 3. Next, the URL will be imported to html source code to prepare for create a test scripts without test data in next step.

Figure 3: An example of web application screen, data acquisition section



Create - MemberSTOU

localhost:5001/Register/Create

TrainingDotNetCoreMVC

Register Form

FULL NAME

ENTER YOUR NAME

EMAIL

ENTER YOUR EMAIL

USERNAME

ENTER YOUR USERNAME

PASSWORD

ENTER YOUR PASSWORD

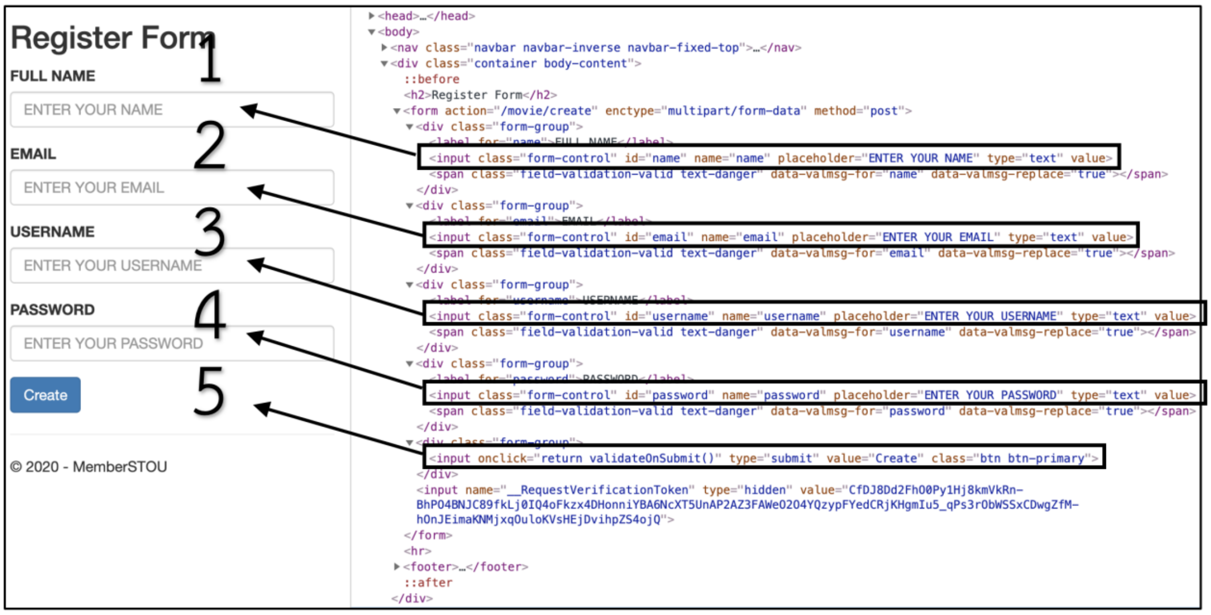
Create

© 2020 - MemberSTOU

3.2 Create test scripts without test data

Second, at this stage, the input field will be created in order to establish the relationship between the input fields and the keywords. Therefore, to eliminate the locator values from the HTML source code files. This started with analyzing the HTML source code files that created the working demand for the input fields. It establishes the working demand from reading HTML source code files. This gives the program working demand that will link the web application screens template. An example of creating a working demand between the HTML source code file and the web application part of the web application screen is shown in Figure 4 where the web application screen template of the input fields are Full Name, Email, Username, and Password. It is reading from the html source code, which create a working demand as in Table 2.

Figure 4: An example of creating an input field working sequence of figure 3


Register Form

FULL NAME

ENTER YOUR NAME

EMAIL

ENTER YOUR EMAIL

USERNAME

ENTER YOUR USERNAME

PASSWORD

ENTER YOUR PASSWORD

Create

© 2020 - MemberSTOU

```
▸<head>…</head>
▾<body>
  ▸<nav class="navbar navbar-inverse navbar-fixed-top">…</nav>
  ▾<div class="container body-content">
      ::before
      <h2>Register Form</h2>
    ▾<form action="/movie/create" enctype="multipart/form-data" method="post">
      ▾<div class="form-group">
          <label for="name">FULL NAME</label>
          <input class="form-control" id="name" name="name" placeholder="ENTER YOUR NAME" type="text" value>
          <span class="field-validation-valid text-danger" data-valmsg-for="name" data-valmsg-replace="true"></span>
        </div>
      ▾<div class="form-group">
          <label for="email">EMAIL</label>
          <input class="form-control" id="email" name="email" placeholder="ENTER YOUR EMAIL" type="text" value>
          <span class="field-validation-valid text-danger" data-valmsg-for="email" data-valmsg-replace="true"></span>
        </div>
      ▾<div class="form-group">
          <label for="username">USERNAME</label>
          <input class="form-control" id="username" name="username" placeholder="ENTER YOUR USERNAME" type="text" value>
          <span class="field-validation-valid text-danger" data-valmsg-for="username" data-valmsg-replace="true"></span>
        </div>
      ▾<div class="form-group">
          <label for="password">PASSWORD</label>
          <input class="form-control" id="password" name="password" placeholder="ENTER YOUR PASSWORD" type="text" value>
          <span class="field-validation-valid text-danger" data-valmsg-for="password" data-valmsg-replace="true"></span>
        </div>
      ▾<div class="form-group">
          <input onclick="return validateOnSubmit()" type="submit" value="Create" class="btn btn-primary">
        </div>
        <input name="__RequestVerificationToken" type="hidden" value="CfDJ8Dd2Fh0OPy1Hj8kmVkRn-
        BhPO4BNJC89fkLj0IQ4oFkzx4DHonniYBA6NcXT5UnAP2AZ3FAWeO2O4YQzypFYedCRjKHgmIu5_qPs3rObWSSxCDwgZfM-
        hOnJEimaKNMjxqOuloKVsHEjDvihpZS4ojQ">
      </form>
      <hr>
    ▸<footer>…</footer>
      ::after
    </div>
```

Table 2: Order of operation of input fields of figure 4

Input field name	Order of work	Field input type
FULL NAME	1	text
EMAIL	2	text
USERNAME	3	text
PASSWORD	4	text
CREATE	5	submit

The next step is to establish the relationship between the input field of HTML source code and the selenium library keyword. This is a procedure that specifies how the input fields that appear on the web application screen for the test script to work. The selenium library keyword is used as a command. Data in Table 3. shows the relationship between the input fields of HTML source code and the selenium library keyword.

Table 3: Relationship between input fields of HTML source code and selenium library keywords.

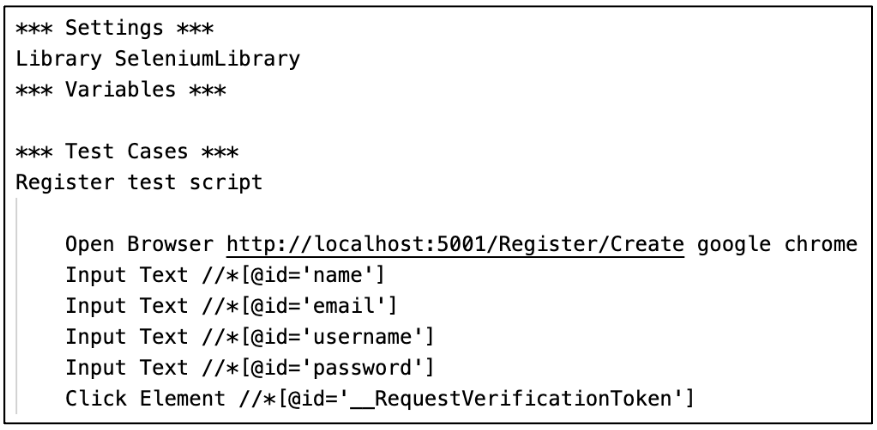
Name	Type	SeleniumLibrary keywords
FULL NAME	text	Input Text
EMAIL	text	Input Text
USERNAME	text	Input Text
PASSWORD	text	Input Text
CREATE	submit	Click Element

Furthermore, in order to send commands to control input fields with the selenium library keyword, it must know the location of the input fields on the web application screen. Meaningly, for each input field, it must be configured the locator first to know the position of each input field. For example, the input field Full Name can be located from the locator with id = ”name”. The email field input can be located from any locator with the value id = ”email”. The input username field can be located from the value locator id = ”username”. After obtaining the keywords for each input field and configuring the locator, the keywords and locator values for each input field are created as in Table 4. and the data is then used to generate a test script with no test data as shown in figure 5.

Table 4: Generating keywords and locator values for each input field.

Keyword	Locator	Keywords and locator values for each input field.
Open Browser	http://localhost:5001/ Member/Create	Open Browser http://localhost:5001/ Member/Create google chrome
Input Text	//*[@id='name']	Input Text //*[@id='name']
Input Text	//*[@id='email']	Input Text //*[@id='email']
Input Text	//*[@id='username']	//*[@id='username']
Input Text	//*[@id='password']	Input Text //*[@id='password']
Click Element	//*[@id=' __RequestVerificationToken']	Click Element //*[@id=' __RequestVerificationToken']

Figure 5: Example of a test script with no test data.



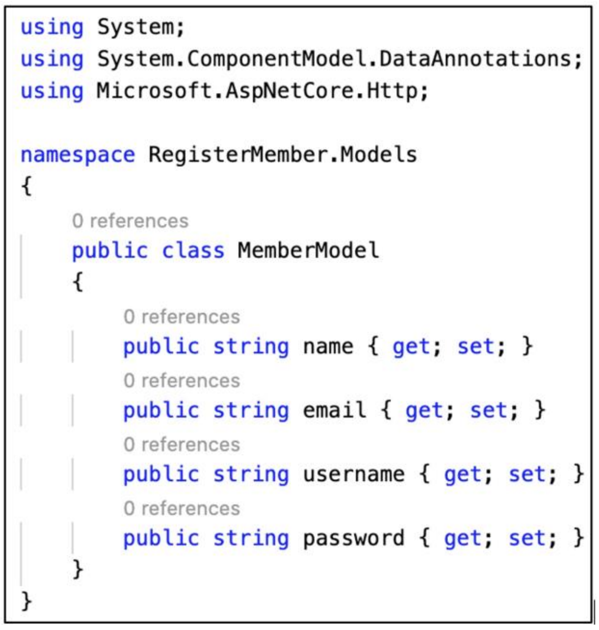
```
*** Settings ***
Library SeleniumLibrary
*** Variables ***

*** Test Cases ***
Register test script

    Open Browser http://localhost:5001/Register/Create google chrome
    Input Text //*[@id='name']
    Input Text //*[@id='email']
    Input Text //*[@id='username']
    Input Text //*[@id='password']
    Click Element //*[@id='__RequestVerificationToken']
```

3.3 Analyze Model files

Third, software testers import the application model file to test and then analyze the model file. The analysis was to perform the results of attributing the information from model files. Figure

6 shows an example of model files that attributed. A software tester uses those processes included name, email, username, and password, to analyze to find program database connection from application files

Figure 6: Example of a test script with no test data.

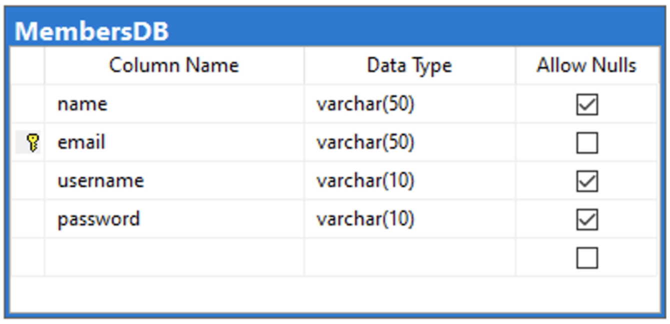


```
using System;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Http;

namespace RegisterMember.Models
{
    0 references
    public class MemberModel
    {
        0 references
        public string name { get; set; }
        0 references
        public string email { get; set; }
        0 references
        public string username { get; set; }
        0 references
        public string password { get; set; }
    }
}
```

3.4 Connect to the database

Fourth, this step is to takes the program database connection information from the application file and connect it to the program database and eliminate the attribute name from the model file to the attribute data from the database schema. Figure 7. shows an example of attribute data in a database schema including name (50-byte varchar), email (50-byte varchar) with primary key, username with data type (10-byte varchar) and password (10-byte varchar).

Figure 7: Example of attribute data in the database schema.

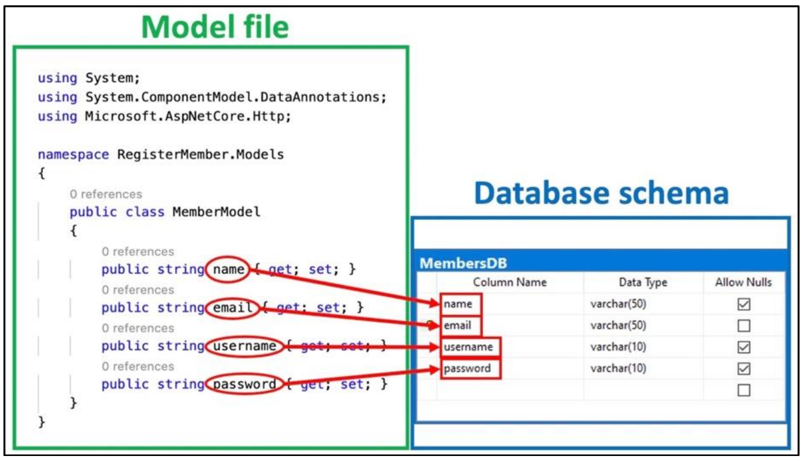


MembersDB

	Column Name	Data Type	Allow Nulls
	name	varchar(50)	☑
🔑	email	varchar(50)	☐
	username	varchar(10)	☑
	password	varchar(10)	☑
			☐

Figure 8. shows an example of extracting attribute names from the model file from the database schema. The model file is related to the web application screen, the data acquisition section. The database schema can be extracted to test data in the next step as shown in Table 5.

Figure 8: An example of extracting attribute names from model files from database schema.



```
using System;
using System.ComponentModel.DataAnnotations;
using Microsoft.AspNetCore.Http;

namespace RegisterMember.Models
{
    0 references
    public class MemberModel
    {
        0 references
        public string name { get; set; }
        0 references
        public string email { get; set; }
        0 references
        public string username { get; set; }
        0 references
        public string password { get; set; }
    }
}
```

MembersDB

Column Name	Data Type	Allow Nulls
name	varchar(50)	✓
email	varchar(50)	☐
username	varchar(10)	✓
password	varchar(10)	✓
		☐

Table 5: Extractable database schema.

Attribute name	Data type	Size (bytes)	Key type
name	varchar	50	
email	varchar	50	Primary key
username	varchar	10	
password	varchar	10	

The next step is to use the extracted database schema and extract the data stored in the database to create the test data in the next step as shown in Table 6.

Table 6: Data stored in the extracted database.

name	email	username	password
somchai jitdee	somchai.j@gmail.com	somchai111	p56789
donald mcway	mcway.s@hotmaill.com	ddmanq	jj8790
peter parker	peter.p@yahoo.com	peter555	ps12345
john smith	john.s@gmail.com	john111	sm999

3.5 Create test data

Fifth, this step is to create test data. This will take the database schema and the data stored in the database. To create the test data for the data input screen and create the test data for the data query type screen as further described.

3.5.1 Create the test data for the data input type screen. Valid the test data and takes the size of the attribute from the database schema to generate the test data. For example, the name attribute is 50 bytes, so the test data is 50 characters or less. And the Invalid test data will create the data, the test with more than 50 characters, as in Table 7.

Table 7: Test data for each input field generated in accordance with in 3.4.1 and the database schema in table 5.

Input field name	Attribute name	Data type	Size (bytes)	Valid data	Invalid data
FULLNAME	name	varchar	50	abcdeabcde	abcdeabcdeabcdeabcde abcdeabcdeabcdeabcde abcdeabcdeabcdeabcde
EMAIL	email	varchar	50	fehijkfehijk	bbbbbbbbbabcdeabcde abcdeabcdeabcdeabcde abcdeabcdeabcdeabcde
USERNAME	username	varchar	10	aaaaaaaaaa	aaaaaaaaaaa abcdeabcde
PASSWORD	password	varchar	10	bbbbbbbb	bbbbbbbbbabcdeabcde

3.5.2 Creating a test data for the data query type screen. This process is to create the test data from the data stored in the database. Valid test data by creating test data in case the data is found in the database and the data is not found in the database as in table 8.

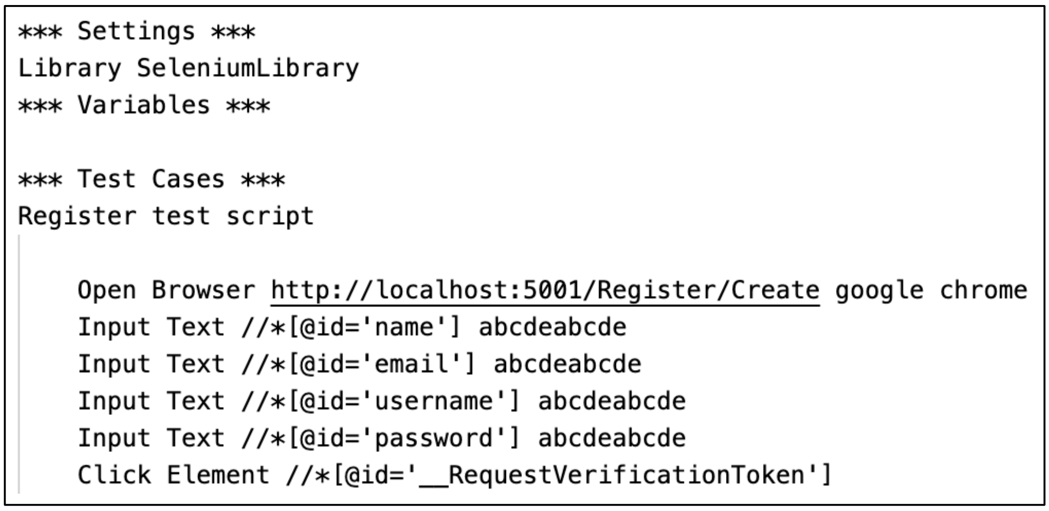
Table 8: Examples of generating test data for data query type screens.

Test Case	Test Data	Expected Output
1	somchai.j@gmail.com	Show data table somchai jitdee, somchai.j@gmail.com, somchai111, p56789
2	mcway.s@hotmaill.com	Show data table donald mcway, mcway.s@hotmaill.com, ddmanq, jj8790
3	peter.p@yahoo.com	Show data table peter parker, peter.p@yahoo.com, peter555, ps12345
4	john.s@gmail.com	Show data table john smith, john.s@gmail.com, john111, sm999
5	abcdeabcde	Show message Data not found
6	pantiswaert	Show message Data not found

3.6 Add test data into the test scripts

The final step of creating the test script is to add the test data into a test scripts for create a complete set of test scripts that is ready for software testers to use. Each test case will be added to the test scripts and will create test data for all cases as. After adding the test data to the test script, there will be a ready to use test scripts available for the software testers to test with a web application as in Figure 15.

Figure 9: Example of a test script containing test data



```
*** Settings ***
Library SeleniumLibrary
*** Variables ***

*** Test Cases ***
Register test script

    Open Browser http://localhost:5001/Register/Create google chrome
    Input Text //*[@id='name'] abcdeabcde
    Input Text //*[@id='email'] abcdeabcde
    Input Text //*[@id='username'] abcdeabcde
    Input Text //*[@id='password'] abcdeabcde
    Click Element //*[@id='__RequestVerificationToken']
```

4. Conclusion

To summarize our proposal on creating the test scripts for a web application based on the database schema, the given web application URL is read to collect the HTML input elements, and the model file from asp.net is imported. After that, the input elements are analyzed to create test scripts without test data. Finally, corrected data from databases to create test data before adding to the test scripts. With this, time consumption in creating test scripts for web applications can be allow a tester to significantly reduce time consumption. In addition, our test scripts can be executed by Robot Framework. Nevertheless, our approach showed that it supports the web applications developed with Asp.net, MVC connected to a Microsoft SQL Server database, even though the web applications that connect to other databases are not supported. For further developments, we are planning to eliminate these limitations by creating the test scripts that can work on other databases.

References

[1] Jorgensen, P.C., Software Testing A Craftsman's Approach Fourth Edition. 2014: Taylor & Francis Group, LLC.

[2] Axelrod, A., Complete Guide to Test Automation. 2018: Springer. pp. 13-15

[3] International Software Testing Qualifications Board (ISTQB), Certified Tester Advanced Level Syllabus Test Automation Engineer 2016. p. 13.

[4] International Software Testing Qualifications Board (ISTQB), Certified Tester Advanced Level Syllabus Test Automation Engineer. 2016. p. 31.

[5] Bhargava, A., Designing and Implementing Test Automation Frameworks with QTP. 2013: Packt Publishing Ltd.

[6] Rashmi and N. Bajpai. A Keyword Driven Framework for Testing Web Applications. In International Journal of Advanced Computer Science and Applications (IJACSA). 2012.

[7] Bisht, S., Robot Framework, in Robot Framework Test Automation. 2013, Packt Publishing Ltd. P. 9-11.

[8] Ry, R.F., SeleniumLibrary. 2020.

[9] SeleniumLibrary. Robot Framework Ry.

[10] Shklar, L. and R. Rosen, Model-View-Controller, in Web application architecture : principles, protocols, and practices. 2009, John Wiley & Sons Ltd. P. 242-244.

[11] Corporation, M., ASP.NET Core MVC. 2020.

[12] ASP.NET Core MVC. Microsoft Corporation.