Task - Performance and Load Testing with Visual Studio: As pre-step You need to install and configure Visual Studio ULTIMATE. Also SQLEXPRESS 2012 (both SERVER & MANAGER) e.g.

<http://www.youtube.com/watch?v=vng0P8Gfx2g>

Be sure that SQLEXPRESS service is running! You can run SQL Server Configuration Manager and start it manually! Be careful for the correct INSTANCE\_NAME. Also in Task Manager > Services. Next go to Start menu and find VS2012 Cross Tools Command Prompt. Type the following:

cd **C:\ Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE**

and

**SQLCMD /S localhost\sqlexpress /i loadtestresultsrepository.sql**

This will create Load test DB schema that is required for VS 2012 Load tests.

1. On the **Load Test** toolbar, choose **Manage Test Controllers**. The **Manage Test Controllers** dialog box is displayed.
2. In the **Load Test Results Connection String,**click the browse button (…) to display the **Connection Properties** dialog box.

|  |
| --- |
| **Note Note** |
| If you change the connection string for a controller, then you must select the controller. |

1. In **Server Name**, type **localhost\sqlexpress** or the name of the server that you used in step 2 such as **ContosoServer1**.
2. Under **Log on to the server**, choose **Use Windows Authentication**.
3. Under **Connect to a database**, choose **Select or enter a database name**. Select **LoadTest** from the drop-down list box.
4. Choose **OK**.

Run tests from 

1. It’s good to create and use COMMENTS when recording Performance tests. For each page or action e.g.

<https://www.youtube.com/watch?v=P1ZqQFj1LPY>

You can set Validation rules at 

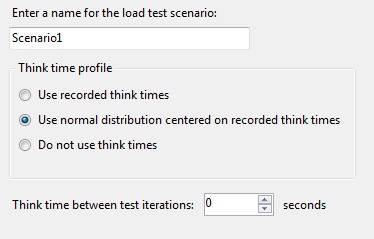
Also it’s nice to use Data Driven approach by adding test data via .

Add CSV file (pre-condition: first create such from .txt and change to .csv in Your Windows file system, then in Solution Explorer add ‘Data’ folder). Add pairs like

|  |
| --- |
| Username, Password |
| qa-academy-1, qa-academy |
| qa-academy-2, qa-academy |

Next You must find in web test where the username and pass are passed as params. When You find them You must replace them with Right Click > Properties (F4) > Value, then Select CSV file column.

1. Create SQL DB for Load tests records. Can be empty at the beginning. Whenadding Load Test You must set in wizard:

* Proper name and ‘think time’ 
* Select a Load Pattern (according to your needs)
* ‘Tests mixed modeled’ e.g. ‘Based on the total number of tests’
* Add already created Performance tests You need
* ‘Network types’ should be ‘LAN’
* In ‘Browser types’ you can include and manage usage
* Set correct ‘Load test duration’

In Run Settings > Run Settings1 [Active] > Counter Set mappings > [CONTROLLER MACHINE] > Load Test > right-click ‘Manage Test Controllers’. You must add Connection String to SQL DB in ‘Load tests result store’ (Data Source=USER-PC\SQLEXPRESS2012; …….).

To generate a report in MS Excel for your Load Tests as pre-step You need to install and configure ‘Load tests tab’. Go to File > Options > Add-ins > Manage (select COM Add-ins) > click Go > Select ‘Load test report Add-in’. In the main Excel menu there is a ‘new’ tab – Load test. Select ‘Load test report’. As ‘Server name’ must be entered user\sql instance (You can get it from SQL Management Studio loading/connecting screen). Collected metrics can be used to compare different runs.

If You are given the source code of a Web application (**ASP.Net-Example.rar**) and Your task is to deploy the application in IIS on your local machine and perform the following tests, You can create ‘startWebDev.ps1’ script and navigate it to .sln/.csproj file in solution folder!

Double check the web**.config** file and <connectionStrings>

<add name="ApplicationServices" connectionString="Data Source=USER-PC\SQLEXPRESS;

If needed change the Authentication to sa/123456. Be careful it’s case-sensitive.

Task - Cross Site Scripting (XSS) Vulnerabilities: Most important is to try to catch ‘unexpected behavior’ (especially if You have different roles ‘admin’ & ‘user’ ) – You can use the following approach:

1. Fill all inputs with XSS like <script>alert(“XSS”);</script> for each role page
2. Go through each role page and find where the JS script is not parsed as simple text (issue can be in the same page, but to require some action e.g. select different ‘Projects’ from menu; click on ‘Edit’ button/link)
3. Try to exploit the XSS Vulnerability that you’ve found

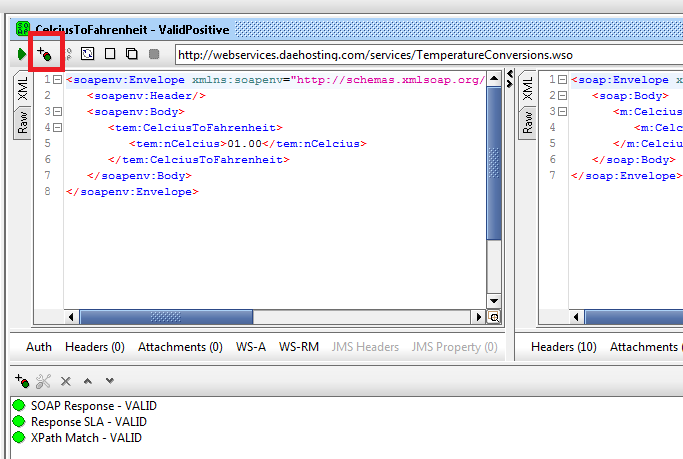
Task - SQL Injection Vulnerabilities: You can use the same approach as for ‘XSS’. But as input this time use single quote ‘ . When you find the input field, try to create SQL script that can exploit it (e.g. delete all records from table). If application provides ‘raw’ Exceptions message with ‘Incorrect syntax’ – try to get DB info from it.

Example SQL:

'; delete from dbo.Bugs; --

Task - URL Manipulation Vulnerabilities: Since You may have the Project code, go to ‘Pages’ folder and try to find page.cs/view that is not for the current role (e.g admin & user have some page in common). Also You can try in .txt file to copy all pages URLs for each role and try to compare them e.g. ../id=1; …/mode=administrator

Task - WebService Testing: As pre-step you need to install and configure SoapUI. When creating ‘New SOAP project’, You need to have WSDL file (for the service Interface). Once added it will provide You with the methods you need to test. For each method you can add ‘Create New request’ (use it as unique test case). Choose them carefully e.g. If Method is for temperature, You must create via BVAnalysis one negative & one positive for ‘Absolute zero’ value!

It’s important in the body of the new request to get the correct XPath for the response value! Each Test case from Interface methods can be added to a Test Suite via right-click ‘Add to test case’. In Test Suite > Test steps, You can add Assertions for each test case via 

It’s very important when dealing with values to use the correct XPath! Please see the example:

REQUEST:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://webservices.daehosting.com/temperature">

<soapenv:Header/>

<soapenv:Body>

<tem:CelciusToFahrenheit>

<tem:nCelcius>01.00</tem:nCelcius>

</tem:CelciusToFahrenheit>

</soapenv:Body>

</soapenv:Envelope>

RESPONSE:

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

<soap:Body>

<m:CelciusToFahrenheitResponse xmlns:m="http://webservices.daehosting.com/temperature">

<m:CelciusToFahrenheitResult>33.8</m:CelciusToFahrenheitResult>

</m:CelciusToFahrenheitResponse>

</soap:Body>

</soap:Envelope>

XPath Match Assertion:

declare namespace m='http://webservices.daehosting.com/temperature';

//m:CelciusToFahrenheitResult[1]

ALWAYS add Expected result value!

When creating Load Tests – ‘Number of threads’ should be assumed as users.

When creating Property Transfer – both files should be in one Test suite! Test Suite > Test Case > Add step > Property transfer. Choose Source (test case donor) drop-down list and Property: Response. Get the XPath for the value (from Source input) and create correct XPath value for Target input value. Target drop-down list should be selected for recipient Test case and Property: Request.