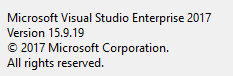
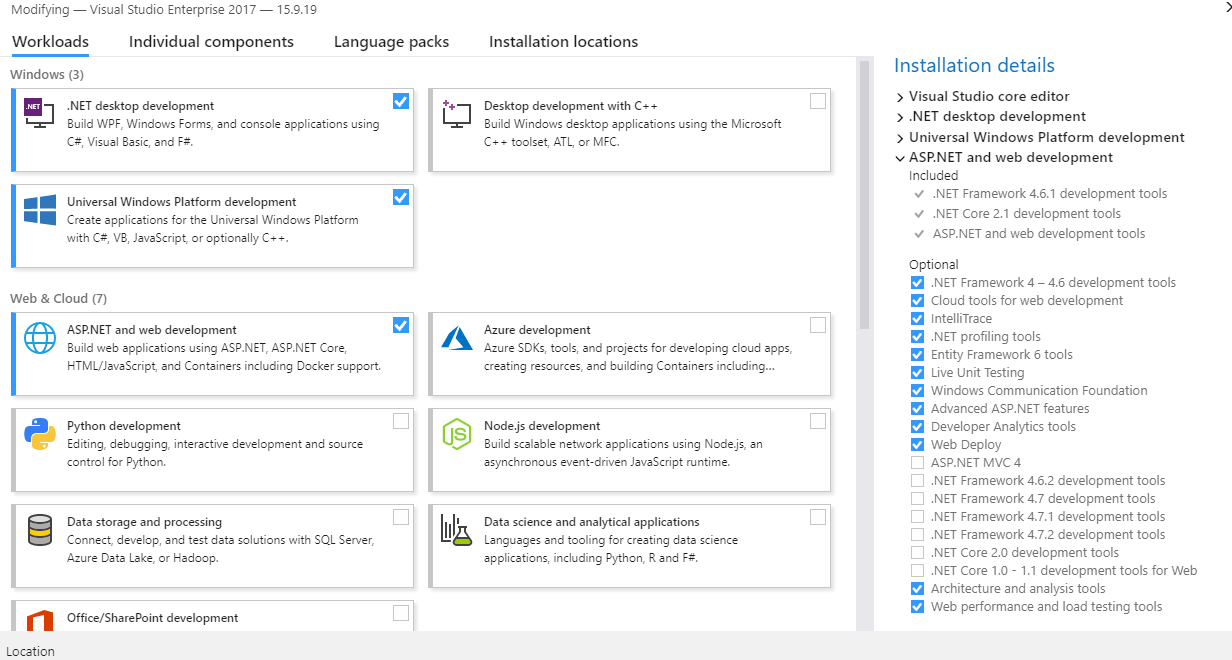
.Building a Performance Test with VS

# Preparing the Performance environment

## Installing VS



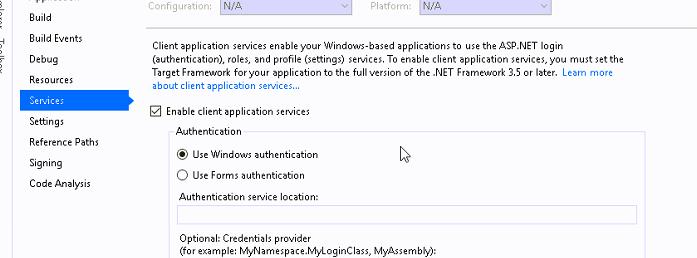


Make sure Web performance and load feature is installed

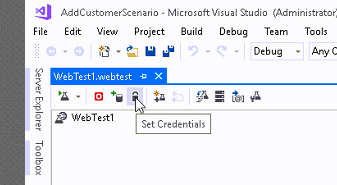
Tools->Get Tools and Features …

## Configure Visual Studio for each test

* Enable Windows Authentication
  + Project->Properties->Services
    - Select “Enable client application services”
    - Select “Use Windows Authentication”



* Add credentials
  + In the WebTest window, find the “Set Credentials” icon
    - Add user/pwd



## Set up IE

<https://devblogs.microsoft.com/devops/using-internet-explorer-11-and-not-able-to-record-a-web-performance-test-successfully-2/>

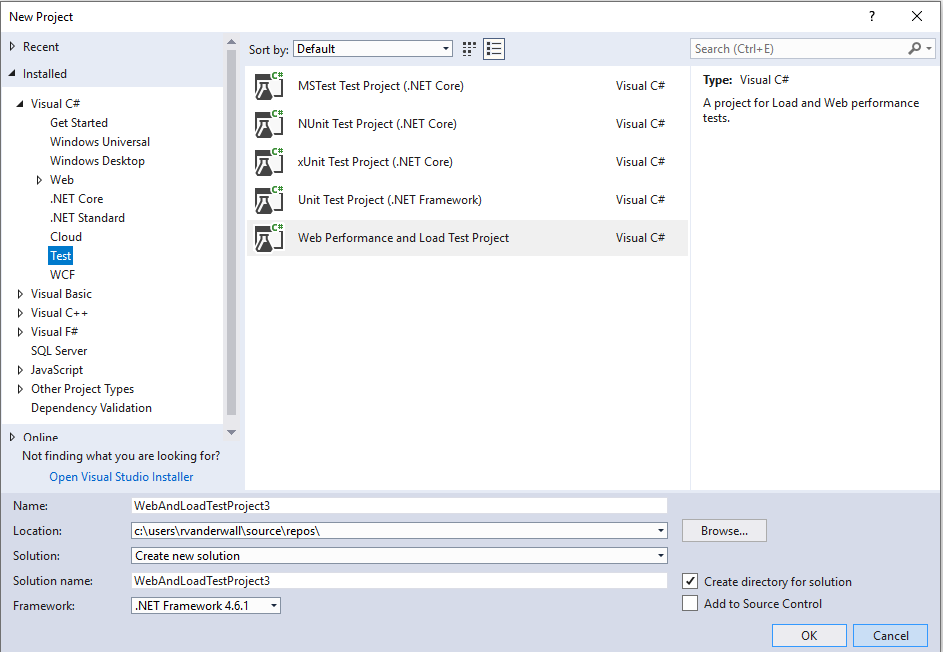
* Settings -> Internet options->Security
  + Uncheck “Enable Protested Mode”
* Settings -> Manage Add-ons -> Toolbars and Extensions
  + Enable Web Test Recorder 15.0

# Create Web Test

File->New->Project

Installed->Visual C#-> Test -> Web Performance and Load Test Project

Provide a Name and Location



# Creating the test

Creating a usable load test require several steps. The basic outline is shown below and following sections provide some details.

1. **Recording**: The first step is to record the actions.
2. **Checkpoint**: During Record, make sure to introduce checkpoints/transactions.
3. **Correlate**: Capture and extract session variables that are used in subsequence requests. Make sure you can run the test repeatedly.
4. **Validation**: Add validation to ensure the responses are as expected.
5. **Parameterize**: Extract parameters such as host, users, etc. so the test can simulate different users

## 1 Recording

* Add, if one doesn’t exist, a webtest to the Test project.
* Select the webtest tab and then the webtest itself. Right click -> Add Recording …
* It’s often a good idea to record a very simple login to make sure the credentials are set up correctly. You should be able to replay it repeatedly without issue.

## 2 Transactions

Data Binding:

{{<db>.<table>.<field>}}

[https://test/{{Datasource1.TheFileName#csv.TheFieldName}}.com](https://test/%7b%7bDatasource1.TheFileName#csv.TheFieldName}}.com)

# AddUserScenario

* Make sure the Web Enrollment service is running
  + Log in and it will minimize
* Record using the front end server
  + Add comment “Logging in”
  + URL: https://crm-fe01q-585.d360.ok.gov
  + Verify credentials are correct before playback
* Add comment “Search Customer”
* Enter LastName=TestUser00001 and Year of Birth=1999
  + Pick a user that doesn’t exist, but you can find in the script
* Click ‘Exact match search’
  + You should see “No results were found.”
* Add comment “Start New transaction”
* Click ‘Add new customer’
* Add comment “Adding new customer”
* On the first New customer page:
  + Click ‘No First name’
  + Click ‘No Middle name’
  + Enter 1/1/1999 for DOB
    - Be sure to tab out of the field
  + Add comment “Entering C360”
  + Click “Customer 360”
* Add comment “Entering DL Original”
* On the Customer 360 Page:
  + Click “DL Original”
* Add comment “Starting DL Process”
* On the DL page
  + Click Scan
  + Category: Issuance
  + Document Type: DL10
  + Scan date
  + Correspondence number: 12345
  + Case Number 54321
  + Click the “Scan” icon in the scanner area
  + Click Save
* On the DL Page
  + Click Application
  + Expiration Month: Issuance
  + SSN: 123456790
  + Real ID Conformant” Compliant
  + Gender: M
  + Eye color: Dichromatic
  + Height: 506
  + Weight: 150
  + Address:
  + Veteran: No
  + US Citizen: Yes
  + Organ Donor: Yes
  + Register to Vote: No Response
  + Vision:
    - 20/20 all fields
    - 166 both eyes
    - Test Result Reason: Test given with no discrepancy
  + Add an Address
    - 3560 N Martin Luther King Blvd, OKC, OK 73111
* Click “End/Rest”

Once done scripting, convert comments to transactions.

### Parameterize

* Convert test to code (“Generate Code”)
* Replace username assignment with call to TestHelper
  + TestHelper th = new TestHelper();
  + this.UserName = th.getUserName();
  + this.Password = th.getPassword();
* Replace query with th. generateQueryString();
* Find all occurrences of “TestUser00001” and replace with call to th.getCustomerName()
* Find all occurrences of the SSN and replace with call to th.getSSN()

# Test Cleanup

## Cleaning up load tests

* Select the loadtest tab, then select the “Open and Manage results”
* Select the old tests and remove them.

## Cleaning up test runs

* Navigate to source/repos/<solution>/TestResults
* Delete the file and the folder
* Restart VS

Manual Correlation:

“Generate Code”

Add this code snippet to the GetEnumerator() method replacing the existing this.UserName

TestHelper th = new TestHelper();

this.UserName = th.getUserName();

this.Password = th.getPassword();

this.Context.CookieContainer = new System.Net.CookieContainer();

Find the call to IDMCRMServices/Transaction/TransactionSer and add this code immediately before the yield

ExtractText extractionRule1 = new ExtractText();

extractionRule1.StartsWith = @"TransactionId"":""";

extractionRule1.EndsWith = @"""";

extractionRule1.Index = 0;

extractionRule1.IgnoreCase = false;

extractionRule1.UseRegularExpression = false;

extractionRule1.HtmlDecode = true;

extractionRule1.Required = false;

extractionRule1.ContextParameterName = "TransactionId";

request17.ExtractValues += new EventHandler<ExtractionEventArgs>(extractionRule1.Extract);

ExtractText extractionRule2 = new ExtractText();

extractionRule2.StartsWith = @"EntityId"":""";

extractionRule2.EndsWith = @"""";

extractionRule2.Index = 0;

extractionRule2.IgnoreCase = false;

extractionRule2.UseRegularExpression = false;

extractionRule2.HtmlDecode = true;

extractionRule2.Required = false;

extractionRule2.ContextParameterName = "EntityId";

request17.ExtractValues += new EventHandler<ExtractionEventArgs>(extractionRule2.Extract);

Be sure to change the request<N> number to match the actual request.

Find the first forms\_/read/page.aspx call after the transaction and add this code:

ExtractText extractionRule3 = new ExtractText();

extractionRule3.StartsWith = @"x22prod\_customeridentityid\x22\x3a\x7b\x22\_visible\x22\x3a\x22inline-block\x22,\x22\_noread\x22\x3a\x22none\x22,\x22value\x22\x3anull,\x22oid\x22\x3a\x22\x7b";

extractionRule3.EndsWith = @"\x7d\x22";

extractionRule3.Index = 0;

extractionRule3.IgnoreCase = false;

extractionRule3.UseRegularExpression = false;

extractionRule3.HtmlDecode = true;

extractionRule3.Required = false;

extractionRule3.ContextParameterName = "CustomerId";

request18.ExtractValues += new EventHandler<ExtractionEventArgs>(extractionRule3.Extract);

Find the transaction ID that was used in the record. You should be able to search on TransactionId

The entity ID can be found in the extraqs of the next call to main.aspx

Find the Customer ID that was used in the record. You should be able to search on customerIdentityId

Assume you find this Transaction ID: 35259B05-366A-EA11-B83B-005056AE29FA

Search for the last 6 characters of the ID and you should find the Entity ID. It will be a GUID that is different by typically 1 character

Transaction ID: 35259B05-366A-EA11-B83B-005056AE29FA

Customer ID: 39259B05-366A-EA11-B83B-005056AE29FA

Entity ID: 3B259B05-366A-EA11-B83B-005056AE29FA

Replace these values in the correlate.py script and run it against the generated CS file