# Green Lantern Framework

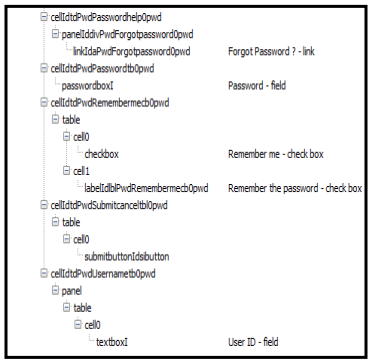
Sridevi Pinjala
IBM

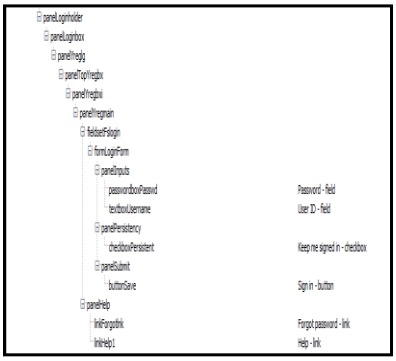
## Assumptions

- Audience is familiar with automation tools.
- Familiar with Record and Play
- Familiar with Capture and Script
- I am positive that you can take this concept and make it your own and implement it with the automation tool of your choice.

- This tutorial Is not about explaining the pros and cons of automation. It is about a new framework that mainly focuses on saving the automation script to handle UI & DOM changes. It simultaneously enables us to write our scripts in English (literally)

# Introduction




What happens to the automated regression scripts if the DOM changes by a little or a lot?

How can the elements used for automation be identified, if the application has complex HTML?

# Introduction




What happens to the automated regression scripts if the UI changes

***Note about Automation Tools***

*Regression automation tools like QTP, Test Complete, Silk Test, Selenium, etc. have been used for regression tests.*
*The tools cannot test a thing. All they do is –*
*Click buttons*
*Type in text in text box fields*
*Click links*
*Click radio buttons*
*Check ON or OFF check boxes*
*Compare text*
*log messages, log warning, log errors etc.*
*They do what we tell them to do.*
*The tool is only as good as the user.*

# Concept

Main elements in a software application are

- Text fields
- Check boxes
- Radio buttons
- Links
- Menus
- Captions
- Images
- Logos

Every test uses some or all the elements in a page

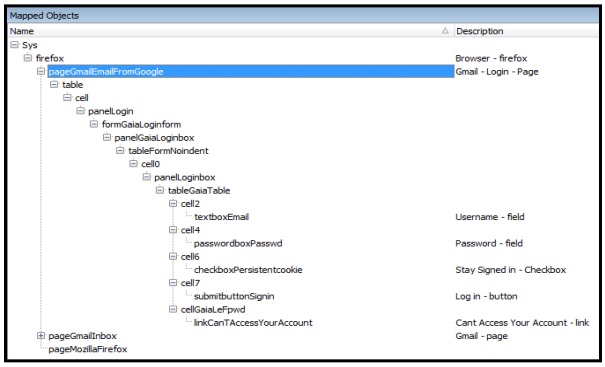
# Green Lantern Steps

1. Choose elements for automation
2. Capture the elements to the repository
3. Rename them in English as they would appear on the page
4. Categorize them as per their type
5. Write scripts using the pseudonym names

# Choose elements for automating login function using Gmail

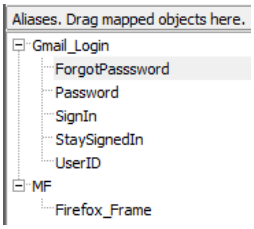

Main elements for this UI are URL, Username, Password, Stay Signed in and Sign in button.

# Capture the elements to the Repository



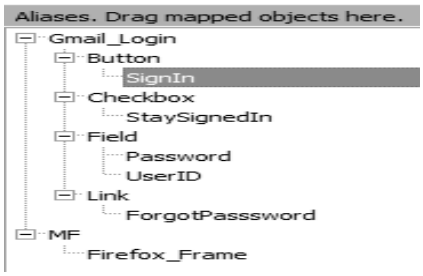

The logical names of the elements are not as they would display on the UI

# Rename the Elements




Drag these elements to the aliases section and rename them in English as they display on the UI.

# Categorize the elements




- Drag a common element for all the elements and name it as "Field", "Button" etc.
- Drag and drop all the field elements under "Field".
- Drag and drop all the button elements under "Button".

# Scripts

Write scripts using the pseudonym names

a) Write a routine for each element
b) Routines for text fields use external datasheets
c) Log Messages, Warning, Errors should be written in the routine
    a) The first part of the routine refers to the page or window
    b) The second part of the routine refers to the type of element that needs to be used for the routine
    c) The third part of the routine refers to the element that will be used for the test
    d) The fourth routine will execute the actions

# Routine for User ID

```
Sub fUserID()

  Aliases.Gmail_Login.Field.UserID.Keys("^a")
  Aliases.Gmail_Login.Field.UserID.Keys("[Del]")
  Aliases.Gmail_Login.Field.UserID.Keys(DDT.CurrentDriver.Value("UserID"))
  Log.Message("UserID = " + Aliases.Gmail_Login.Field.UserID.value)

End Sub
```

- Routine Name = to suite the type and name of the element (should be easy to remember)
- The scripts start with the Aliases, Name (of page) Type (of element), Name of the element
- Data used for testing could be static (hard coded) or dynamic (driven from external data sheets)
- Messages & Warning are very important for any routine. (I like to log the Values from the elements rather than from the Data sheets.)

# Routine for Password

```
Sub fPassword()

  Aliases.Gmail_Login.Field.Password.Keys("^a")
  Aliases.Gmail_Login.Field.Password.Keys("[Del]")
  Aliases.Gmail_Login.Field.Password.Keys(DDT.CurrentDriver.Value("Password"))
  Log.Message("Password = " + Aliases.Gmail_Login.Field.Password.value)

End Sub
```

- Copy paste the UserID routine and replace the value “UserID” with “Password”.

# Routine for button Sign In

```
Sub bSignIn()
  Aliases.Gmail_Login.Button.SignIn.Click()
  Log.Message("Button Sign In Clicked")
End Sub
```

- Modify the script for Button Sign in, in a similar way or rewrite it.
- The scripts start with the Aliases, Name (of page) Type (of element), Name of the element
- Buttons are for clicking. So, change “Keys()” to “Click()”

# Routine for checkbox

```
Sub cbSignedIn()

    Aliases.Gmail_Login.Checkbox.StaySignedIn.ClickChecked(False)
    Log.Message("Checkbox Stay Signed In Checked OFF")

End Sub
```

- Copy and paste the Sign In routine. Remove the 2nd line.
- Replace the value “Button” with “Checkbox”.
- Replace the value “SignIn” with “StaySignedIn”.
- Replace the value “Click() with “Click(false)”.
- Insert a message to the log to suggest the check box Stay Signed In was checked ON. So add – **Log**.Message("Uncheck check box Stay Signed In")

# Routine to Verify Login

```
Sub VerifyLogin()
' Verify Login into Gmail
    If Not Aliases.Gmail.Exists Then
       Call Log.Picture(Aliases.Gmail, "Gmail Invoked", "Gmail Invoked", 300, 0,
-1)
       Log.Message("Gmail invoked via Mozilla Firefox")

    ElseIf Aliases.Gmail_Login.Exists Then
       Call Log.Picture(Aliases.Gmail_Login, "Gmail Login Failed", "Gmail Login
 Failed", 300, 0, -1)
       Log.Warning("Gmail Login Failed")

    Else
       Call Log.Picture(Aliases.MF.Firefox_Frame, "Gmail Not Invoked or
 Recognized", "Gmail Not Invoked or Recognized", 300, 0, -1)
       Log.Message("Gmail Not Invoked or Recognized")
     End If
End Sub
```

# Routine to combine steps

```
Sub LoginSteps()

    Call fUserID
    Call fPassword
    Call cbSignedIn
    Call bSignIn
    Call VerifyLogin

End Sub
```

- We need a routine that will keep all the actions (Routines) in one place
- This way we can call this routine as many times as we need

# Script for Login Routine

```
''' ############## LOGIN SCRIPT FOR GMAIL VIA MOZILLA FIREFOX ################
Sub Login()

    Set Driver = DDT.ExcelDriver("C:\SriluBalla\DataSheets\Login.xlsx", "Gmail",
 True)
    Driver.Name = "Login"
    Log.AppendFolder("Gmail LogIn with Browser Mozilla Firefox")

  If Not Aliases.Gmail_Login.Exists Then

       TestedApps.TerminateAll()
       TestedApps.firefox.Run()

     Aliases.MF.ToURL(DDT.CurrentDriver.Value("URL"))
     Log.Message("URL =  " + DDT.CurrentDriver.Value("URL"))

     If Aliases.Gmail_Login.Exists Then
        Call LoginSteps
     Else
        Call Log.Picture(Aliases.MF.Firefox_Frame, "Unknown Error", "Unknown
 Error", 300, 0, -1)
        Call Log.Error("Gmail Log In Not available")
        Call Runner.JustStop(False)
     End If

      If Not Aliases.Gmail.Exists Then
         Call Log.Picture(Aliases.MF.Firefox_Frame, "Unknown Error", "Unknown
 Error", 300, 0, -1)
         Call Log.Error("Gmail page In Not Available or Log in Failed")
      End If

  End If
  Log.PopLogFolder()

 DDT.CloseDriver(Driver.Name)

End Sub
```

# Run the final Script

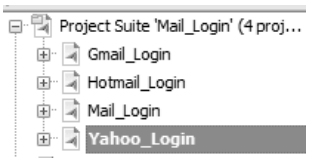

- Actions performed
  - User Name is typed in
  - Password is typed in
  - Check box is check ON (or OFF)
  - Sign in button is clicked

# Verify the Log file

| Type | Message | Time | Priority | Has Pict... |
|---|---|---|---|---|
| ⊟ | Gmail LogIn with Browser Mozilla Firefox | 11:1... | Normal | |
| | The application "C:\Program Files (x86)\Mozilla Firefox\firefox.exe" started. | 11:1... | Normal | |
| | URL = www.gmail.com | 11:1... | Normal | |
| | UserID = Srilu.Balla | 11:1... | Normal | |
| | Password = | 11:1... | Normal | |
| | The check box is already unchecked. | 11:1... | Normal | |
| | Checkbox Stay Signed In Checked OFF | 11:1... | Normal | |
| | Button Sign In Clicked | 11:1... | Normal | |
| | Gmail Invoked | 11:1... | Normal | |
| | Gmail invoked via Mozilla Firefox | 11:1... | Normal | |

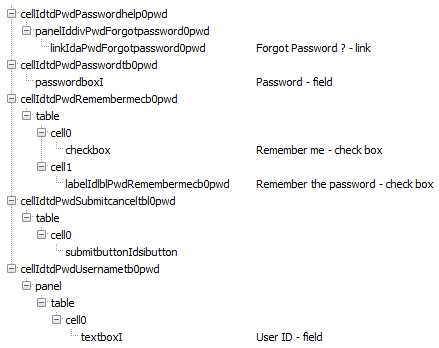
- URL used for the test
- User ID used for the test
- Password = (not displayed for privacy purpose)
- Checkbox unchecked
- Sign in button clicked on
- Gmail invoked via Mozilla Firefox

# Use the Gmail Login script to log into Hotmail



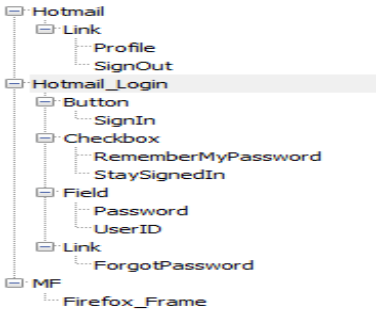

Create another project suite and name it Hotmail_Login

# Create Repository for Hotmail



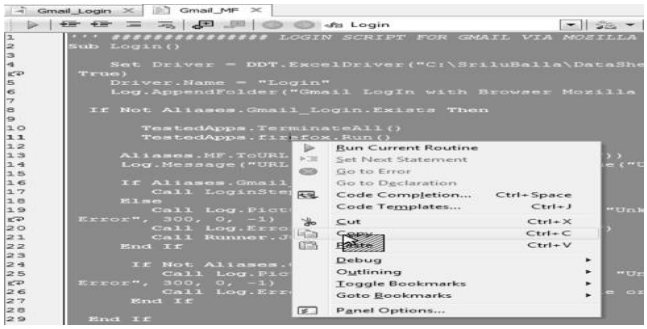

Capture elements

# Rename the elements and categorize them



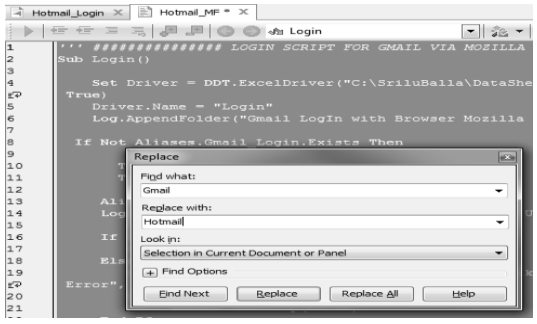

- Maintain the Hierarchy as is.

## Copy & Paste the Login Script for Gmail




- Open a blank script and name it Hotmail_Login.
- Copy paste the whole Gmail Login script

## Replace “Gmail” with “Hotmail”

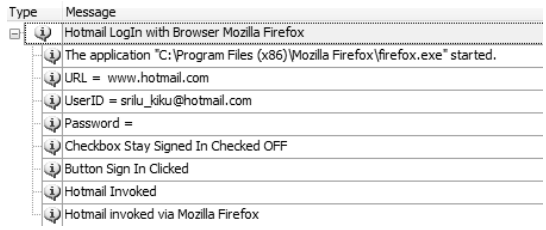



Call ‘Find and Replace (Ctrl+H)’ and replace word – “Gmail” with “Hotmail”

# Verify the replaced text & Run the Script


- Verify the replaced text and run the script.

# Verify the results




We have a successful run

# Use the Gmail Login script to log into Yahoo

- Create new Project
- Capture necessary elements to the repository
- Add the elements to the Aliases
- Rename the elements
- Add them to categories
- Copy paste the Gmail Login script to the new script
- Replace word – “Gmail” to “Yahoo”
- Verify the changes and run the script.

# Tool created Script

```
Sub Test1

  Set firefox = Aliases.firefox
  Call firefox.ToUrl("https://www.google.com")
  Set table = firefox.pageGmailEmailFromGoogle.table.cell.panelLogin.formGaiaLoginform.panelGaiaLoginbox
.tableFormNoindent.cell.panelLoginbox.tableGaiaTable
  Set textbox = table.cell.textboxEmail
  Call textbox.Click(81, 5)
  Call textbox.SetText("som")
  Call table.cell1.passwordBoxPassword.SetText("som")
  Set page = firefox.pageGmailEmailFromGoogle
  Set table = page.table.cell.panelLogin.formGaiaLoginform.panelGaiaLoginbox.tableFormNoindent.cell.panelLoginbox
.tableGaiaTable
  Call table.cell2.checkboxPersistentcookie.ClickChecked(True)
  table.cell3.submitbuttonSignin.Click
  page.Wait

End Sub
```

- The above script was created by recording actions
- If the original DOM is not renamed and categorized, a developer will need to painstakingly assume the next element that needs to be used to get to the desired element.
- If the DOM properties or names were changed, the script will also need to be adjusted along with the Name Mapping

# Green Lantern Script

```
Sub MailLogin()
    Aliases.MF.ToURL("www.gmail.com")
     Log.Message("URL =  " + "www.gmail.com")

  Aliases.Gmail_Login.Field.UserID.Keys("Som")
  Log.Message("User ID =  " + "Som")

  Aliases.Gmail_Login.Field.Password.Keys("Som")
  Log.Message("Password =  " + "Som")

  Aliases.Gmail_Login.Checkbox.StaySignedIn.ClickChecked(False)
  Log.Message("Check box - Stay Signed In = Checked OFF")

  Aliases.Gmail_Login.Button.LogIn.Click()
  Log.Message("Clicked button Login")

End Sub
```

- The above script was created with renamed aliases
- A developer does not need to assume the next element that needs to be used to get to the desired element. They know exactly which element they require to complete the routine.
- If the DOM properties or names were changed, the script does not need to be adjusted. Only the Name Mapping needs to be adjusted

# Conventional Automation Limitations

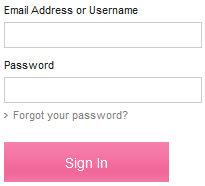
- Most automation is record and play. This technique will execute a few steps on an application, but won't log full length messages, warnings and won't do comparisons.
- A lot of time goes into automating regression suits. A lot of effort goes into automation. A lot of ideas go into automation. All of this effort goes waste with a few tweaks to the UI.
- When several developers (from all over the world) developing scripts for automation, unless there was a coding standard defined, each developer would pursue their own style of repository naming and script writing. The automation suits will not be consistent.
- Automated suits with various coding standards are hard to merge and maintain.
- Automated scripts created for one application cannot be used if changes are made to the application. They will need to be updated as well.
- A test had failed due to recognition issues, by looking at the logical names in the repository or the script alone it is not enough to identify which element needs updating. Because sometimes several elements could be named same and several times the names given on the page are not used for the logical name.

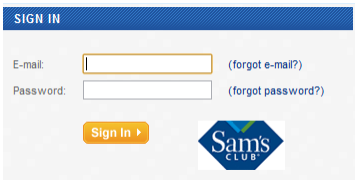
# Summary

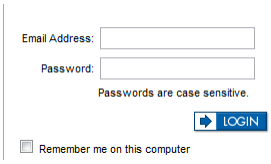
Archetypical Green Lantern DOM

- Page
  - Type
    - Element
- Gmail_Login, Yahoo_Login, Hotmail_Login
  - Field, Button, Link
    - UserID, Sign In, Forgot Password

# Many UI's with similar function

# Conclusion

Typically each application has its own team of testers and engineers to come up with scenarios for testing and automating.
With the help of Green Lantern Framework the testers time can be freed up to explore and implement more scenarios across applications and UI.

Companies that focus on testing can come up with test scripts for similar applications or domains –
Suitable Test cases, Automated scenarios and the data files with all possible Data are ready to be used for manual or automated testing.

# THE END

www.linkedin.com/in/**sriluballa**
http://sriluballa.wordpress.com
Sridevi.Pinjalaa@Gmail.com