# Telerik SQA Track 2015

# Exam #1 Preparation

## Bug Reports

You are given one video showing bug in the Telerik's ASP.NET FileExplorer functionality. Your task is to watch them, navigate to the URL shown in the video and prepare bug reports.

* Identify the bug
* Reproduce it
* Write bug report

### ASP AJAX control bug

Watch the [video](http://screencast.com/t/ymyJH35zc8VF).

Go to <http://demos.telerik.com/aspnet-ajax/fileexplorer/examples/asyncupload/defaultcs.aspx> and try to reproduce the bug from it.

Prepare a **bug report** containing all **necessary information** for logging the report in a bug tracking system. Try to **isolate** the minimum steps required for reproducing the bug, removing any steps that are not really influencing the behavior.

The **reports** should be presented in a **table in word** document.

## Automation Tests

### Magento Incorrect Login Test

Navigate to the following URL: <http://demo.nostresscommerce.cz/>

1. Click the **My account** link. Try to login with incorrect e-mail and password. Verify that the login is rejected.
2. **Make another test**, but this time try to login without entering values in the e-mail and password fields. Verify that the user is prompted to fill in the empty fields.

### Random Generator Test

Navigate to the following URL: <http://www.random.org/integers/>. Make a test that generates a random number between 1 and 2. If the number generated is 1 – the test navigates to the Frequently Asked Questions (FAQ) page (you’ll find a link in the **Learn more** menu), verifies that the correct page is loaded and navigates to the home page of the site.  
If the number generated is 2 – the test should navigate directly to the home page.  
Verify that the home page is loaded.

### BugTracker Test

Make functional web tests using Selenium IDE, Selenium Web Driver or Telerik Test Studio (as long as the tool provides the functionality you would need for the test). The tests should navigate to the following URL: <http://ifdefined.com/btnet/bugs.aspx>

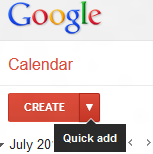
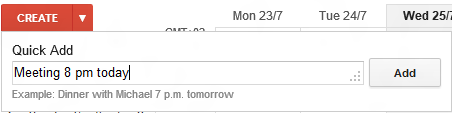
1. Choose a number, so that there could be no existing bug with that ID.   
   Try searching for a bug with that number and verify that the site responds correctly to that search.
2. Go to the page for creating a new bug with the “**add new bug**” link.  
   The form for creating a new bug has a drop-down list named **Project-specific** that is visible only if the “**HasCustomFieldsProject**” option is chosen in the **Project** drop-down list. Make a test that verifies that this functionality is applied correctly.

### Google Calendar

Make functional web tests using **Selenium IDE**. The tests should navigate to the following URL: <http://www.google.com/> .

Prerequisite: Create an account in google which can be used for tests.

1. Log into a valid google account and verify that the log in was successful and the correct user is logged in.
2. While you are logged and your calendar is shown create an event using the quick add option of the create button. In the text field write the following text “Meeting 8 pm today”. Verify that there is an event today that has such title, and then delete the event.



1. Take the test above and instead of creating the event with title Meeting, create it with unique heading every time using JavaScript.(Tip: You can generate your unique name in separate variable)

**Web driver**

* Convert the tests to C# and test if the backend e-mail validation works properly. Think of an appropriate way to replace your JavaScript functionality (if any). Use different user profiles and run the test with different browsers.

### Yahoo Test

Make functional web tests using **Selenium IDE**.

The tests should navigate to the following URL: <http://www.yahoo.com/>.

Prerequisite: Create an account in Yahoo which will be used for tests.

Record the following tests:

1. Log into a valid Yahoo account and verify that the **log in** was successful and the correct user is logged in
2. While you are logged in make a test that **sends email** to you. In the message subject write the following text "QA Exam #1"
3. Make sure that the message is sent using the information about **Sent** messages

* Verify that the field **To** contains your email address
* Verify that the field **Subject** has title "QA Exam #1"

1. Take the test above and instead of writting "QA Exam #1", create it with an unique subject using JavaScript. (Tip: You can generate an unique name in separate variable)

### jqGrid Test

Make functional web tests using **Selenium IDE**.

Navigate to the following URL: <http://www.trirand.net/demo/aspnet/webforms/jqgrid/> and select Hierarchy -> Sub Grid (2 nested levels) from the left menu. That will open the jqGrid that we are going to use. (Note that you should follow these steps every time you make a test).

1. Verify that the table headers are exactly (ID, CompanyName, ContactName, Phone and City)
2. Expand the row with ID="ANTON" and verify that the CompanyName value from the expaned row is equal to ShipName value in the the first row of the inner jqGrid.

**Selenium WebDriver**

Convert the test cases (A, B) from the task to C# and test if they work properly. Run the tests a couple of times to verify that they work correctly.

Make functional web test using Selenium WebDriver to test paging functionality of the jqGrid. From the combo box select **consequently 10, 20 and 30** and verify that changing the page count is working correctly i.e. when 10 is selected, the rows in the **main table** should be exactly 10 etc. Be careful to count only these rows that are visible in the table.

## Selenium Web Driver

You are the only QA Engineer who tests the Telerik Academy Learning System and your task is to make a test to apply for Software Academy using Selenium Web Driver.

Environment:

* Testing instances of the real system - <http://stage.telerikacademy.com/>
* Data base without application stored

Scenario:

* Go to <http://stage.telerikacademy.com/> and log in.
* Apply for Software Academy (“Софтуерна академия” -> “Кандидатстване за академията”). Fill all required fields to make your application successful.
* Go to Candidates Administration (Admin panel - > “Кандидати” modul). Verify that your application is stored, using your unique information.
* Delete your application and verify that it is deleted.

Think of an appropriate way to organize your tests and use

* + **Page Object Model**
  + **Base test class** with methods missing in selenium WebDriver

Hint: Always delete your application manually or automatically to know that the table is empty!

### DataFilterControl

Make functional web tests using **Test Studio**. The tests should navigate to the following URL: <http://demos.telerik.com/silverlight/>.

* Navigate to all controls, chose DataFilter and verify you are on the correct page.
* Add a filter to the grid and verify it is applied correctly.
  + Apply filter “Quantity is equal to 9”. Each row, shown in the table should have value 9 in column Quantity. Use coded step for that validation. Find parent element of type RadGridView. With that element you can reach rows and cells. In order to have proper Assertion extract the values from the other two filter fields and think how to use them. The assertions should look like that Assert.AreEqual(extectedValue, actualValue). If you need to print something to the Log you can use Log.Writeline() command. Do not forget to use the Add to Element Explorer option.
* Add a data source to the test above. Use the data below. Locate your elements very carefully.

|  |  |  |
| --- | --- | --- |
| **Field** | **Expression** | **Value** |
| **Quantity** | Is equal to | 9 |
| **Name** | Starts with | P |
| **UnitPrice** | Is less than | 50 |

### Test Studio Stand Alone

Make functional web tests using **Test Studio**.

Download the project [here](http://tinyurl.com/qa-academy-test-studio):

Run the project following this steps:

1. Unzip the project
2. Open Visual Studio and choose File -> Open -> Web Site
3. Navigate to the unzipped project
4. Run the project

Once you run the project record the following tests:

* Create a new record.
* Add a data source to the test above for Product Name and Unit Price value. The data source should look like this:

|  |  |
| --- | --- |
| Product Name | Unit Price |
| Product1 | 1 |
| Product2 | 2 |
| Product3 | 3 |

* Filter the products by Product Name *Contains* “Product”.

Use coded step for the folowing validations:

* + Vetify that the **Count** in the footer shows 3.
  + Verify that the **First** element in the footer displays the first row’s Product name.
  + Verify the products are filtered by checking if the sum of the unit prices of the three products equals the **Sum** of the footer.
* Verify that the editing and deleting is working correctly. Use coded step(s) for clicking (locating) the elements and validations.

Hint: Using **Reset Date to its initial state** button might be useful.

## Test Studio

1. Make functional web test using Test Studio. The test should navigate to the following URL: <http://demos.telerik.com/kendo-ui>. Verify the existence of Grid control demo and open it.

Go to Inline Editing page of the demo (use the navigation links on the left).

* Verify that:
* The grid is loaded
* The buttons (except for the pager control) exist and check their text
* The Pager control is enabled
* The count of the grid rows
* Verify the “Go to Last Page” functionality of the Pager control.

1. Make a second functional web test. It should navigate to the following URL: <http://test.telerikacademy.com>.
   * Make a negative login test that provides wrong credentials. Verify the validation message.
   * Now provide a valid credentials (your own) and Login. Verify that the correct page is loaded and that the correct account logged in.

Hint: If you wish to hide your passwords, there is a way to encrypt in the recorded step.

* + Now use the two steps that provide user and password to login with two different users – yours and academy/telerik. The logged user verification step should be also adjusted so it verifies the correct logged user.

1. Create a new web test that uses as first step the test from task 1. Make sure that you are on the first page of the Grid.
   * Verify in **one step** that the first column of the first four rows exists and in another step (coded) write its text content in the execution log.

Hint: The find expressions of these four columns are very similar with each other. Use element find expression data binding.

Hint: If you have problems with an element, don’t forget that you can always change its find expression.

When navigating through pages, don’t forget to make sure that the right content is loaded.