Key Points in an interview:

**3.** [**Lack of user input**](https://www.finextra.com/blogposting/6836/7-reasons-why-software-development-is-so-hard)

For over 10 years the research company, The Standish Group, have surveyed companies about their IT projects. The No. 1 factor that caused software projects to become challenged was "Lack of User Input".

1. Given an Amazon Shopping Cart with 5 items, provide 10 test cases for each of the following test areas:
   1. Functional (remove an item, proceed to checkout, etc…)
   2. UI (buttons, text boxes, links, etc…)
   3. Pick your favorite that is not #1 or #2

1. What would you include in a test plan for the above Amazon shopping cart?
   * 1. Give a brief summary with a targeted purpose for each item you would include.

1. Answer at least 2 of the following questions:
   * 1. What test exclusions would you call out in your test plan?
   1. What is the responsibility of the developers in your test plan?
   2. What Security testing would you do in the shopping cart?
2. Given an Amazon Shopping Cart with 5 items, provide 10 test cases for each of the following test areas:
   1. Functional (remove an item, proceed to checkout, etc…)
3. **Start by removing all 5 items. Confirm removal is successful.**
4. **Add a Single Product to Cart**
5. Select a product.
6. Select Add to Cart.
   1. Verify that the correct product display page - PDP - is showing all necessary links and information.
   2. Verify product successfully added to cart and in the Cart icon.  Only 1 item is displaying.
7. **Added multiple Products to Cart**

1. Select a product.

2. Select Add to Cart – add the First Product.

3. Select Add to Cart – add Second Product.

1. Verify that the correct product display page - PDP - is showing all necessary links and information.

2. Verify product successfully added to cart.

3. Verify two products are showing in the cart and in the cart icon, 2 should be displaying.

1. **Validate Cart after editing Quantity**

1. Select a product.

2. Select Add to Cart – add the First Product..

3. Select Add to Cart – add Second Product.

4. Editing the quantity of items in the cart with valid and invalid quantities (like spaces, negative values, characters, 0, etc).

1. Verify that the correct PDP page is displaying all the necessary

links and information.

2. Verify product is successfully added to cart.

3. Verify the items quantity is successfully edited. If user gives a -ve value displayed a warning message to user.

1. **Cart – Validate Backend sync**

1. Select a product with an inventory count that is less than 5 (to be validated at backend)

2. Select Add to Cart icon to add the Product.

3. Edit the quantity of the product in cart with a 5 and attempt to checkout.

1. Verify that the correct product display page - PDP - is showing all necessary links and information.

2. Verify product successfully added to cart and in the Cart icon.

3. Verify if items quantity count is greater than the actual item’s inventory count then checkout should prevent the user from proceeding and the proper error should be presented to the user.

1. **Remove a single product from Cart**

1. Select a product.

2. Select Add to Cart.

3. Select the Cart.

4. Select the Cross symbol to remove the product from the cart.

1. Verify that the correct product display page - PDP - is showing all necessary links and information.

2. Verify product successfully added to cart and in the Cart icon.  Only 1 item is displaying.

3. Verify the product was added and it’s displaying correctly in the cart

4. Verify the Product is removed from cart and Cart icon is displaying 0 items.

1. **Remove multiple products from Cart**

1. Select a product.

2. Select Add to Cart – add the First Product.

3. Select Add to Cart – add Second Product.

4. Select the Cross symbol to remove any and all of the product from the cart.

1. Verify that the correct product display page - PDP - is showing all necessary links and information.

2. Verify product successfully added to cart and in the Cart icon.  Only 1 item is displaying.

3. Verify both products are showing in the cart and that the Cart Icon is displaying 2 items only.

4. Verify the product has been removed from the cart and Cart icon is displaying 1 items.

1. **BuyNow**

1. Select a product.

2. Select the Buy Now option.

1. Verify that the correct product display page - PDP - is showing all necessary links and information.

2. Verify Product purchase page is showing all available payment options should to the user.

1. **Guest Checkout**

1. Launch the test app - Don’t sign in.

2. Select a product to buy.

3. Now sign-in and Checkout.

1. Validate the test app is successfully launched and a msg of Hi Guest is clearly displaying at for the user.

2. Verify the correct PDP page includes all the payment options should showing to the user.  The app should ask the user to login.

3. Verify user can successful checkout.

1. **Validate CheckOut for a Registered user**

1. Start the test app.

2. login.

2. Select a product to buy.

3. Checkout.

1. Verify test app is successfully launching and that Hi, Username is dispalying.

2. Verify that the correct PDP page with all the payment options are shown to the user.

3. Validate user is able to successful checkout.

1. **Gift Coupon Validation**

1. Select a product which allows a Gift Coupon.

2. Confirm the Coupon is valid while checking out.

1. Verify Gift Coupon is valid and that it’s code is tagged to the product.

2. Confirm that the amount of the Gift Coupon is successfully deducted at checkout.

* 1. UI (buttons, text boxes, links, etc…)

1. Make sure the **buttons** are all there
2. Make sure all the **text** **boxes** are there
3. Make sure all the **colors** are correctly displayed on various devices.
4. Make sure no **spelling** mistakes
5. Make sure all **links** are valid
6. Make sure **text boxes** are able to lower and upper case numbers
7. Make sure all **images** are shown correctly
8. Make sure all **tabs** are working
9. Make sure **scroll** on page works
10. Make sure **zoom** and zoom out is working
11. Make sure **color**-blind people are able to navigate site
12. Make sure **page titles** are correctly displayed.
13. Emails on page are working.

* 1. Pick your favorite that is not #1 or #2

(This isn’t really clear. Does this mean not functional nor UI? If so below is my answer)

**Performance (**how well a software can handle user traffic**)**

I would test the performance of the website. The website needs to load quickly to retain customers otherwise customers will get frustrated and move to other sites. There are many tools to use to test the performance of a website.

Breaking it down to:

* + **Load testing:** how quickly does it load when one then 15 then 1000 then 1000 ..etc users visits the website
  + **Stress testing**: to challenge the limits of the application e.g. if you get a billion users! (which is outside the sites limits) What happens! Crash?!?

Some of those performance tests are:

1. Have 100 users press **add** item at the same time.
2. Have a 100 users press **change** amount at the same time
3. Have a 100 users press **buyNow** at the same time
4. Have a 100 users press add item and leave it for a week without buying and come back to it.
5. Have a 100 users press **purchase** at the same time. (buyNow is different from purchase)
6. Have a 1000 users **enter their credit card information** and press next at the same time
7. Have a 10000 users **cancel** their purchase at the same time.
8. Have a 100 users enter **the same coupon** at the same time
9. Have a 1000 users **proceed to checkout** at the same time.
10. Have a 1000 users **order the same product** (which you only have one item of) at the same time.
11. **Put this test in a loop**:
12. Add an item to the cart
13. Then Browse for another item
14. Add s second item to the cart
15. Go to checkout
16. Complete the checkout process

Then watch for performance bottlenecks. This will help tune and remove bottlenecks.

1. What would you include in a test plan for the above Amazon shopping cart?
   1. Give a brief summary with a targeted purpose for each item you would include.

**\*\* Test Plan**

**QA**/**QC** is the combination of

**quality assurance**, the process or set of processes used

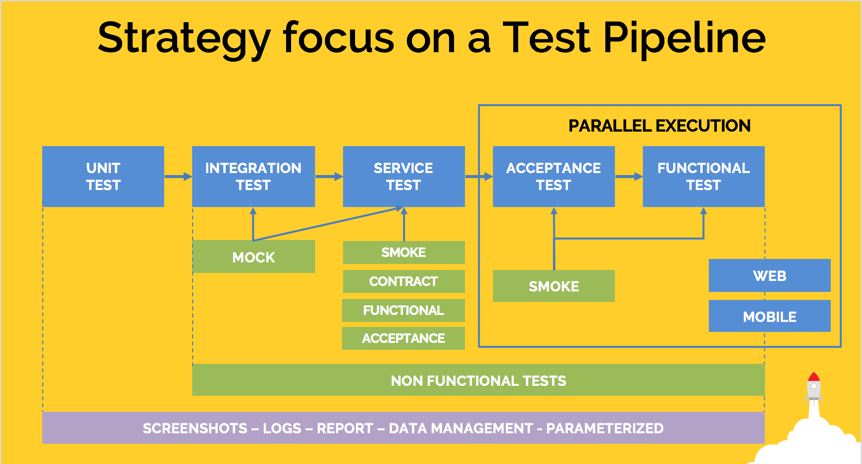
\* to measure

\* and assure the **quality** of a product,

and **quality control**,

the process of ensuring

products and services meet consumer expectations.



Basics of Testing an e-Commerce Website:

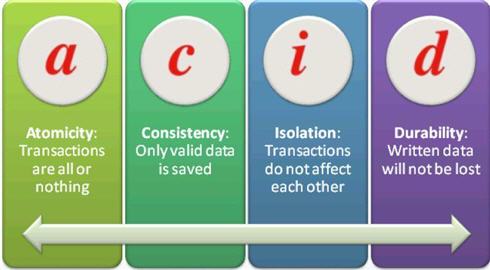
<https://dzone.com/articles/8-cases-to-skyrocket-your-e-commerce-website-testi>

e-commerce sites and applications or mobile applications undergo:

* A/B testing
* Usability Testing (User Acceptance Testing - UAT) -
* Database Testing
* [UI Testing](https://www.testing-whiz.com/blog/how-to-overcome-ui-automation-testing-challenges?utm_source=Quora&utm_medium=QnA&utm_campaign=Rachana&utm_content=Content)
* [API Testing](https://dzone.com/articles/10-effective-ways-for-successful-api-testing?utm_source=Quora&utm_medium=QnA&utm_campaign=Rachana&utm_content=Content)
* Mobile Application Testing
* Security Testing
* Performance Testing
* Load Testing
* [Functional Testing](https://www.testing-whiz.com/blog/6-essential-factors-for-successful-functional-test-automation?utm_source=Quora&utm_medium=QnA&utm_campaign=Rachana&utm_content=Content)

**In Details**

* A/B testing (**split testing** or **bucket testing**) - is a method of comparing two versions of a webpage or app against each other to determine which one performs better.
* Usability Testing (User Acceptance Testing - UAT) - is a method by which users of a product are asked to perform certain tasks in an effort to **measure** the product's **ease-of-use**, **task time**, and the **user's perception** of the experience.
* Database Testing –
  1. Data Mapping:
     1. are the UI fields mapped to the data fields?
     2. Correspondence between UI actions and [CRUD (Create, Retrieve, Update and Delete)](https://www.softwaretestinghelp.com/database-testing-process/)
  2. ACID properties validation:
     1. **Atomicity**
     2. **Consistency**
     3. **Isolation**
     4. **Durability**

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2013/08/DB-Testing.jpg)

* 1. Data integrity
  2. Business rule conformity:
* [UI Testing](https://www.testing-whiz.com/blog/how-to-overcome-ui-automation-testing-challenges?utm_source=Quora&utm_medium=QnA&utm_campaign=Rachana&utm_content=Content) GUI testing
* [API Testing](https://dzone.com/articles/10-effective-ways-for-successful-api-testing?utm_source=Quora&utm_medium=QnA&utm_campaign=Rachana&utm_content=Content) API testing is a type of software testing that involves testing application programming interfaces directly and as part of integration testing to determine if they meet expectations for **functionality**, **reliability**, **performance**, and **security**. Since APIs lack a
* Mobile Application Testing
* Security Testing
* Performance Testing
* Load Testing (such as load runner, Visual Studio Enterprise edition,

Several types of load testing are employed

* Static testing is when a designated constant load is applied for a specified time.
* Dynamic testing is when a variable or moving load is applied.
* Cyclical testing consists of repeated loading and unloading for specified cycles, durations and conditions.
* [Functional Testing](https://www.testing-whiz.com/blog/6-essential-factors-for-successful-functional-test-automation?utm_source=Quora&utm_medium=QnA&utm_campaign=Rachana&utm_content=Content)

Before listing the test plan it’s important to understand the big picture, i.e the funnel of an e-Commerce website:

The traditional e-commerce funnel has 5 stages:

1. Acquisition: This is the first step. It’s were the user visits an e-commerce website
2. Engagement: The user is browsing the product(s) or service(s) catalogue, and they start evaluating existing options. Thereafter the user starts to consider buying by adding items to the shopping cart
3. Activation: The user is now set on purchasing so the user checkouts the desired items and makes the payment.
4. Retention: Is when a user is back at the website the user visits the e-commerce website or app and shops again.
5. Referral: The customer/user is satisfied with that the post purchasing service was good enough to recommend. This means the user will return to the website and will advocate for the website.

Make sure Clicks, registrations, payments, subscriptions all works.

**Requirements**

User should be able to add, remove, and change quantities of the items in their cart.

User should be able to get back to item pages from the cart by clicking on individual cart item lines.

User should be able to put in their address information to get shipping information estimates.

User should be able to add coupons, promocodes which and website should confirm if codes are expired or not.

Pay attention to cases such as logging in (if user has items in their cart as an anonymous user, but also have cart items from a previous authenticated session), or adding another item of the same SKU as others in their cart.

**Shopping Cart Test Plan**

**Pages of the Product category**

**If user clicks on any product in the basket should be taken to product. In the product:**

* Filters such as product or items in basket filters, colors, sizes, types of product, etc.
* Sorting products or items in basket by name, price, size, etc.
* Add/Remove form basket to a shortlist or wish list facility.
* Add/Remove from basket.
* Tools tips work.
* Drop down menus work
* Select/deselect
* Scrolling? Auto scroll?
* Hover function?
* Rendering on different browsers and different resolutions.

**Shopping basket**

* Add product(s) to shopping cart.
  + - * Item should go added
    - Empty Cart
      * When I click add from a product page
      * Then the cart should show a new line item
* Product Info images ..etc displayed correctly.
* Links between product in cart and product page works.
  + - * If cart has an item
      * Clicking the item link should take user to item page
* Change quantities of product in shopping cart.
  + - * Then quantity should update with new quantity
      * New price should show
      * Decrease quantity below 0
      * Use should be warned
      * When user adds another item of the same sku
      * User should have 1 cart line item with quantity increased
      * User should not have 2 copies of the same item in the cart
* Remove product from shopping cart.
  + - * Item should go deleted
      * When user clicks on an item link for a deleted item (sad path)
      * User should get a 404 page or proper error msg
* Check error messages if no items are in shopping cart.
* Check discounts, taxes and delivery costs are correct etc
* Min order works with free delivery
* Subtotal adds accurately.
* Valid discount code added
* Invalid discount code added
  + - * Given user can view the cart
      * When user input a coupon containing a SQL injection
      * Then the database should remain unpwned
* After purchase correct links, return policy …etc is displayed.
* Click Checkout or Pay Now to process to Checkout process.
  + - Pay attention to cases such as logging in (if user has items in their cart as an anonymous user, but also have cart items from a previous authenticated session), or adding another item of the same SKU as others in their cart.
    - Pay attention to cases such as logging in (if user has items in their cart as an anonymous user, but also have cart items from a previous authenticated session), or adding another item of the same SKU as others in their cart.

**Social Media**

Given an item listed in Social Media are you able to click on it and be taken to the site/app and the cart lists the item(s).

**Mobile Device Compatibility**

Does it work on mobile devices (compatibility)?

* Various phones
* Various browsers on phones
* Apps on different phones

**Browser Compatibility**

Various browsers testing is essential:

* Internet Explorer, MS Edge
* Mozilla Firefox (latest version)
* Google Chrome (latest version)
* Safari

**Checkout and Payment Systems**

Check the payment system of the products selected.

Process:

* Guest checkout works.
* Existing customers able to login.
* Existing customers can login through social networks.
* Users can register a new account.
* Delivery address details can be entered.
* Alternative billing address are available and can be entered.
* User able to select payment method.
* Final payment amount is correct, after the taxes coupons …etc
* Test making changes to the products being ordered, changing delivery options, etc. and make sure that this final amount updates correctly.

**Payment System**

Test payment using each payment method that are being offered such as debit cards, credit cards, Paypal, Google Checkout, etc.

An example list of payment types is as follows:

* Place Paypal payment
* Place Visa payment
* Place Visa Debit payment
* Place Visa Electron payment
* Place Mastercard payment
* Place Amex payment
* Place false payment
* Test cancelling order
* If a user is signed in for a long time, ensure there is a session timeout. All websites have different thresholds.
* Message/email confirmation with the order number that is generated after the purchase is made.
  + - The Credit Card
  1. check the credit card types allowed
  2. check the expiration dates of the card whether adding or
  3. updating the card info
  4. check the credit card billing address whether the address is
  5. updated or the card is updated

**Shipping:**

* Shipping address information

If user did NOT input address information

When user clicks to check out

User should be asked for address information

If user had no input address information

When user views the cart

Then there should not be a shipping estimate

If a user didn’t input address information

When user input address information

Then user should see a shipping estimate

**Login Handler:**

Given user not logged in

When user add an item to cart

Then user should be able to view the item in my cart

Given a user is not logged in

When user adds an item to cart

And user log in

And user add another item to the cart

Then user should see both items in the cart

Given user is logged in

When user add an item to the cart

And user log out

Then user should not see the item in the cart

**Check Analytics**

Make sure check analytics are installed for the shopping cart.

1. Answer at least 2 of the following questions:
2. What test exclusions would you call out in your test plan?
3. What is the responsibility of the developers in your test plan?
4. What Security testing would you do in the shopping cart?

a. What test exclusions would you call out in your test plan?

1. Exclude old OS systems
2. Exclude unpopular or unsupported browsers
3. Exclude unpopular or unsupported phones apps
4. Exclude unpopular or unsupported phones browsers
5. Feature that are not included in this version of the software
6. Feature that are low-risk
7. Features that are not impacted by current changes
8. products that are no longer supported
9. Products that have already been tested and not selling.

b. What Security testing would you do in the shopping cart?

[Security Testing Test Scenarios URL](https://www.softwaretestinghelp.com/sample-test-cases-testing-web-desktop-applications/)

1. Check for SQL injection attacks.

2. Secure pages should use the HTTPS protocol.

3. Page crash should not reveal application or server info. Error page should be displayed for this.

4. Escape special characters in the input.

5. Error messages should not reveal any sensitive information.

6. All credentials should be transferred over an encrypted channel.

7. Test password security and password policy enforcement.

8. Check application logout functionality.

9. Check for Brute Force Attacks.

10. Cookie information should be stored in encrypted format only.

11. Check session cookie duration and session termination after timeout or logout.

11. Session tokens should be transmitted over a secured channel.

13. The password should not be stored in cookies.

14. Test for Denial of Service attacks.

15. Test for memory leakage.

16. Test unauthorized application access by manipulating variable values in the browser address bar.

17. Test file extension handing so that exe files are not uploaded and executed on the server.

18. Sensitive fields like passwords and credit card information should not have to autocomplete enabled.

19. File upload functionality should use file type restrictions and also anti-virus for scanning uploaded files.

20. Check if directory listing is prohibited.

21. Password and other sensitive fields should be masked while typing.

22. Check if forgot password functionality is secured with features like temporary password expiry after specified hours and security question is asked before changing or requesting a new password.

23. Verify CAPTCHA functionality.

24. Check if important events are logged in log files.

25. Check if access privileges are implemented correctly.

Penetration testing test cases – I’ve listed around 41 test cases for penetration testing on this page.

1. ~~Use stolen cc numbers~~
2. ~~Use false names with cc no~~
3. ~~Use false cc nos~~
4. ~~Use wrong addresses with cc~~
5. ~~Use wrong zip code with cc no~~
6. ~~Make part of the address to be wrong~~
7. ~~Make part of the name to be wrong~~
8. ~~Use cc nos less than what a American express would have~~
9. ~~Select American express for a cc and enter numbers for a visa cc~~
10. ~~Make part of the zip code to be wrong~~
11. SQL Injections,
12. Ethical hacks on Login,
13. Register,
14. Payment gateway, and other various pages.
15. Cookies Security settings Testing
16. Registration:
    * 1. Account:
         1. Unique user ID
         2. Each user has a unique account
         3. Passwords
            1. Are they stored encrypted?
            2. Are they encrypted in user logs?
17. SSL
    * 1. Is the registration process secure?
      2. Can user break out from registration?
      3. SSL adds numerous features to that stream, between two computers or processes that are exchanging data. The TCP/IP protocol simply sends anonymous error-free stream. SSL encrypts that data exchange.
         1. Authentication and non repudiation of the server, using digital signatures
         2. Authentication and non repudiation of the client, using digital signatures
         3. Data confidentiality through the use of encryption
         4. Data integrity through the use of message authentication codes
18. Cookies: What to test
    1. Test your site’s cookies according to the content of the cookie.
    2. Test with cookies and without.
    3. Test the expire date, domain, path and security setting.
    4. Test with Internet Explorer and other browsers
19. Customers
    1. What information is maintained about each customer?
    2. Is your site customized?
    3. What information is stored encrypted?
    4. What are the rules for sending emails?
20. Session
    1. Session timeout
    2. Unique session IDs
    3. Session caching
    4. User logs
21. Servers
    1. Server re-directs
    2. SSL
    3. Configuration files
    4. User logs

**\*\* Design Pattern**

[**https://www.geeksforgeeks.org/design-patterns-set-1-introduction/**](https://www.geeksforgeeks.org/design-patterns-set-1-introduction/)

**Types of Design Patterns**  
There are mainly three types of design patterns:  
**1. Creational**

**2. Structural**

**3. Behavioral**

**\*\* Selenium**

**Interview Questions**

[**https://www.youtube.com/watch?v=YWdAh3gTc6c**](https://www.youtube.com/watch?v=YWdAh3gTc6c)

**Selenium vs. Cypress: Is WebDriver on Its Way Out?**

<https://dzone.com/articles/selenium-vs-cypress-is-webdriver-on-its-way-out-1>

Cypress:  open source [test automation tools](http://crossbrowsertesting.com/automated-testing)

**Selenium**

**Selenium WebDriver is the open source *browser automation tool***

which interacts with browser and automate end to end tests of a web application.

**How many scripts execute in a day**

It depends on the scenario

Depends on the framework.

**Selenium Consist of:**

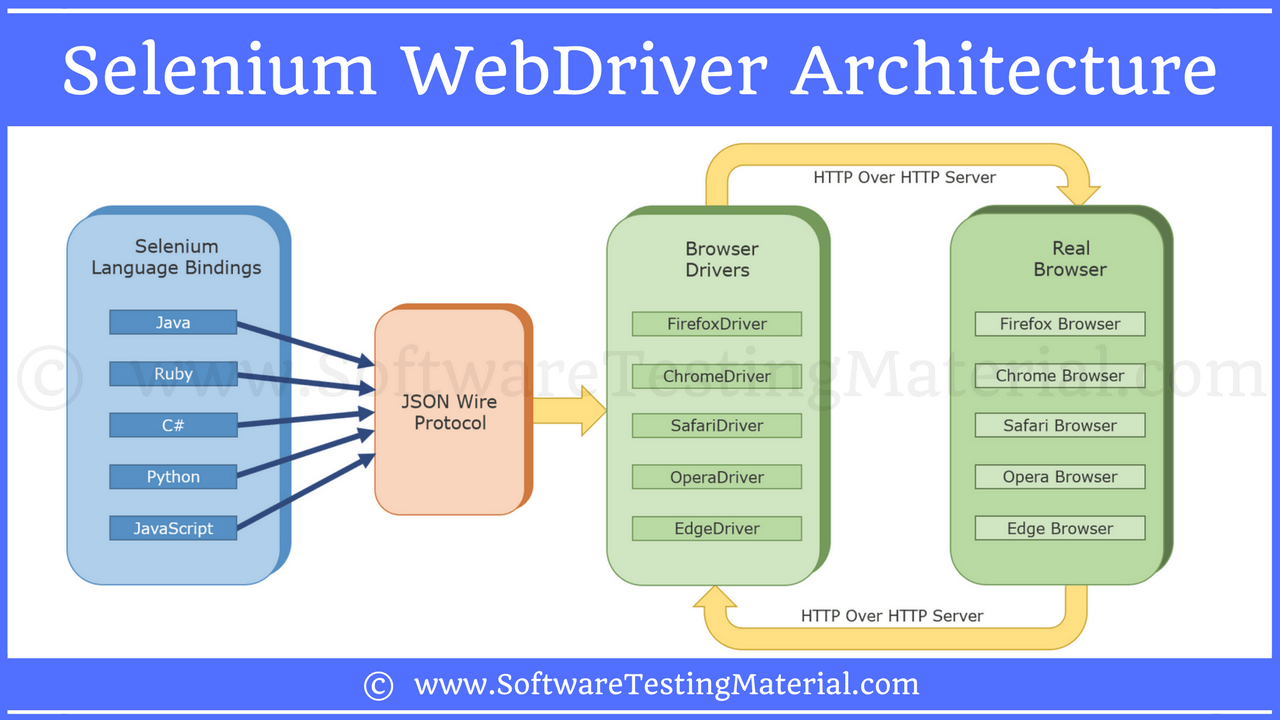
* + 1. Selenium IDE,
    2. ~~Selenium RC,~~ **~~officially deprecated in Selenium 2.~~**
    3. Selenium Webdriver,
    4. Selenium Grid

1. **Selenium IDE:** It allows us to record and playback the scripts. Even though we can create scripts using Selenium IDE, we need to use Selenium RC or Selenium WebDriver to write more advanced and robust test cases.
2. **Selenium RC,**
3. **Selenium Webdriver,** Selenium WebDriver is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser-specific driver. It controls the browser by directly communicating with it
4. **Selenium Grid** Selenium Grid is a **tool used** together with Selenium RC **to run tests** on different machines against different browsers in parallel.

**Selenium WebDriver Architecture**

There are four components of Selenium Architecture:

1. Selenium Client Library
2. JSON Wire Protocol over HTTP
3. Browser Drivers
4. Browsers



<https://wiki.saucelabs.com/display/DOCS/Getting+Started+with+Selenium+for+Automated+Website+Testing>

**There are seven basic steps in creating a Selenium test script, which apply to any test case and any application under test (AUT).**

1. Create a WebDriver instance.
2. Navigate to a Web page.
3. Locate an HTML element on the Web page.
4. Perform an action on an HTML element.
5. Anticipate the browser response to the action.

**Advanced Interview Questions**

<https://www.softwaretestingmaterial.com/test-automation-framework-interview-questions/>

1. **What are the Assertions and actions in Seleinum**

A **Selenese** tells Selenium what to do. Selenium commands (Selenese) are of three types : Actions, Accessors, and Assertions.

**Actions** generally manipulate the state of the application like “**click this link**” and “**select that option**”. If an Action fails, or has an error, the execution of the current test is stops.

**Accessors** examine the state of the application and store the results in variables, e.g. “**Title**”.

Assertions verify that the state of the application is same to what we are expecting. Selenium

**Assertions** can be of three types: “assert”, “verify”, and ” waitFor”.

When an “**assert**” fails, the test is aborted.

When a “**verify**” fails, the test will continue execution, logging the failure.

A “**waitFor**” command waits for some condition to become true. They will fail and halt the test if the condition does not become true within the current timeout setting. Perhaps, they will succeed immediately if the condition is already true.

When, we talk about the assertions used in WebDriver using TestNg framework, we have two types of assertions; hard assertion and soft assertion. Let’s see about them briefly:

[**Hard Assertion In Webdriver Using TestNg**](https://webkul.com/blog/assertion-in-selenium-webdriver/) **link**

* + 1. assertEquals
    2. assertNotEquals
    3. assertTrue
    4. assertFalse
    5. assertNull
    6. assertNotNull
    7. **assertEquals**

Assertion fails and is logged if what’s expected isn’t like what’s actual. The suite continues to run with the next @Test annotation(if any).

Assert.assertEquals(actual,expected);

* + 1. **assertNotEquals**

Assert.assertNotEquals(actual,expected,Message);

* + 1. **assertTrue**

if the condition is false/fails, this assertion skips the current method from execution.

Assert.assertTrue(condition);

* + 1. **assertFalse**

opposite to assertTrue

Assert.assertFalse(condition);

* + 1. **assertNull**

When an object is ‘null’ the assertion returns an exception resulting in aborting the test.

Assert.assertNull(object);

* + 1. **assertNotNull**

When a object has some value, the assertion aborts the method with an exception.

Assert.assertNotNull(object);

**Soft Assert**

import org.testng.asserts.SoftAssert;

Use softAssertion to overcome this drawback of hard assertion we can use soft assertions in testNg.

package hardAndSoftAssertion;

import org.testng.Assert;

import org.testng.annotations.Test;

import org.testng.asserts.SoftAssert;

public class HardAssertion {

SoftAssert softAssert = new SoftAssert();

@Test

public void **hardAssertion**(){

**Assert**.***assertEquals***("pass","pass");

System.out.println("This line is executed because assertEquals "

+ "passed as both the strings are same");

Assert.assertNull("assertion");

System.out.println("Since the object under assertion"

+ " is not null, the assertion will fail. "

+ "This line will not be executed");

}

@Test

public void **softAssertion**(){

**softAssert**.***assertNull***("assertion");

System.out.println("We are using Soft assertion in this method,"

+ " so this line of code will also be executed even if "

+ "the assetion fails.Wherever we want to execute full "

+ "testcase/method, we should use SoftAssertion");

softAssert.assertAll();

}

}

1. **What are the challenges and limitations of Selenium Webdriver**

* Can NOT test windows apps
* Can NOT use test mobile apps
  + Use Apium to handle …. and Android
* Limited Reporting
  + Basic reporting but use TestNg or extentReports to report.
* Handling dynamic elements
  + Some web elements are dynamic, changing ID, use dynamicXPath, or use dynamic CSS selectors: functions like startwith like contains
* Handling PageLoad different elements depends on different users or different drop down like country so use Wait to wait for the display.
* Handling pop-up windows: Use Auto ID to handle the popup windows
* Handling Captcha: 3rd party to handle Captcha

1. [**What Locators in Selenium**](https://www.protechtraining.com/content/selenium_tutorial-locators)

There are 8 locators )locators are objects) strategies included in Selenium:

* Identifier
* Id.
* Name.
* Partial **Link** Text.
* DOM.
* XPath.
* CSS.
* UI-element.

**In details:**

* Identifier

<html>

<body>

<form id="login">

<input name="username" type="text"/>

<input name="password" type="password"/>

<input name="submit" type="submit" value="Continue!"/>

</form>

</body>

</html>

Valid identifiers:

* **identifier=login**
* **identifier=username**
* **submit**
* Id.

**<label id="my\_id" />**

**name=pancake**

**id=my\_id**

or just **my\_id**

* Name

<html>

<body>

<div id="pancakes">

<button type="button" name="pancake" value="Blueberry">Blueberry</button>

<button type="button" name="pancake" value="Banana">Banana</button>

<button type="button" name="pancake" value="Strawberry">Strawberry</button>

</div>

</body>

</html>

**name=pancake**

**name=pancake value=Strawberry**

same as

**name=pancake index=2**

* Link: **link=The text of the link**
* DOM:
* **dom=document.div['pancakes'].button[0]**
* **document.div[0].button[2]**
* **dom=function foo() { return document.getElementById("pancakes"); }; foo();**
* XPath

XPath is the standard navigation tool for XML its’ like HTML.

**xpath=//button[@value="Blueberry"]**: matches the Blueberry button

**//div[@id="pancakes"]/button[0]**: same thing

* CSS.

The CSS locator strategy uses CSS selectors to find the elements in the page. Selenium supports CSS 1 through 3 selectors syntax excepted CSS3 namespaces and the following:

| **pseudo-classes** | **pseudo-elements** |
| --- | --- |
| **:nth-of-type** | **::first-line** |
| **:nth-last-of-type** | **::first-letter** |
| **:first-of-type** | **::selection** |
| **:last-of-type** | **::before** |
| **:only-of-type** | **::after** |
| **:visited** |  |
| **:hover** |  |
| **:active** |  |
| **:focus** |  |
| **:indeterminate** |  |

**css=div[id="pancakes"] > button[value="Blueberry"]**

selects the button with its value property set at *Blueberry* if children of the *pancakes* div

* UI-element.

As a general rule, keep in mind that if a locator matches several elements, only the first one will be effectively used by Selenium

1. **How you build Object Repository in your project**

* No default Object Repository concept in Selenium
* In Selenium we call objects as locators (such as ID, Name, Class Name, Tag Name, Link Text Partial Link Text, XPath, and CSS).
* Object Repository is a collection of objects
* Best way to create Object Repository is to use a Page Object Model
* In the page object model design pattern each web page is represented as a class
* All the objects related to a particular page of a web application are stored in a class.

QTP recording captures the objects.

**\*\* POM – Page Object Model**

1. **What is** [**page object model**](https://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html) **in Selenium?**

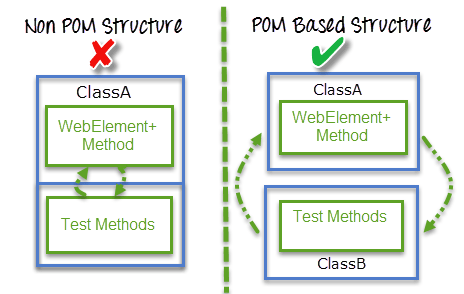
**Page Object Model** is an **Object** Repository design pattern in **Selenium** WebDriver. POM creates our testing code maintainable, reusable (reduce code duplication).

Or

A Page Object Model is a widely used design pattern in Selenium for enhancing test maintenance and reducing code application.

A Page Object is an object-oriented class that serves as an interface to a page of your Application Under Test (UAT). The tests then us the method of this page object class whenever they need to interact with the User Interface (UI) of that page.

The benefit is that if the UI changes for the page, there is no need to change the tests. Only the code within the code object needs to changed. Subsequently all changes to support that new UI is located in one place.



1. **What is Page Factory**

A ‘Page Factory’ is one way to implement ‘Page Object Model’.

**Page** Factory is an optimized way to create **object** repository in POM concept

The **Page Factory** Class in Selenium is an extension to the **Page** Object design pattern. It is used to initialize the elements of the **Page** Object or instantiate the **Page** Objects itself. ... It is used to initialize elements of a **Page** class without having to use 'FindElement' or 'FindElements'

1. **Difference between Page Object Model (POM) and Page Factory**

**Page Object is a class:** It represents a webpage and hold the functionality and members.

**Page Factory:** is a way to initialize the web elements you want to interact with within the page object when you create an instance of it

1. **What are the advantages of Page Object Model?**
   1. Code reusability – write the code once and use it in different tests.
   2. Code Maintainability – Clean separation between test code and page specific code such as locators and layouts. Code changes only on page object classes. And then when a UI changes it enhances test maintenance and reduces code duplication.

* 1. Object Repository: each page will be defined as a java class. All the fields in the page will be defined in the page as a member. The class will then implement the page.
  2. Readability: Improves readability due to separation between test code and page specific code.

1. **Why use POM?**

Starting an UI Automation in Selenium WebDriver is NOT a tough task. You just need to find elements, perform operations on it.

1. **How to take a screen shoot in Selenium**

This <https://www.softwaretestingmaterial.com/selenium-interview-questions/>

Or this

<https://mindmajix.com/selenium-interview-questions>

Test cases might fail during execution the test scripts. While executing the test cases manually we just take a screen shot and place in the result repository. Same can be done when using Selenium WebDriver.

Application Issues

Assertion Failures

Difficult to find WebElements on the web page

Timeout to find WebElements on the web page.

*Syntax to capture and save the screenshot.*

File = screenshotFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.File);

*Syntax to store it in our local drive*

FileUtils.copyFile(screenshotFile, new File(“filename\_with\_path”));

e.g.

FileUtils.copyFile(screenshotFile, new File(“D:\\screenshot.png”));

1. **How to drag and drop webelements in Selenium Webdriver**

Can’t use basic elements. Can be done using **Action** **Class**.

To get source locator:

WebElement *sourceLocator* = driver.findElements(By.xpath(“xpath”));

To get target locator:

WebElement *sourceLocator* = driver.findElement(By.xpath(“xpath”));

To create object ‘action’ of Actions class:

Action action = new Actions(driver);

Use dragAndDrop() method:

action.drageAndDrop(sourceLocator, targetLocator).build().perform();

1. **What are the types of frameworks in Selenium?**

**What framework you used**

* Linear Scripting Framework
* Modular Testing Framework
* **Data driven Testing Framework**
* Keyword driven Testing Framework
* Hybrid Testing Framework
* **Behavior Driven Development Framework - BDD**

**In Details**

* Linear Scripting Framework

Known as “record and playback” framework. Used for small websites.

* Modular Testing Framework
* Data driven Testing Framework
* Keyword driven Testing Framework
* Hybrid Testing Framework
* **Behavior Driven Development Framework**

The purpose of this framework is to create a platform which allows everyone (such as Business Analysts, Developers, Testers etc, ) to participate actively. It requires increased collaboration between development and Test Teams. It doesn't require the users to be acquainted with a programming language. We use non-technical, natural language to create test specifications. Some of the tools available in the market for Behavior driven development are JBehave, Cucumber, etc.,

* **What was your challenge in designing a framework.**
  + Identify all the right component which **will fulfil current and future requirements**
    - Think about the future
    - Enhance the framework
    - Future challenges during framework design.
    - Migration to other framework
  + How you write
    - Libraries
    - Page functions
    - Listener functions

1. **How to use Excel in Seleinum**

Use ‘JXL’ or Apache POI library work

1. Apache POI

Open source library developed by Apache to manipulate Excel and MS products.

1. How to handle Ajax in Selenium

We would not know when the Ajax call compeleted and the page completed.

Ajax: Asynchronous Java Script and XML

Allows the page to retrieve small amounts of data from the server without reloading the entire page. Sends HTTP request from Client to Server. And process the server response without loading entire page.

With Ajax **wait** might not work. Because actual page is not going to refresh the entire page.

Best solution is to use Dynamic Wait (i.e WebDriverWait in combination with ExpectedCondition)

e.g

* + 1. **titleIs()** wait for page with expected title.

wait.until(ExpectedConditions.titleIs("Deal of the Day"));

* + 1. elementToBeClickable()

wait.until(ExpectedConditions.elementToBeClickable(By.xpath(“xpath”)));

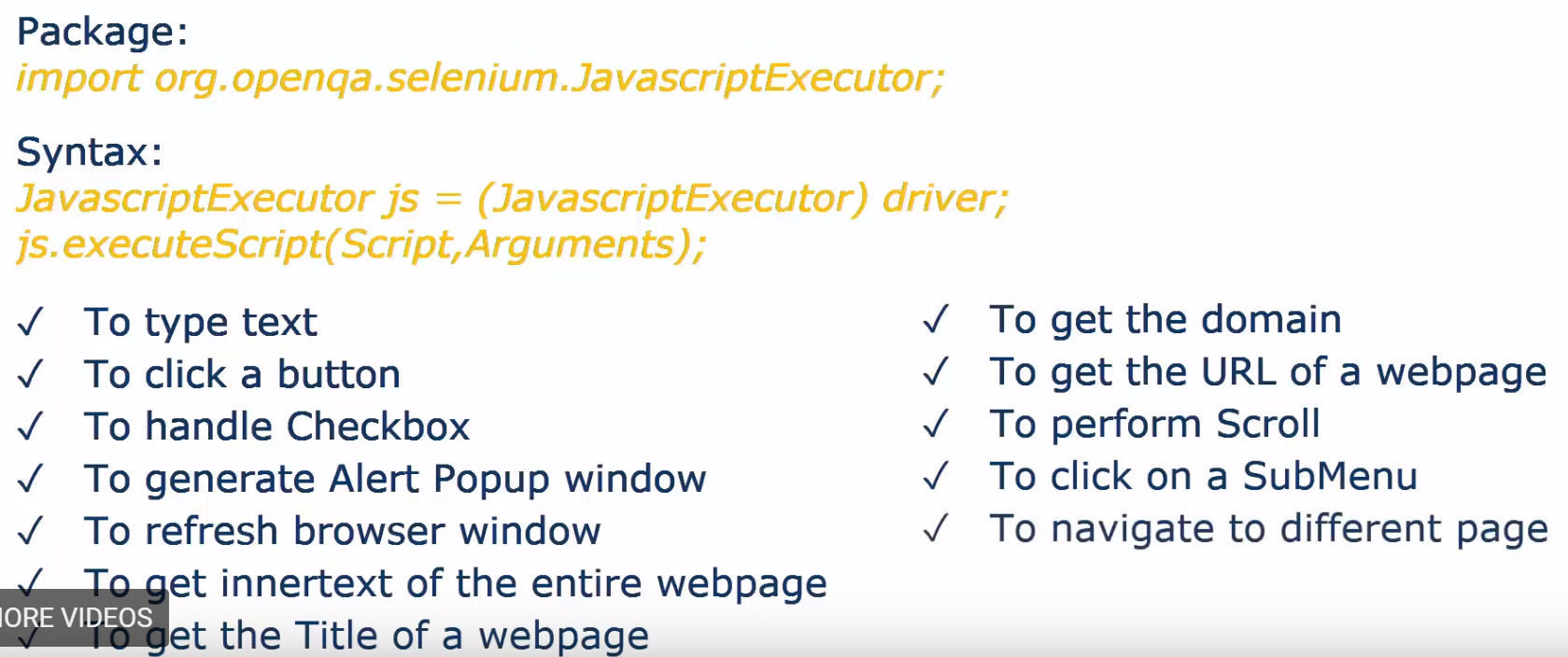
* + 1. alertIsPresent()

wait.until(ExpectedConditions.elementToBeClickable(By.xpath(“xpath”)));

* + 1. texToBePresentInElement()

wait.until(ExpectedConditions.texToBePresentInElement(By.id(“title”), “text to be found”));

1. What is JavascriptExecutor and were we use it in automation

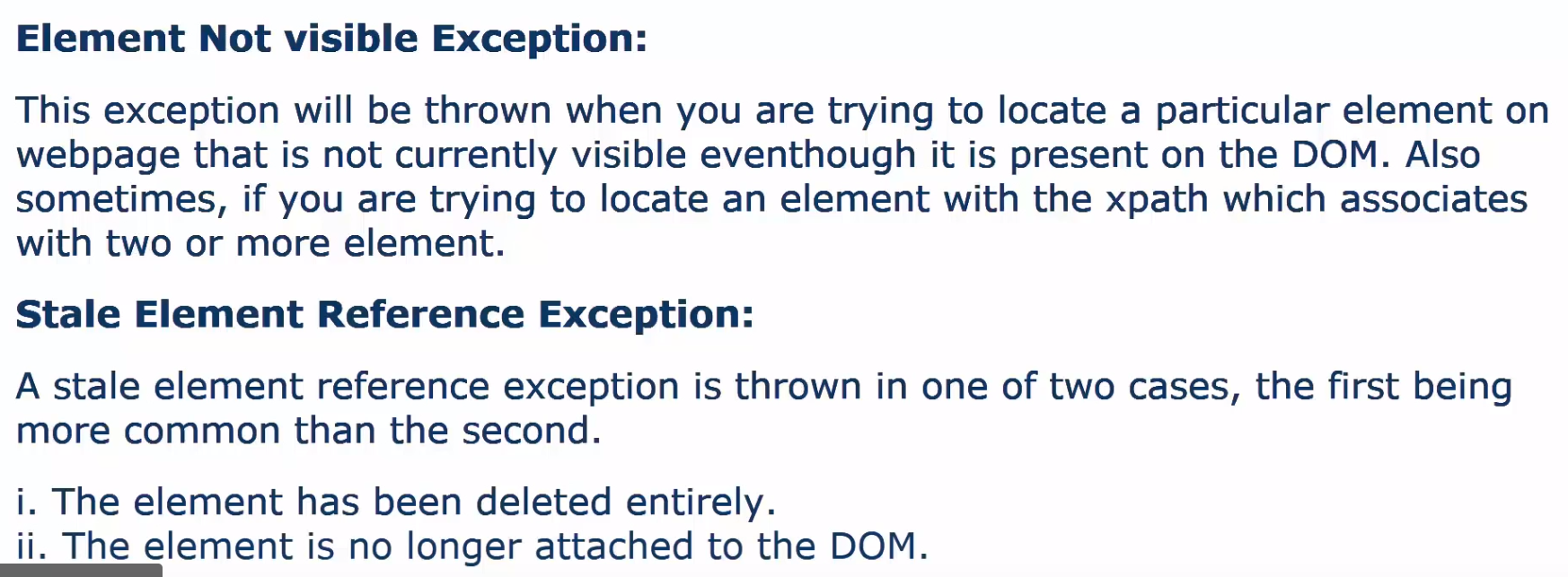


1. **What are the exceptions you have faced in your project?**

ElementNotVisibleException

Even if element

StaleElementReferecnceException



1. What is the differences between [findelement and findelements](https://www.softwaretestingmaterial.com/difference-between-findelement-and-findelements-methods/)

**findElement** method is used to access a single web element on a page. It returns the first matching element. It throws a NoSuchElementException exception when it fails to find If the element.

driver.findElement(By.xpath("Value of Xpath"));

e.g.

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

public class FindelementFindElements

{

public static void main (String [] args)

{

// Open browser

WebDriver driver = new FirefoxDriver();

// To maximize the window

driver.manage.window.maximize();

// Open Application

driver.get("https://www.google.co.in/?gws\_rd=ssl#q=softwaretestingmaterial.com");

// Get text of a particular link

String FindElement = driver.**findElement**(By.xpath("//\*[@id='rso']/div[1]/div/div/h3/a")).getText();

// Print the value of the link

System.out.println(FindElement);

// Click on the link

driver.**findElement**(By.xpath("//\*[@id='rso']/div[1]/div/div/h3/a")).click();

}

}

**Another Example:**

List<WebElement> el = driver.findElements(By.xpath("//\*"));

int count=0;

for ( WebElement e : el ) {

System.out.println( e.getTagName()+" "+e.getText());

count++;

}

System.out.println(count );

**findElements** method returns the list of all matching elements. The findElement method throws a **NoSuchElementException** exception when the element is not available on the page. Whereas, the **findElements** method **returns an empty list** when the element is not avai

List link = driver.findElements(By.xpath("Value of Xpath"));

e.g.

package seleniumTutorial;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.firefox.FirefoxDriver;

public class FindelementFindElements

{

public static void main (String [] args)

{

// Open browser

WebDriver driver = new FirefoxDriver();

// To maximize the window

driver.manage().window().maximize();

// Open application

driver.get("https://www.google.co.in/?gws\_rd=ssl#q=softwaretestingmaterial.com");

// Get the list of all links

List link = driver.**findElements**(By.xpath("//\*[@id='rso']/div/div/div/h3/a"));

// Using for loop to display the text of all the links

for(WebElement element:link)

{

System.out.println(element.getText());

}

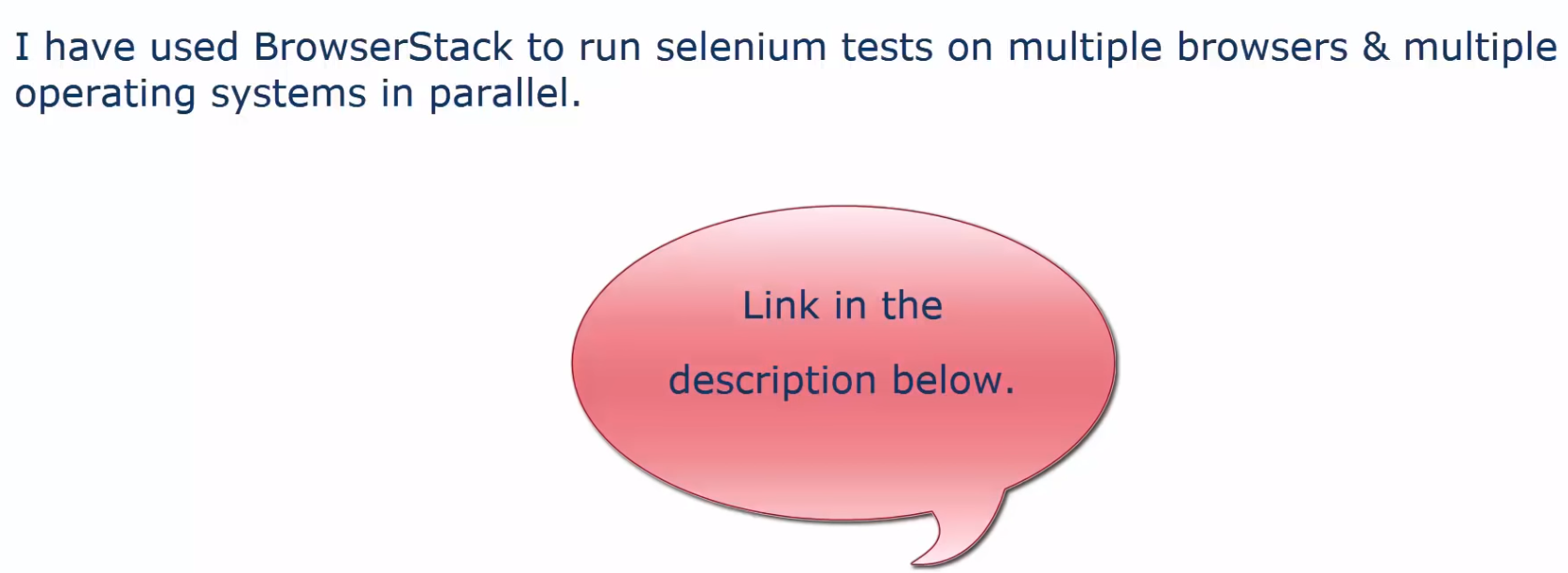
// Click on the first link

driver.findElement(By.xpath("//\*[@id='rso']/div/div/div/h3/a")).click();

}

}

1. Have you used crossbrowsertesting tool to run selenium scripts and cloud



1. How to handle uploading and downloading files in your project

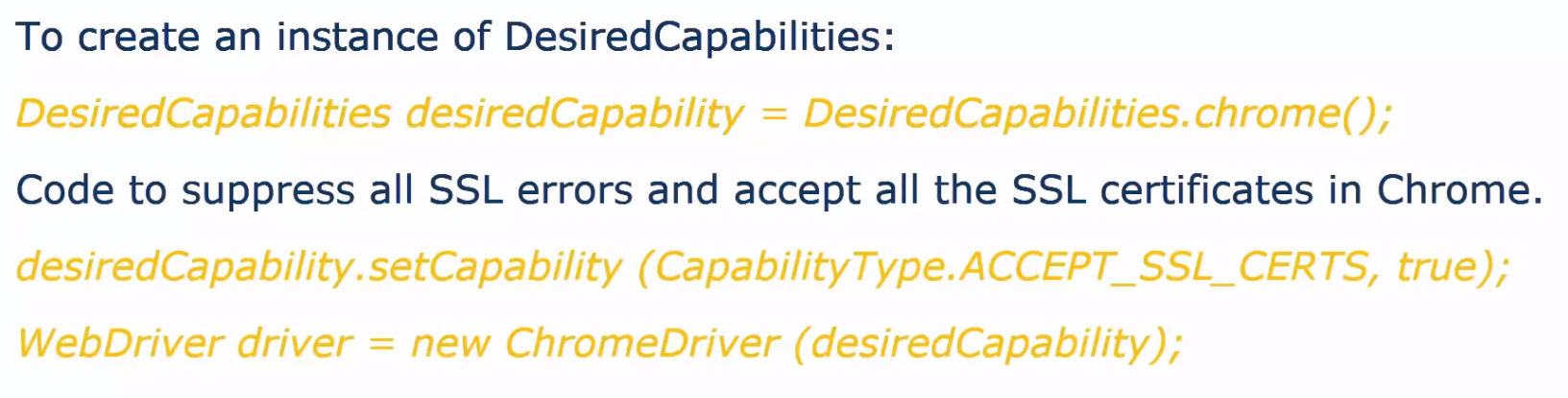
Robot class or AutoIT or Sikuli



1. Desired Capabilities?

Use Capabilies to handle SSL certs in Chrome browser.

capabilities in



1. CI /CD devops

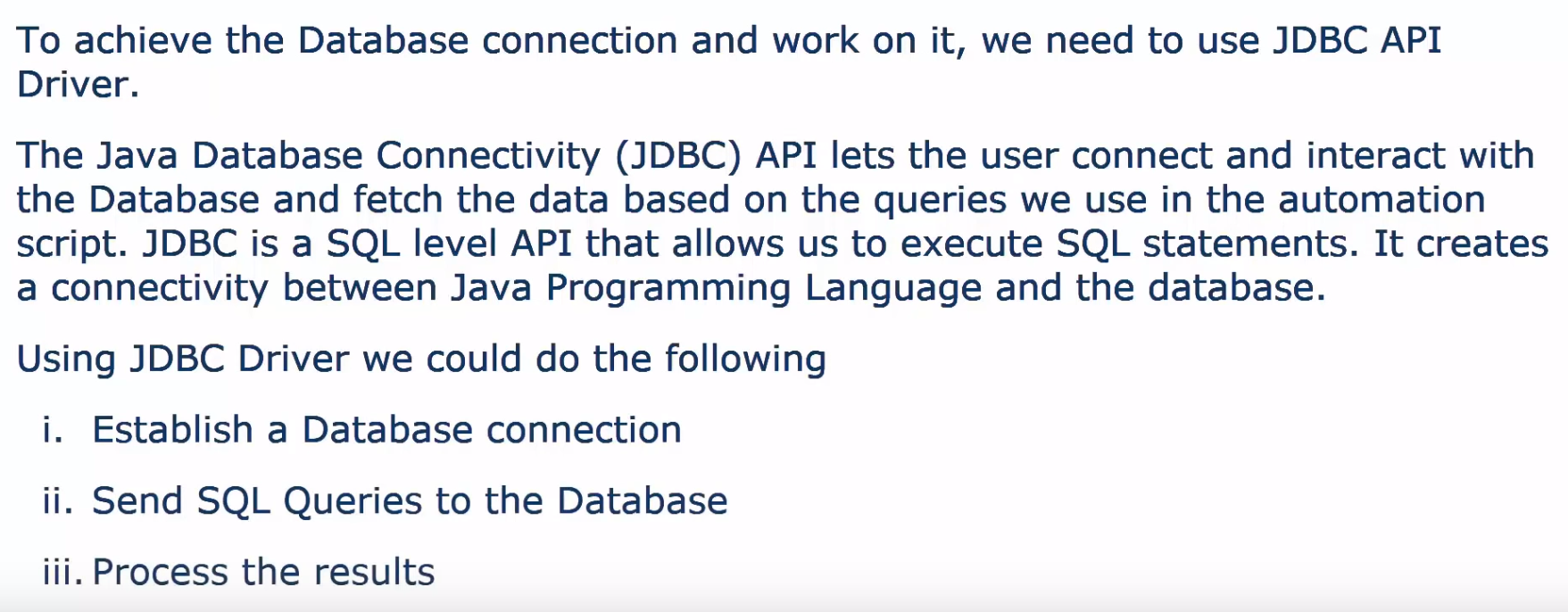
**CI continuously merge all dev checkins and build and test them.**

**CI**/**CD** development model and reap the benefits, companies need to build an effective pipeline to automate their build, integration and testing processes.

CI tools: Jenkins, TeamCity, Bamboo, Travis, Circle Ci, Bitbucket

1. How to use Selenium to Test a Database.

JDBC API Driver to achieve DB connection and work on it



<https://docs.microsoft.com/en-us/dotnet/framework/network-programming/how-to-send-data-using-the-webrequest-class>

**POM - how to enumerate the objects in a webpage selenium**

<https://stackoverflow.com/questions/6754284/selenium-how-to-capture-all-web-page-elements-and-associated-locators-on-a-pag>

use FindElements or FindElementsn

or use a Chrome Extension

"Selenium page object Generator"

**\*\* HTTP**

**C# Post GET**

WebRequest request = WebRequest.Create("http://www.contoso.com/");

request.Credentials = CredentialCache.DefaultCredentials;

((HttpWebRequest)request).UserAgent = ".NET Framework Example Client";

request.Method = "POST";

request.ContentLength = byteArray.Length;

request.ContentType = "application/x-www-form-urlencoded";

Stream dataStream = request.GetRequestStream ();

dataStream.Write (byteArray, 0, byteArray.Length);

dataStream.Close ();

WebResponse response = request.GetResponse();

Console.WriteLine (((HttpWebResponse)response).StatusDescription);

Stream data = response.GetResponseStream;

response.Close();

using System;

using System.IO;

using System.Net;

using System.Text;

namespace Examples.System.Net

{

public class WebRequestPostExample

{

public static void Main ()

{

// Create a request using a URL that can receive a post.

**WebRequest** request = WebRequest.Create ("http://www.contoso.com/PostAccepter.aspx ");

// Set the Method property of the request to POST.

request.**Method** = "**POST**";

// Create POST data and convert it to a byte array.

string postData = "This is a test that posts this string to a Web server.";

byte[] byteArray = Encoding.UTF8.GetBytes (postData);

// Set the ContentType property of the WebRequest.

request.ContentType = "application/x-www-form-urlencoded";

// Set the ContentLength property of the WebRequest.

request.ContentLength = byteArray.Length;

// Get the request stream.

Stream dataStream = request.GetRequestStream ();

// Write the data to the request stream.

dataStream.Write (byteArray, 0, byteArray.Length);

// Close the Stream object.

dataStream.Close ();

// Get the response.

WebResponse response = request.GetResponse ();

// Display the status.

Console.WriteLine (((HttpWebResponse)response).StatusDescription);

// Get the stream containing content returned by the server.

dataStream = response.GetResponseStream ();

// Open the stream using a StreamReader for easy access.

StreamReader reader = new StreamReader (dataStream);

// Read the content.

string responseFromServer = reader.ReadToEnd ();

// Display the content.

Console.WriteLine (responseFromServer);

// Clean up the streams.

reader.Close ();

dataStream.Close ();

response.Close ();

}

}

}

## Example GET

using System;

using System.Net;

using System.Net.Http;

using System.Net.Http.Headers;

using System.Threading.Tasks;

namespace HttpClientSample

{

public class Product

{

public string Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public string Category { get; set; }

}

class Program

{

static HttpClient client = new HttpClient();

static void ShowProduct(Product product)

{

Console.WriteLine($"Name: {product.Name}\tPrice: " +

$"{product.Price}\tCategory: {product.Category}");

}

static async Task<Uri> CreateProductAsync(Product product)

{

HttpResponseMessage response = await client.PostAsJsonAsync(

"api/products", product);

response.EnsureSuccessStatusCode();

// return URI of the created resource.

return response.Headers.Location;

}

static async Task<Product> GetProductAsync(string path)

{

Product product = null;

HttpResponseMessage response = await client.GetAsync(path);

if (response.IsSuccessStatusCode)

{

product = await response.Content.ReadAsAsync<Product>();

}

return product;

}

static async Task<Product> UpdateProductAsync(Product product)

{

HttpResponseMessage response = await client.PutAsJsonAsync(

$"api/products/{product.Id}", product);

response.EnsureSuccessStatusCode();

// Deserialize the updated product from the response body.

product = await response.Content.ReadAsAsync<Product>();

return product;

}

static async Task<HttpStatusCode> DeleteProductAsync(string id)

{

HttpResponseMessage response = await client.DeleteAsync(

$"api/products/{id}");

return response.StatusCode;

}

static void Main()

{

RunAsync().GetAwaiter().GetResult();

}

static async Task RunAsync()

{

// Update port # in the following line.

client.BaseAddress = new Uri("http://localhost:64195/");

client.DefaultRequestHeaders.Accept.Clear();

client.DefaultRequestHeaders.Accept.Add(

new MediaTypeWithQualityHeaderValue("application/json"));

try

{

// Create a new product

Product product = new Product

{

Name = "Gizmo",

Price = 100,

Category = "Widgets"

};

var url = await CreateProductAsync(product);

Console.WriteLine($"Created at {url}");

// Get the product

product = await GetProductAsync(url.PathAndQuery);

ShowProduct(product);

// Update the product

Console.WriteLine("Updating price...");

product.Price = 80;

await UpdateProductAsync(product);

// Get the updated product

product = await GetProductAsync(url.PathAndQuery);

ShowProduct(product);

// Delete the product

var statusCode = await DeleteProductAsync(product.Id);

Console.WriteLine($"Deleted (HTTP Status = {(int)statusCode})");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

Console.ReadLine();

}

}

}

## Example POST

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.OleDb;

using System.IO;

using System.Linq;

using System.Net.Http;

using System.Text;

using System.Threading.Tasks;

using System.Web.Script.Serialization;

namespace ConsoleApplication1

{

class Customer

{

public string Name { get; set; }

public string Address { get; set; }

public string Phone { get; set; }

}

public class Program

{

private static readonly HttpClient \_Client = new HttpClient();

private static JavaScriptSerializer \_Serializer = new JavaScriptSerializer();

static void Main(string[] args)

{

Run().Wait();

}

static async Task Run()

{

string url = "http://www.example.com/api/Customer";

Customer cust = new Customer() { Name = "Example Customer", Address = "Some example address", Phone = "Some phone number" };

var json = \_Serializer.Serialize(cust);

var response = await Request(HttpMethod.**Post**, url, json, new Dictionary<string, string>());

string responseText = await response.Content.ReadAsStringAsync();

List<YourCustomClassModel> serializedResult = \_Serializer.Deserialize<List<YourCustomClassModel>>(responseText);

Console.WriteLine(responseText);

Console.ReadLine();

}

/// <summary>

/// Makes an async HTTP Request

/// </summary>

/// <param name="pMethod">Those methods you know: GET, POST, HEAD, etc...</param>

/// <param name="pUrl">Very predictable...</param>

/// <param name="pJsonContent">String data to POST on the server</param>

/// <param name="pHeaders">If you use some kind of Authorization you should use this</param>

/// <returns></returns>

static async Task<HttpResponseMessage> Request(HttpMethod pMethod, string pUrl, string pJsonContent, Dictionary<string, string> pHeaders)

{

var httpRequestMessage = new HttpRequestMessage();

httpRequestMessage.Method = pMethod;

httpRequestMessage.RequestUri = new Uri(pUrl);

foreach (var head in pHeaders)

{

httpRequestMessage.Headers.Add(head.Key, head.Value);

}

switch (pMethod.Method)

{

case "**POST**":

HttpContent httpContent = new StringContent(pJsonContent, Encoding.UTF8, "application/json");

httpRequestMessage.Content = httpContent;

break;

}

return await \_Client.SendAsync(httpRequestMessage);

}

}

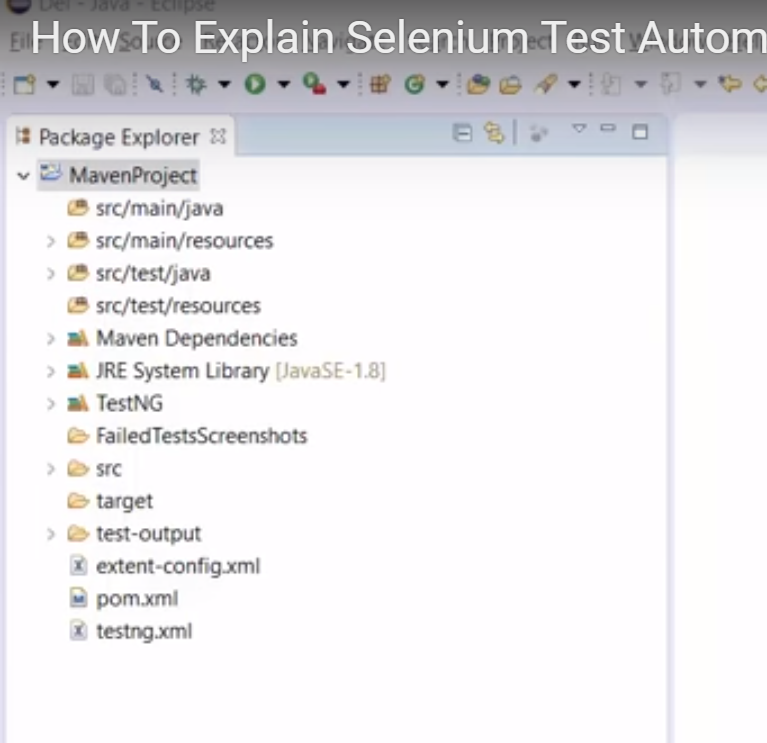
}

**\*\* Architect Design**

<http://toolsqa.com/selenium-webdriver/page-object-model/>

The Basic components of an Automation FrameWork Design:

<https://www.youtube.com/watch?time_continue=1&v=sPqf4ujbc0g>



1. Programming Lang – C#
2. Type of Framework – Data Driven by using POM with Page Factory.
3. Test Base Class -
4. POM –
   * Webpage

Maintained a class for every webpage. Each webpage has a separate class which holds the functionality and members of the webpage.

* + Tests

We have a separate class for every individual test

* + Packages

We have separate packages for pages and test scripts

e.g.

1. All the **webpage** related classes come under pages (package)

Src\main\resources\pages

e.g. HomePage & LoginPage are kept in separate classes/files.

1. All the **test** classes come under tests (package) Src\test\C#\tests\

Src\main\resources\tests\LoginTest

Src\main\resources\tests\TestBase

e.g. A separate class for HomePage & LoginPage to store element locators

and which call the methods from the homepage class and LoginPage class at:

Src\main\resources\pages

1. All Other are kept in **Src\main\resources**:
   * + - **Config** (**Config.Properties file**)

**Src\main\resources\pages\config\config.properties**

**To ease of maintenance**

Stores info that remain **static across** the **entire framework**

* + - * + Browser specific information
        + Application URL
        + Screen Shoots paths etc
      * **Element locator Pages**

Src\main\resources\pages\**HomePage.Class** ,

Src\main\resources\pages\**LoginPage.Class**

* + - * **Utility Files**

Src\main\resources\pages\Util\**TestUtil.class**

**Utility Files**: Or **Utility Class** or **Functions Class**.

**To achieve reusability** across entire framework.

It **extends** the **TestBase** Class to inherit the properties of test base in testUtil

**Are repeated tasks commonly used across the entire framework:**

Stores and handles the functions, **waits**, actions, **capturing screen shots**, accessing Excel sheets, and **sending emails**.

* + - * **testData (handle data driven testing)**

Src\main\resources\**testData**\Controller.xlsx

All historical test data will be kept in Excel Sheet. (using Apach POI to connect to MS software)

* + - * **failedTestScreenshots**

Src\main\**failedTestScreenshots**

Screen shots are also added in the extentReprt

Src\main\**test-output\extentReprt.html**

* + TestBase Class

Src\main\resources\tests\TestBase.class

Deals with all the common functions used by all the pages.

This class is responsible for:

* + - loading the configuration from property files
    - Initializing the WebDriver
    - Implicit waits
    - Extent Reports
    - Create the object of “File Input Stream” which responsible for pointing to the file from which the data should be read.

1. **Functions**

Src\main\resources\pages\Util\**TestUtil.class**

**Utility Files**: Or **Utility Class** or **Functions Class**.

**To achieve reusability** across entire framework.

It **extends** the **TestBase** Class to inherit the properties of test base in testUtil

**Are repeated tasks commonly used across the entire framework:**

Stores and handles the functions, **waits**, actions, **capturing screen shots**, accessing Excel sheets, and **sending emails**.

1. **Property files**
   * + - **Config** (**Config.Properties file**)

**Src\main\resources\pages\config\config.properties**

**To ease of maintenance**

Stores info that remain **static across** the **entire framework**

* + - * + Browser specific information
        + Application URL
        + Screen Shoots paths etc

1. **TestNg**

For assertion, grouping, and parallel execution

1. Parameterization - Excel files
2. **Error Screenshots**
3. **Sending Emails**
4. **Version Control**

Git to store test scripts.

1. **CI tool**

Jenkins to build daily bases. Nightly Executions. Test results are sent to the peers using Jenkins

1. How to Execute **Maven** Project Using Jenkins

Using Maven (Nuget) to Build, execution and dependency purpose.

the test scripts were triggered through Jenkins

one VM’s run Chrome, another run FirFox.

1. **ExtentReport**

Nice HTML reports. Logs, Screen shots for failed test cases.

## \*\* C# Basic Questions

# **Sample LINQ Queries**

[**http://www.tutorialsteacher.com/linq/sample-linq-queries**](http://www.tutorialsteacher.com/linq/sample-linq-queries)

# **101 LINQ Samples**

[**https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b**](https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b)

**15 Essential C# Interview Questions \***

[**https://www.toptal.com/c-sharp/interview-questions**](https://www.toptal.com/c-sharp/interview-questions)

**Callback Operation By Delegate Or Interface**

[**https://www.c-sharpcorner.com/UploadFile/1c8574/delegate-used-for-callback-operation/**](https://www.c-sharpcorner.com/UploadFile/1c8574/delegate-used-for-callback-operation/)

**Callback by Delegate**

[**https://www.infoworld.com/article/2996770/application-architecture/how-to-work-with-delegates-in-c.html**](https://www.infoworld.com/article/2996770/application-architecture/how-to-work-with-delegates-in-c.html)

There are three steps in using delegates.

These include: declaration, instantiation, and invocation.

The signature of a delegate looks like this:

delegate result-type identifier ([parameters])

The following statement shows how you can **declare a delegate**.

public delegate void MyDelegate(string text);

 instantiate the delegate declared above.

MyDelegate d = new MyDelegate(ShowText);

invoke the method that the delegate points to easily.

d("Hello World...");

You can also invoke the method that the delegate instance points to using the Invoke() method on the delegate instance as shown below.

d.Invoke("Hello World...");

**The following is an example of Callback by Delegate**

1. **public** **delegate** **void** TaskCompletedCallBack(**string** taskResult);
2. **public** **class** CallBack
3. {
4. **public** **void** StartNewTask(TaskCompletedCallBack taskCompletedCallBack)
5. {
6. Console.WriteLine("I have started new Task.");
7. **if** (taskCompletedCallBack != **null**)
8. taskCompletedCallBack("I have completed Task.");
9. }
10. }
11. **public** **class** CallBackTest
12. {
13. **public** **void** Test()
14. {
15. TaskCompletedCallBack callback = TestCallBack;
16. CallBack testCallBack = **new** CallBack();
17. testCallBack.StartNewTask(callback);
18. }
19. **public** **void** TestCallBack(**string** result)
20. {
21. Console.WriteLine(result);
22. }
23. }
24. **static** **void** Main(**string**[] args)
25. {
27. CallBackTest callBackTest = **new** CallBackTest();
28. callBackTest.Test();
29. Console.ReadLine();
30. }

**Output:***I have started new Task.*

*I have completed Task.*

**Threads**

[**https://www.pluralsight.com/guides/how-to-write-your-first-multi-threaded-application-with-c**](https://www.pluralsight.com/guides/how-to-write-your-first-multi-threaded-application-with-c)

Starting Thread

public class SimpleThreadExample

{

public void StartMultipleThread()

{

DateTime startTime = DateTime.Now;

Thread t1 = new Thread(() =>

{

int numberOfSeconds = 0;

while (numberOfSeconds < 5)

{

Thread.Sleep(1000);

numberOfSeconds++;

}

Console.WriteLine("I ran for 5 seconds");

});

Thread t2 = new Thread(() =>

{

int numberOfSeconds = 0;

while (numberOfSeconds < 8)

{

Thread.Sleep(1000);

numberOfSeconds++;

}

Console.WriteLine("I ran for 8 seconds");

});

//parameterized thread

Thread t3 = new Thread(p =>

{

int numberOfSeconds = 0;

while (numberOfSeconds < Convert.ToInt32(p))

{

Thread.Sleep(1000);

numberOfSeconds++;

}

Console.WriteLine("I ran for {0} seconds", numberOfSeconds);

});

t1.Start();

t2.Start();

//passing parameter to parameterized thread

t3.Start(20);

//wait for t1 to fimish

t1.Join();

//wait for t2 to finish

t2.Join();

//wait for t3 to finish

t3.Join();

Console.WriteLine("All Threads Exited in {0} secods", (DateTime.Now - startTime).TotalSeconds);

}

}

## Terminating Thread

Thread.Abort()destroy the running thread.

 better to let the CLR do the dirty work for you

public class DestroyThreadExample

{

public bool IsCancelled { get; set; }

public Thread MyThread { get; set; }

public void StartThread()

{

MyThread = new Thread(() =>

{

int numberOfSeconds = 0;

while (numberOfSeconds < 8)

{

if (IsCancelled == false)

{

break;

}

Thread.Sleep(1000);

numberOfSeconds++;

}

Console.WriteLine("I ran for {0} seconds", numberOfSeconds);

});

}

//Destroy thread

public void Abort()

{

MyThread.Abort();

}

//Graceful exit

public void GracefulAbort()

{

IsCancelled = true;

}

}

## BackgroundWorker

## Issues with Threads

* Deadlocks and
* Race conditions are few to name.

**Solution use:**

Lock keyword

 it will allow only one thread to execute the code within the lock block.

**Lambda**

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions>

Lambda Expressions (C# Programming Guide)

delegate bool D();

delegate bool D2(int i);

class Test

{

D del;

D2 del2;

public void TestMethod(int input)

{

int j = 0;

// Initialize the delegates with lambda expressions.

// Note access to 2 outer variables.

// del will be invoked within this method.

del = () => { j = 10; return j > input; };

// del2 will be invoked after TestMethod goes out of scope.

del2 = (x) => {return x == j; };

// Demonstrate value of j:

// Output: j = 0

// The delegate has not been invoked yet.

Console.WriteLine("j = {0}", j); // Invoke the delegate.

bool boolResult = del();

// Output: j = 10 b = True

Console.WriteLine("j = {0}. b = {1}", j, boolResult);

}

static void Main()

{

Test test = new Test();

test.TestMethod(5);

// Prove that del2 still has a copy of

// local variable j from TestMethod.

bool result = test.del2(10);

// Output: True

Console.WriteLine(result);

Console.ReadKey();

}

}

**Singelton**

Singelton

-----------

There are various ways to implement the Singleton Pattern in C#. The following are the common characteristics of a Singleton Pattern.

A single constructor, that is private and parameterless.

The class is sealed.

A static variable that holds a reference to the single created instance, if any.

A public static means of getting the reference to the single created instance, creating one if necessary.

Singelton Example

-------------------

// Bad code! Do not use!

public sealed class Singleton

{

//Private Constructor.

private Singleton()

{

}

private static Singleton instance = null;

public static Singleton Instance

{

get

{

if (instance == null)

{

instance = new Singleton();

}

return instance;

}

}

}

Full Thread Safe

----------------

**namespace Singleton**

{

class Program

{

static void Main(string[] args)

{

Calculate.Instance.ValueOne = 10.5;

Calculate.Instance.ValueTwo = 5.5;

Console.WriteLine("Addition : " + Calculate.Instance.Addition());

Console.WriteLine("Subtraction : " + Calculate.Instance.Subtraction());

Console.WriteLine("Multiplication : " + Calculate.Instance.Multiplication());

Console.WriteLine("Division : " + Calculate.Instance.Division());

Console.WriteLine("\n----------------------\n");

Calculate.Instance.ValueTwo = 10.5;

Console.WriteLine("Addition : " + Calculate.Instance.Addition());

Console.WriteLine("Subtraction : " + Calculate.Instance.Subtraction());

Console.WriteLine("Multiplication : " + Calculate.Instance.Multiplication());

Console.WriteLine("Division : " + Calculate.Instance.Division());

Console.ReadLine();

}

}

public sealed class Calculate

{

private Calculate()

{

}

private static Calculate instance = null;

public static Calculate Instance

{

get

{

if (instance == null)

{

instance = new Calculate();

}

return instance;

}

}

public double ValueOne { get; set; }

public double ValueTwo { get; set; }

public double Addition()

{

return ValueOne + ValueTwo;

}

public double Subtraction()

{

return ValueOne - ValueTwo;

}

public double Multiplication()

{

return ValueOne \* ValueTwo;

}

public double Division()

{

return ValueOne / ValueTwo;

}

}

}

**Kusto**

------

Kusto is a log analytics cloud platform optimized for ad-hoc big data queries. Read more about it here: http://aka.ms/kdocs. Equal to IBM Solar

https://docs.microsoft.com/en-us/connectors/kusto/

**SQL**

**SQL IDE:**

**-----------**

<https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_min>

**Amazing Find Second Highest**

**Find second lowest**

[**https://www.youth4work.com/Talent/MySql/Forum/118665-sql-query-for-find-second-highest-salary-of-employee**](https://www.youth4work.com/Talent/MySql/Forum/118665-sql-query-for-find-second-highest-salary-of-employee)

SELECT name, MAX(salary) AS salary FROM employee

WHERE

salary <

(SELECT MAX(salary) FROM employee);

**Find Second and find second to second last**

SELECT MAX(Price) FROM Products

WHere

Price <

(SELECT MAX(Price) FROM Products

Where

Price <

(SELECT MAX(Price) FROM Products));

Find Second and find second to secon last

SELECT MAX(Price) FROM Products

WHere

Price <

(SELECT MAX(Price) FROM Products

Where

Price <

(SELECT MAX(Price) FROM Products));

**\*\* Architect Positions Interview**

When creating a Test Automation Framework, we should consider the following main points:

* To be able to **create automated tests quickly** by using appropriate **abstraction layers**
* The framework should have **meaningful** **logging** and **reporting** structure
* Should be easily **maintainable** and **extendable**
* Should be **simple** **enough for testers to write automated** tests
* A **retry mechanism** to rerun failed tests – this is especially useful for WebDriver UI tests

**Capgemini Interview**

**List the AWS Services**

1. [Amazon EC2 - Virtual Servers in the Cloud](https://aws.amazon.com/ec2/?c=1&pt=1)
2. [Amazon EC2 Auto Scaling - Scale Compute Capacity to Meet Demand](https://aws.amazon.com/ec2/autoscaling/?c=1&pt=2)
3. [Amazon Elastic Container Service - Run and Manage Docker Containers](https://aws.amazon.com/ecs/?c=1&pt=3)
4. [Amazon Elastic Container Service for Kubernetes - Run Managed Kubernetes on AWS](https://aws.amazon.com/eks/?c=1&pt=4)
5. [Amazon Elastic Container Registry - Store and Retrieve Docker Images](https://aws.amazon.com/ecr/?c=1&pt=5)
6. [Amazon Lightsail - Launch and Manage Virtual Private Servers](https://aws.amazon.com/lightsail/?c=1&pt=6)
7. [AWS Batch - Run Batch Jobs at Any Scale](https://aws.amazon.com/batch/?c=1&pt=7)
8. [AWS Elastic Beanstalk - Run and Manage Web Apps](https://aws.amazon.com/elasticbeanstalk/?c=1&pt=8)
9. [AWS Fargate - Run Containers without Managing Servers or Clusters](https://aws.amazon.com/fargate/?c=1&pt=9)
10. [AWS Lambda - Run your Code in Response to Events](https://aws.amazon.com/lambda/?c=1&pt=10)
11. [AWS Serverless Application Repository - Discover, Deploy, and Publish Serverless Applications](https://aws.amazon.com/serverlessrepo/?c=1&pt=11)
12. [VMware Cloud on AWS - Build a Hybrid Cloud without Custom Hardware](https://aws.amazon.com/vmware/?c=1&pt=12)
13. [AWS Outposts - Run AWS services on-premises](https://aws.amazon.com/outposts/?c=1&pt=13)

**Amazon** Web Services is known primarily as an **IaaS** (infrastructure as a service), and with good reason: The **Amazon** cloud is practically synonymous with public**cloud computing** in general and with **IaaS** in particular. Yet, many of the services available in **AWS** are comparable to PaaS (platform as a service) offerings.

Sas 🡪 Paas 🡪 Iaas

**SQL Interview Questions**

<https://www.w3schools.com/sql/default.asp>

SELECT TOP 3 \* FROM Customers;

**Same as:**

SELECT \* FROM Customers  
LIMIT 3;

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns.

SELECT COUNT(CustomerID), Country  
FROM Customers  
GROUP BY Country;

**Inner Join**

All matching row between two tables

**Left Join**

The LEFT JOIN keyword returns all records from the left table (table1), and the matched records from the right table (table2). The result is NULL from the right side, if there is no match.



**Example: Both Exactly the same. But result is different.**

left Outer Join

----------

SELECT c.customerid,

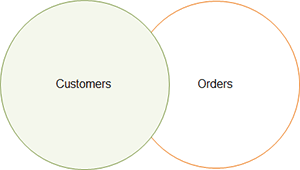
c.companyName,

orderid

FROM customers c

LEFT JOIN orders o ON o.customerid = c.customerid

ORDER BY orderid



right outer join

------------------

SELECT c.customerid,

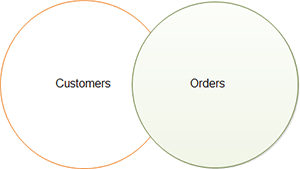
c.companyName,

orderid

FROM customers c

RIGHT JOIN orders o ON o.customerid = c.customerid

ORDER BY orderid



**SQL FULL OUTER JOIN Keyword**

Returns everything. The combination of both sides.



* all rows in the left table table\_A.
* all rows in the right table table\_B.
* and all matching rows in both tables.

SELECT c.customerid,

       c.companyName,

       orderid

FROM customers c

FULL OUTER JOIN orders o ON o.customerid = c.customerid

ORDER BY orderid

Also can be:

SELECT column1, column2...

FROM table\_A

LEFT JOIN table\_B ON join\_condition

UNION

SELECT column1, column2...

FROM table\_A

RIGHT JOIN table\_B ON join\_condition

<https://www.guru99.com/sql-interview-questions-answers.html>

**Top Experienced SQL interview Questions**

<https://www.edureka.co/blog/interview-questions/sql-interview-questions>

**Data Modeling Interview Questions**

<https://www.softwaretestinghelp.com/data-modeling-interview-questions-answers/>

**Primary Key:**

* 1. A unique key that identify each record/row
  2. Automatic unique constraint defined on
  3. Only one primary key column per table
  4. Can NOT be null.

**Foreign key**

* A primary key of another table which connects two tables.
* Can contain Null values.
* enforces referential integrity. (every value in the second table where the key is a foreign key, has a corresponding value in the main table where the key is primary. And if a the primary key and its value is deleted in one table its also deleted in the other table.)

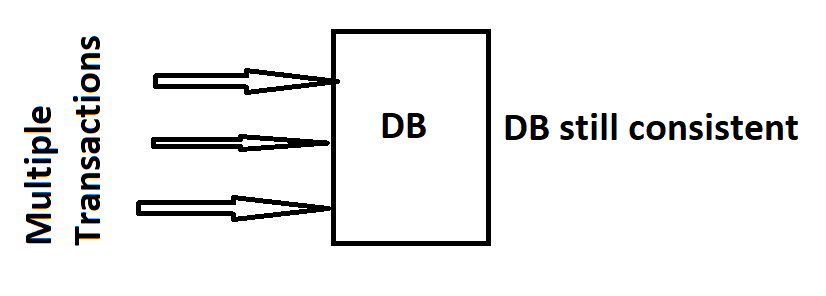
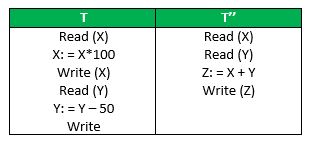
This is a special kind of unique key, and it has implicit NOT NULL constraint. It means,

[**What is ACID properties**](https://www.geeksforgeeks.org/acid-properties-in-dbms/)

Entity frame work diagram.

* **Atomicity**
* **Consistency**
* **Isolation**
* **Durability**

**Entity frame work diagram In Details.**

* **Atomicity -** Atomicity is also known as the ‘**All or nothing rule’**
  + —**Abort**: If a transaction aborts, changes made to database are not visible.  
    —**Commit**: If a transaction commits, changes made are visible.
  + Say transaction fails after write(x) and before write (y) then it’s all reverted.
  + 
* **Consistency** 
  + correctness of a database
  + Integrity constraints must be maintained (so **db is** **consistent** before and after the transaction T)
  + E.g the sum of (200+500 = 700) is true before and after transaction T
  + Inconsistency occurs in case **T1** completes but **T2** fails. As a result T is incomplete.
* **Isolation** 
  + **Multiple transactions** can occur concurrently
  + Without leading to **(no) inconsistency** of database state.
  + Transactions occur independently without interference
  + Transactions written to memory or has been committed only then can be visible to other transactions.
  + 
  + 
* **Durability**
  + On transaction that has been completed/executed.
  + updates and modifications to the database are stored in and written to disk and they persist even if system failure occurs.
  + These updates now become permanent and are stored in a non-volatile memory.
  + Transaction are therefore never lost.

The **ACID** properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored.

LISA Test: like MT - Microsoft Test to

Provides an automated testing solution for distributed **application**architectures that leverage SOA, BPM, integration suites, and ESBs. Product teams can use LISA Test to design and execute automated unit, functional, regression, integration, load, and performance tests.

Confluence vs Jira

**Confluence** (**Sharepoint**) is a collaboration wiki tool **used** to help teams to collaborate and share knowledge efficiently. With **confluence**, we can capture project requirements, assign tasks to specific users, and manage several calendars at once with the help of Team Calendars add-on

**Jira's** (**VSOnline**) - a bug-tracking system and **Confluence** is a team collaboration/knowledge management/wiki system.

* + 1. Bug tracking and
    2. Agile project management.

**Jenkins**

**Jenkins** is a continuous integration tool with

**What kind of Jobs have you created in Jenkins?**

Build jobs, Run Smoke Test, Run Regression Test

**What are those jobs do?**

Build Jobs

Run Test Suite Jobs

**What technology have you used to write those Jobs?**

**And what framework have you used to write those job in****?**

Go inside a Jenkins Job

What tasks do you see inside a Jenkins Job

**Have you configured a Jenkins Job?**

**What is a pipeline in Jenkins?**

**Branching Strategy**

**Git Commands**

<https://www.youtube.com/watch?v=SWYqp7iY_Tc>

<https://rubygarage.org/blog/most-basic-git-commands-with-examples>

Add all files and directories to the staging area in GIT.

git add --all

Remove files and directories from the staging area in GIT.

git rm --cached my-file.ts

or use

git reset some-file.js

Will commit from the staging area to the local repository

git commit -m "Add three files"

Advanced GIT: add + commit

git commit -a -m "Do something once more"

will add the files then commit them. Using the option -a

add the 10th file to an already committed 9th files. i.e committed 9 files but forgot to add the 10th. This is a way around that

Way One: undo the commit:

git reset --soft HEAD^

--soft" option means that the commit is canceled and moved before HEAD

--soft HEAD^ can be read as Undo the last commit in the current branch and move HEAD back by one commit”

^

Way Two:

git add file-i-forgot-to-add.html

git commit --amend -m "Add the remaining file”

The "--amend" option lets you amend the last commit by adding a new or multiple files

"

**Remote Repository**

Bind this remote repository to your local repository:

Or

Connect your local and remote repositories

git remote add origin https://github.com/YourUsername/some-small-app.git

Now GitHub *repository* will accept code from your local *repository*

What is “origin”

The "origin" option is the default name for the server on which your remote repository is located.



*Once you run the command above, Git will connect your local and remote repositories. But what does this liaison actually mean? Can you already access your code online? Unfortunately, not yet.*

**Push**: You push commits to a remote repository.

With Git, copying your code to a remote repository looks like this

git push -u origin master

first time use -u so next time push is used one can just say:  
git push

It's obvious that the command "push" tells Git to push your files to a remote repository. What we also specified is the server our local repo is connected to (origin) and the branch we're pushing, which is master.

Get a list of repositories (-v will list all remote repositories user connected to)

Mars@Dell01 MINGW64 ~/source/MyownCode (master)

$ git remote -v

origin https://github.com/hazimt/MyownCode.git (fetch)

origin https://github.com/hazimt/MyownCode.git (push)

Clone

If you don't like the name of the repository you're cloning, just change to

git clone git@github.com:YourUsername/your-app.git this-name-is-much-better

Cloning will:

* Download the entire project into a specified directory; and
* Create **a remote repository** called **origin** and point it to the URL you pass.

When you run the "pull" command, Git will:

* Pull changes in the current branch made by other developers; and
* Synchronize your local repository with the remote repository.

Difference between

Fetch and Pull

**Git fetch =** Main command used to **download** contents from a remote repository.

**Usage:**

**Fetching** is what you **do** when you want to see what everybody else has been **working** on

**git pull** =  **git** fetch + **git** merge FETCH\_HEAD

What is a pull command do?

How

No access to master branch

**Branch 1**: Stable branch = for stable, working application

**Branch 2**: Development branch = developing a new feature

**Branch 3**: Testing Branch = test and fix app branch before pushing to production.

Create a new branch with the name user-profile

git branch user-profile

List all branches

git branch

The output will be:

\*master

user-profile

Switch branches

git checkout user-profile

Do an RI

1. first switch back to the main branch

git checkout master

1. Now merge the user-profile branch to master.

git merge user-profile

1. If the user-profile branch is no longer needed delete it

git branch -d user-profile

But this will fail:

error: Cannot delete the branch 'user-profile' which you are currently on.

For this step refer to:  
<https://stackoverflow.com/questions/2003505/how-do-i-delete-a-git-branch-both-locally-and-remotely>

The Short Answers

If you want more detailed explanations of the following commands, then see the long answers in the next section.

Deleting a remote branch:

git push origin --delete <branch> # Git version 1.7.0 or newer

git push origin :<branch> # Git versions older than 1.7.0

Deleting a local branch:

git branch --delete <branch>

git branch -d <branch> # Shorter version

git branch -D <branch> # Force delete un-merged branches

Deleting a local remote-tracking branch:

git branch --delete --remotes <remote>/<branch>

git branch -dr <remote>/<branch> # Shorter

git fetch <remote> --prune # Delete multiple obsolete tracking branches

git fetch <remote> -p # Shorter

The Long Answer: there are 3 different branches to delete!

When you're dealing with deleting branches both locally and remotely, keep in mind that there are 3 different branches involved:

The local branch X.

The remote origin branch X.

The local remote-tracking branch origin/X that tracks the remote branch X.

To delete that actual remote branch, you need

git push origin --delete bugfix

This cmd (use this):

git checkout -b admin-panel

is equal these two:

git branch my-new-branch

git checkout my-new-branch

**Cloud Experience**

**Performance**

Explain what tools and the technologies you’ve used

* Performance Wizard
* Other similar ones are
  + JMeter
  + LoadRunner (free up to 50 users)
  + Visual Studio Profiler

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

If the server is slow. What are those parameters you’ll be looking for?

<https://www.google.com/search?q=If+the+server+is+slow.+What+are+those+parameters+you%E2%80%99ll+be+looking+for%3F&oq=If+the+server+is+slow.+What+are+those+parameters+you%E2%80%99ll+be+looking+for%3F&aqs=chrome..69i57.586j0j7&sourceid=chrome&ie=UTF-8>

What parameters would you tweek

Billing System

More advanced UI technology

Selenium

Java

DB queries