

**Selenium WebDriver Questions:**

1. **What are the different types of locators in Selenium?**
   * **Answer**: Selenium supports these types of locators:
     + ID
     + Name
     + Class Name
     + Tag Name
     + Link Text
     + Partial Link Text
     + CSS Selector
     + XPath These locators are used to identify web elements on a webpage.
2. **When do you use XPath over CSS locators?**
   * **Answer**: Use XPath when:
     + You need to navigate through parent/child relationships.
     + The element has dynamic attributes.
     + The structure of the HTML changes frequently.
     + XPath is more flexible for complex queries.
3. **How to get a specific value from a dropdown and reuse it in verifications?**
   * **Answer**:

python

Copy code

from selenium.webdriver.support.ui import Select

dropdown = Select(driver.find\_element(By.ID, "dropdown\_id"))

selected\_option = dropdown.first\_selected\_option.text

assert selected\_option == "Expected Value"

1. **When do we use JavaScript Executors?**
   * **Answer**: Use JavaScript Executors when:
     + Performing actions like clicking or scrolling that are not possible with WebDriver.
     + Handling dynamic elements that require JavaScript execution.

python

Copy code

driver.execute\_script("arguments[0].click();", element)

1. **Is it possible to validate Captcha using Selenium? If yes, how?**
   * **Answer**: Captcha is not automatable with Selenium directly due to its design. Use:
     + Third-party tools like OCR (e.g., Tesseract) or APIs to handle Captchas.
     + Bypass by disabling Captcha in test environments.
2. **What should be the ideal way to store data using Selenium WebDriver only?**
   * **Answer**: Store data using:
     + Properties files.
     + JSON, XML, or CSV files.
     + In-memory data structures like dictionaries or lists.
3. **Is it possible to use XPath like parent/child/node/..? If yes, provide an example where to use it?**
   * **Answer**: Yes, use XPath for:
     + Navigating DOM structures.

xpath

Copy code

//div[@class='parent']/child::span

1. **What happens if you receive browser notifications during test automation execution?**
   * **Answer**: Use browser capabilities to disable notifications:

python

Copy code

from selenium.webdriver.chrome.options import Options

options = Options()

options.add\_argument("--disable-notifications")

1. **Why does Stale Element Exception occur, and how to handle it?**
   * **Answer**: Occurs when:
     + The element is no longer attached to the DOM.
   * Handle it by:
     + Refreshing the page.
     + Re-locating the element.
2. **What is Invalid Certificate Exception?**
   * **Answer**: Happens when the browser encounters a site with an expired or untrusted SSL certificate. Use capabilities to ignore this:

python

Copy code

options = Options()

options.add\_argument('--ignore-certificate-errors')

**API Questions:**

1. **What are the components of an HTTP request?**
   * **Answer**: An HTTP request consists of:
     + **Request Line**: Includes the HTTP method (e.g., GET, POST), the requested URL, and the HTTP version.
     + **Headers**: Provide metadata about the request (e.g., Content-Type, Authorization).
     + **Body (optional)**: Contains data sent to the server (used in POST/PUT requests).
     + **Query Parameters**: Key-value pairs appended to the URL for filtering or specifying data.
2. **What is the difference between API and unit testing?**
   * **Answer**:
     + **API Testing**:
       - Focuses on testing the APIs directly to ensure they return the correct responses for given requests.
       - Validates business logic, data responses, and performance.
       - Example: Sending a POST request to create a user and validating the response status (e.g., 201).
     + **Unit Testing**:
       - Focuses on testing individual units (functions/methods) in isolation.
       - Does not involve external dependencies like databases or APIs.
       - Example: Testing a function that calculates the total cost of an order.
3. **What is an HTTP response?**
   * **Answer**: An HTTP response is sent by the server to the client after processing an HTTP request. It consists of:
     + **Status Code**: Indicates the result of the request (e.g., 200 OK, 404 Not Found).
     + **Headers**: Metadata about the response (e.g., Content-Type, Content-Length).
     + **Body (optional)**: Contains the data returned by the server (e.g., JSON, HTML).
4. **How can we add validation points in Postman?**
   * **Answer**: Postman allows validation using test scripts written in JavaScript. Example:

javascript

Copy code

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

pm.test("Response time is less than 500ms", function () {

pm.expect(pm.response.responseTime).to.be.below(500);

});

pm.test("Response contains expected value", function () {

pm.expect(pm.response.json().key).to.eql("expected\_value");

});

1. **What do you understand by server-side validation?**
   * **Answer**:
     + Server-side validation ensures that the data sent from the client is checked and verified on the server before processing.
     + Prevents security issues like SQL injection, XSS, and incorrect data submissions.
     + Example: If a client-side form bypasses validation, the server ensures that only valid data is stored in the database.
2. **What is 3-tier architecture?**
   * **Answer**: 3-tier architecture divides an application into three layers for modularity and scalability:
     + **Presentation Layer**: User interface (e.g., browser, mobile app).
     + **Business Logic Layer**: Processes business rules and logic (e.g., API services).
     + **Data Layer**: Manages database operations (e.g., SQL database).
3. **What is the difference between web services and APIs?**
   * **Answer**:
     + **Web Services**:
       - Subset of APIs that use network protocols (e.g., HTTP, SOAP) to enable communication between applications.
       - Must be accessed over a network.
     + **APIs**:
       - Broader concept that allows communication between software components.
       - Can use various protocols like HTTP, TCP, or library-based communication.
4. **What is REST, SOAP, and GraphQL in APIs?**
   * **Answer**:
     + **REST (Representational State Transfer)**:
       - Stateless, uses HTTP methods (GET, POST, etc.).
       - Data is typically returned in JSON or XML.
     + **SOAP (Simple Object Access Protocol)**:
       - Uses XML and requires a strict protocol structure.
       - Heavier compared to REST.
     + **GraphQL**:
       - Query language for APIs that allows fetching only the required data.
       - More flexible compared to REST.
5. **What do you test in standalone APIs?**
   * **Answer**:
     + Functional correctness: Verify that the API returns the expected response for given inputs.
     + Error handling: Validate how the API handles invalid inputs.
     + Performance: Test response time under load.
     + Security: Check for vulnerabilities like unauthorized access.
     + Data integrity: Ensure consistent and accurate data processing.
6. **What do you test in 3rd-party integrated APIs?**
   * **Answer**:
     + Validate data synchronization between your application and the 3rd-party API.
     + Test authorization and authentication mechanisms (e.g., OAuth tokens).
     + Verify the API's handling of edge cases and errors.
     + Ensure the API meets SLA (Service Level Agreement) requirements for performance and uptime.

**Postman Questions:**

1. **When to use collections, environments, and global variables?**
   * **Answer**:
     + **Collections**: Group related API requests for better organization and sharing.
     + **Environments**: Store environment-specific variables (e.g., base\_url) for different setups like Dev, QA, Prod.
     + **Global Variables**: Use for values accessible across all collections (e.g., authentication tokens).
2. **How to execute a collection end-to-end?**
   * **Answer**: Use the **Runner** in Postman:
     + Open Postman, select the collection, and click on the **Runner**.
     + Configure the environment, iteration count, and delay.
     + Click **Run Collection** to execute all requests sequentially.
3. **How to validate that an API response has the correct status code?**
   * **Answer**: Add test scripts in the **Tests** tab:

javascript

Copy code

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

1. **What happens when an API response returns Form Data instead of JSON, and how to validate it?**
   * **Answer**:
     + Postman can handle Form Data using the pm.response.text() method:

javascript

Copy code

pm.test("Response is form data", function () {

pm.expect(pm.response.text()).to.include("expected\_value");

});

1. **How to set up Basic Auth in Postman?**
   * **Answer**:
     + Go to the **Authorization** tab of your request.
     + Select **Basic Auth** and enter the username and password.
2. **Where do you store environment credentials?**
   * **Answer**: Store credentials in environment variables to avoid hardcoding sensitive data.
3. **How to save a demo response for an API request?**
   * **Answer**:
     + Send the request, view the response, and click **Save Response** under the response section.
4. **How will you validate an API request if VPN is required for it to work?**
   * **Answer**:
     + Ensure your VPN is connected.
     + Use tools like Postman or cURL through the VPN tunnel to validate the request.
5. **How do you filter results in an API request using Postman?**
   * **Answer**:
     + Add query parameters in the **Params** tab to filter data (e.g., ?key=value).
     + Validate the response using Postman scripts.
6. **How to set up custom headers in Postman?**
   * **Answer**:
     + Go to the **Headers** tab in your request.
     + Add key-value pairs for the custom headers (e.g., Authorization: Bearer <token>).

**Git Questions:**

1. **What are the different stages in committing the code to GitHub?**
   * **Answer**:
     + **Working Directory**: Modified files.
     + **Staging Area**: Files added using git add.
     + **Repository**: Committed files after running git commit.
2. **Is it possible to revert changes in a remote repository? If yes, how?**
   * **Answer**: Yes, use:
     + **Revert**: Create a new commit to reverse changes:

bash

Copy code

git revert <commit\_hash>

* + - **Reset**: Forcefully undo changes (not recommended for shared branches):

bash

Copy code

git reset --hard <commit\_hash>

git push --force

1. **When do you commit your code? After committing, how do you validate that everyone has the updated code?**
   * **Answer**:
     + Commit your code after completing a logical set of changes.
     + Validate by pulling the latest code:

bash

Copy code

git pull origin <branch\_name>

1. **How to merge staged changes in a local repository?**
   * **Answer**:
     + Commit the staged changes:

bash

Copy code

git commit -m "Your commit message"

* + - Merge the branch:

bash

Copy code

git merge <branch\_name>

1. **Why do we need a .gitignore file? How do you add files to it?**
   * **Answer**:
     + **Purpose**: To exclude specific files or directories from being tracked by Git (e.g., logs, temporary files, credentials).
     + Add patterns or file names to .gitignore:

bash

Copy code

node\_modules/

\*.log

**JMeter Questions with Detailed Answers:**

1. **What is JMeter, and why is it used?**
   * **Answer**: JMeter is an open-source performance testing tool primarily used for load testing, functional testing, and stress testing web applications, APIs, and database servers. It simulates a heavy load of users and analyzes the performance of applications under test.
2. **What are the key components of a JMeter Test Plan?**
   * **Answer**:
     + **Thread Group**: Defines the number of users and the load.
     + **Samplers**: Represent different types of requests (HTTP, FTP, JDBC, etc.).
     + **Listeners**: Collect and display results.
     + **Timers**: Control delays between requests.
     + **Assertions**: Validate response data.
     + **Pre-Processors/Post-Processors**: Add custom logic before or after requests.
3. **How can you record a test scenario using JMeter?**
   * **Answer**:
     + Install the JMeter HTTP(S) Test Script Recorder.
     + Configure the proxy settings in your browser to use JMeter.
     + Record actions by navigating the application.
     + Save the recorded script for further testing.
4. **What are Thread Groups in JMeter, and why are they important?**
   * **Answer**:
     + Thread Groups define the number of virtual users, ramp-up time, and duration for the test.
     + It is the starting point of any JMeter test and controls the load simulation.
5. **What is the difference between Samplers and Listeners in JMeter?**
   * **Answer**:
     + **Samplers**: Generate requests to the server (e.g., HTTP Request Sampler).
     + **Listeners**: Collect and display responses (e.g., Summary Report, Graph Results).
6. **How do you perform parameterization in JMeter?**
   * **Answer**:
     + Use a **CSV Data Set Config** to read test data from a file.
     + Reference variables in requests using ${variable\_name}.
7. **How do you handle dynamic data or correlation in JMeter?**
   * **Answer**:
     + Use a **Regular Expression Extractor** to extract dynamic values from server responses.
     + Pass the extracted values to subsequent requests using variables.
8. **What is the use of Assertions in JMeter, and how do you implement them?**
   * **Answer**:
     + Assertions validate response data (e.g., Response Code, Response Time).
     + Add an **Assertion** (e.g., Response Assertion) and set validation conditions.
9. **How can you test a database with JMeter?**
   * **Answer**:
     + Use the **JDBC Request Sampler**.
     + Configure a JDBC Connection Configuration with the database details.
     + Write SQL queries to execute and validate.
10. **What is the role of timers in JMeter, and how do they affect performance tests?**
    * **Answer**:
      + Timers add a delay between requests to simulate real-world user behavior.
      + Common Timers: Constant Timer, Gaussian Random Timer.
11. **What are Pre-Processors and Post-Processors in JMeter? Provide examples.**
    * **Answer**:
      + **Pre-Processors**: Execute logic before the request (e.g., User Parameters).
      + **Post-Processors**: Execute logic after the request (e.g., Regular Expression Extractor).
12. **What is distributed testing in JMeter, and how do you set it up?**
    * **Answer**:
      + Distributed testing uses multiple machines to simulate a higher load.
      + Configure a master-slave setup by editing the jmeter.properties file and starting the remote server.
13. **How can you monitor real-time performance metrics while running a JMeter test?**
    * **Answer**:
      + Use tools like JMeter's **Backend Listener** to send metrics to monitoring tools (e.g., Grafana).
      + Use plugins like **PerfMon** for real-time monitoring.
14. **How do you generate and analyze JMeter reports?**
    * **Answer**:
      + Use the **Dashboard Report** by enabling it in the test plan.
      + Analyze metrics like throughput, response time, error rate, and latency.

**SQL Questions with Detailed Answers:**

1. **What is the difference between SQL and MySQL?**
   * **Answer**:
     + SQL: A language for managing relational databases.
     + MySQL: A relational database management system that uses SQL.
2. **What are the different types of SQL commands?**
   * **Answer**:
     + **DML**: Data Manipulation Language (e.g., INSERT, UPDATE, DELETE).
     + **DDL**: Data Definition Language (e.g., CREATE, ALTER, DROP).
     + **DCL**: Data Control Language (e.g., GRANT, REVOKE).
     + **TCL**: Transaction Control Language (e.g., COMMIT, ROLLBACK).
     + **DQL**: Data Query Language (e.g., SELECT).
3. **What is the difference between WHERE and HAVING clauses?**
   * **Answer**:
     + **WHERE**: Filters rows before aggregation.
     + **HAVING**: Filters aggregated data.
4. **What is a JOIN in SQL? Explain different types of JOINs.**
   * **Answer**:
     + JOIN combines rows from multiple tables based on a condition.
     + Types:
       - **INNER JOIN**: Matches rows in both tables.
       - **LEFT JOIN**: All rows from the left table and matching rows from the right table.
       - **RIGHT JOIN**: All rows from the right table and matching rows from the left table.
       - **FULL JOIN**: All rows from both tables.
5. **How do you fetch the first N records from a table?**
   * **Answer**:

sql

Copy code

SELECT \* FROM table\_name LIMIT N;

1. **How do you find duplicate records in a table?**
   * **Answer**:

sql

Copy code

SELECT column\_name, COUNT(\*)

FROM table\_name

GROUP BY column\_name

HAVING COUNT(\*) > 1;

1. **What is the difference between TRUNCATE, DELETE, and DROP?**
   * **Answer**:
     + **TRUNCATE**: Removes all rows from a table but keeps the structure.
     + **DELETE**: Removes specific rows and can use WHERE.
     + **DROP**: Removes the entire table, including structure.
2. **What is a transaction in SQL, and how do you implement it?**
   * **Answer**:
     + A transaction is a sequence of operations performed as a single unit.
     + Use:

sql

Copy code

BEGIN;

UPDATE table\_name SET column=value WHERE condition;

COMMIT;

1. **What is the ACID property of transactions?**
   * **Answer**:
     + **Atomicity**: All operations complete or none.
     + **Consistency**: Data remains consistent after a transaction.
     + **Isolation**: Transactions are independent.
     + **Durability**: Changes persist after a transaction.
2. **How do you optimize SQL queries for better performance?**
   * **Answer**:
     + Use indexes on frequently queried columns.
     + Avoid SELECT \*; fetch only required columns.
     + Use EXPLAIN to analyze query plans.
     + Optimize JOINs and WHERE conditions.