Selenium Handling Alerts and Pop-ups

"**Alerts and Pop-ups**" are important components in UI automation and testing (especially in Selenium and other UI testing tools), as they often interrupt or alter user flow. Here's a detailed explanation, followed by examples for automation handling:

🔹 Types of Alerts/Pop-ups in Web Applications

1. **JavaScript Alerts (Browser Alerts)**
    - **Alert** – Shows a message with "OK" button.
    - **Confirm** – Shows message with "OK" and "Cancel" buttons.
    - **Prompt** – Shows message with input field, "OK", and "Cancel" buttons.
2. **HTML Pop-ups / Modals**
    - Created using HTML/CSS/JS (e.g., Bootstrap modals).
    - Elements can be inspected using DevTools.
3. **Authentication Pop-ups**
    - Browser pop-ups requiring username/password (Basic Auth).
4. **File Upload/Download Pop-ups**
    - Native OS pop-ups (need workarounds, handled with tools like Robot, AutoIt, etc.).

🔹 Handling JavaScript Alerts in Selenium (Java)

✅ Alert Example:

```java
import org.openqa.selenium.Alert;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class AlertExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("https://example.com/alert");

        // Switch to alert
        Alert alert = driver.switchTo().alert();

        // Accept alert (click OK)
        alert.accept();

        // Or: Dismiss alert (click Cancel)
        // alert.dismiss();

        // Or: Get alert text
        // String alertText = alert.getText();

        // Or: Send input if it's a prompt
        // alert.sendKeys("Test input");
    }
}
```

🔹 Handling HTML Pop-ups (Modal Dialogs)

java

```java
// Identify modal elements by locators (Inspect in DevTools)
WebElement closeButton = driver.findElement(By.id("closeModal"));
closeButton.click();
```

These are handled like normal web elements (using locators: `By.id`, `By.xpath`, etc.).

🔹 Handling Authentication Pop-ups

✅ Using URL injection (if not secured with modern browser restrictions):

java

```java
driver.get("https://username:password@example.com");
```

If this method doesn't work due to modern browser policies, use tools like:

- **AutoIt (Windows only)**
- **Robot class**
- **Selenium 4 DevTools API**

🔹 Handling File Upload Pop-ups

For `<input type="file">`:

java

```java
WebElement uploadBtn = driver.findElement(By.id("upload"));
uploadBtn.sendKeys("C:\\path\\to\\file.txt");
```

For native upload windows:

- Use `Robot` class or external tool like `AutoIt`.

🔹 Example: Robot Class for Native Alerts

java

```java
import java.awt.Robot;
import java.awt.event.KeyEvent;

Robot robot = new Robot();
```

```
robot.keyPress(KeyEvent.VK_ENTER);
robot.keyRelease(KeyEvent.VK_ENTER);
```

💡 Best Practices

- Always **wait** for the alert/pop-up using **explicit wait** (**`WebDriverWait`**) if it loads dynamically.
- Use try-catch to handle **`NoAlertPresentException`**.

Would you like a **Selenium mini-project** or a **step-by-step lab** that involves alerts and pop-ups handling?

Great! Here's a **step-by-step Selenium mini-project/lab** that focuses on **handling different types of alerts and pop-ups** using **Selenium WebDriver with Java**.

🧪 Mini Project: Automating Alerts & Pop-ups in a Web Application

🔧 Objective:

Automate the following:

1. JavaScript alert
2. JavaScript confirm box
3. JavaScript prompt box
4. HTML modal pop-up
5. File upload pop-up (input type)

🖥️ Tools Required:

- **Java JDK** (8 or above)
- **Eclipse / IntelliJ IDEA**
- **Selenium WebDriver**
- **ChromeDriver**
- (Optional: Maven/Gradle for dependency management)

📁 Project Structure:

```
css

popup_automation_project/
│
├── src/
│   └── alerts/
│       └── HandlePopups.java
│
```

📄 Step-by-Step Code: `HandlePopups.java`

```
java

package alerts;

import org.openqa.selenium.*;
import org.openqa.selenium.chrome.ChromeDriver;
import java.time.Duration;

public class HandlePopups {

    public static void main(String[] args) throws InterruptedException {
        // Set up ChromeDriver path
        System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

        WebDriver driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

        // Open a test page that contains different types of alerts
        driver.get("https://the-internet.herokuapp.com/javascript_alerts");

        // Maximize browser
        driver.manage().window().maximize();

        // 1️⃣ Handle Simple Alert
        driver.findElement(By.xpath("//button[text()='Click for JS Alert']")).click();
        Alert simpleAlert = driver.switchTo().alert();
        System.out.println("Alert Text: " + simpleAlert.getText());
        simpleAlert.accept();

        // 2️⃣ Handle Confirm Box (OK / Cancel)
        driver.findElement(By.xpath("//button[text()='Click for JS Confirm']")).click();
        Alert confirmAlert = driver.switchTo().alert();
        System.out.println("Confirm Text: " + confirmAlert.getText());
        confirmAlert.dismiss(); // or confirmAlert.accept();

        // 3️⃣ Handle Prompt Box (with input)
        driver.findElement(By.xpath("//button[text()='Click for JS Prompt']")).click();
        Alert promptAlert = driver.switchTo().alert();
        System.out.println("Prompt Text: " + promptAlert.getText());
        promptAlert.sendKeys("Selenium Test");
        promptAlert.accept();

        // 🔁 Go to another site for HTML modal popup and file upload
        driver.get("https://demoqa.com/modal-dialogs");

        // 4️⃣ Handle HTML Modal Pop-up
        driver.findElement(By.id("showSmallModal")).click();
        WebElement closeModalBtn = driver.findElement(By.id("closeSmallModal"));
        closeModalBtn.click();

        // 5️⃣ Handle File Upload Pop-up
        driver.get("https://demoqa.com/upload-download");
        WebElement uploadInput = driver.findElement(By.id("uploadFile"));
        uploadInput.sendKeys("C:\\path\\to\\your\\file.txt");

        // ✅ Done
        System.out.println("All pop-ups handled successfully!");

        Thread.sleep(3000);
        driver.quit();
    }
}
```

📝 Notes:

- Replace `"path/to/chromedriver"` with your actual ChromeDriver path.
- Replace `"C:\\path\\to\\your\\file.txt"` with the actual file path on your system.
- The websites used:
    - `https://the-internet.herokuapp.com/javascript_alerts`
    - `https://demoqa.com/modal-dialogs`
    - `https://demoqa.com/upload-download`

🧠 What You'll Learn:

- How to handle different alert types using `Alert` interface
- How to interact with HTML modals and regular elements
- How to handle file uploads using `sendKeys()`

Would you like me to turn this into a **Maven project** with dependencies and folder structure ready to run?

Perfect! Here's how to **prepare for an interview** based on **Alerts and Pop-ups handling in Selenium**, with:

✅ Interview Preparation: Alerts and Pop-ups in Selenium WebDriver

---

🔹 1. Key Interview Questions & Ideal Answers

❓Q1. What are the different types of pop-ups in web applications?

Answer:
There are mainly four types of pop-ups:

- **JavaScript alerts** – Native browser alerts (`alert`, `confirm`, `prompt`)
- **HTML-based pop-ups/modals** – Built using HTML, CSS, JS (can be located like any element)
- **Authentication pop-ups** – Browser-level login prompts (e.g., HTTP Basic Auth)
- **File upload/download pop-ups** – OS-level native windows for file browsing

---

❓Q2. How do you handle JavaScript Alerts in Selenium WebDriver?

Answer:
JavaScript alerts are handled using the `Alert` interface.

```java
Alert alert = driver.switchTo().alert();
alert.accept();       // Clicks OK
alert.dismiss();      // Clicks Cancel
alert.getText();      // Retrieves alert message
alert.sendKeys("input");  // For prompt alerts
```

❓Q3. What is the difference between `accept()` and `dismiss()` in Selenium?

Answer:

- `accept()` is used to click **OK** on alerts.
- `dismiss()` is used to click **Cancel** on confirm or prompt alerts.

❓Q4. Can Selenium handle HTML pop-ups?

Answer:
Yes. HTML pop-ups are just web elements created using HTML and CSS. We handle them like regular web elements using locators:

```java
WebElement closeBtn = driver.findElement(By.id("closeModal"));
closeBtn.click();
```

❓Q5. How do you handle file upload pop-ups in Selenium?

Answer:
If the upload button is implemented using `<input type="file">`, we can use:

```java
WebElement upload = driver.findElement(By.id("uploadFile"));
upload.sendKeys("C:\\Users\\koti\\Documents\\resume.pdf");
```

If it's a native OS pop-up (not accessible via DOM), we use tools like:

- `Robot` class (Java AWT)
- `AutoIt` (Windows)
- `Sikuli`

❓Q6. How do you handle authentication pop-ups?

Answer:
Use **URL-based** authentication (if allowed):

```java
driver.get("https://username:password@yoururl.com");
```

For modern browsers that block this, use **Selenium DevTools (CDP)** or external tools.

❓Q7. What exception do you get if no alert is present but you try to switch to it?

Answer:
`NoAlertPresentException`

❓Q8. How do you validate the text present in an alert?

Answer:

```
java

Alert alert = driver.switchTo().alert();
String alertText = alert.getText();
Assert.assertEquals(alertText, "Expected alert message");
```

❓Q9. How do you wait for an alert to appear in Selenium?

Answer:

```
java

WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.alertIsPresent());
```

🛠️ Hands-on Practice Summary:

Task	Approach
Handle alert	**`driver.switchTo().alert().accept()`**
Handle confirm	**`alert.dismiss()`**
Handle prompt	**`alert.sendKeys("test"); alert.accept();`**
Handle modal (HTML) pop-up	Use **`By.id`**, **`By.xpath`**, **`By.cssSelector`**
Handle file upload	**`input.sendKeys("C:\\file.txt")`**
Handle authentication pop-up	**`https://user:pass@url.com`**

📌 Tips for Interview:

- Clarify whether it’s a JavaScript alert or HTML modal.
- Emphasize the use of the **`Alert`** interface for browser pop-ups.
- Mention **`Robot`** and **`AutoIt`** if asked about OS-level pop-ups.
- Always include real-world examples if asked "how you used it in your project".

Would you like **mock interview questions** or **project-based scenario questions** next?