# Selenium File Upload and download

File upload and download are common tasks in web automation. Selenium can handle both, but how it's done depends on the type of elements (HTML) involved and how the system handles file dialogs.

## 🔼 File Upload in Selenium Java

### ✅ Case 1: Upload using `<input type="file">` (most common)

### 💡 Explanation:

- Selenium can directly send file paths to `<input type="file">`.
- It bypasses the OS file dialog.

### ✅ Code Example:

```
java

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class FileUploadExample {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");  // set correct
path
        WebDriver driver = new ChromeDriver();

        driver.get("https://www.w3schools.com/howto/howto_html_file_upload_button.asp");

        // Locate the input type="file" element and upload file
        driver.findElement(By.id("myFile")).sendKeys("/path/to/your/file.txt");

        // Optional: Click upload button if needed
        driver.findElement(By.xpath("//input[@type='submit']")).click();

        driver.quit();
    }
}
```

### ⚠️ Notes:

- The file path must be absolute.
- This works **only if** the file input is not hidden. If hidden, you may need JavaScript to interact.

## ⬇️ File Download in Selenium Java

### 💡 Explanation:

- Selenium **cannot** interact with OS-level download dialogs.
- Instead, configure the **browser settings** to automatically download files.

## ✅ Code Example: Download a file using ChromeOptions

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import java.util.HashMap;
import java.util.Map;

public class FileDownloadExample {
    public static void main(String[] args) {
        String downloadFilepath = "/path/to/download/folder";

        // Set Chrome preferences for auto-download
        Map<String, Object> prefs = new HashMap<>();
        prefs.put("download.default_directory", downloadFilepath);
        prefs.put("download.prompt_for_download", false);
        prefs.put("profile.default_content_settings.popups", 0);

        ChromeOptions options = new ChromeOptions();
        options.setExperimentalOption("prefs", prefs);

        System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
        WebDriver driver = new ChromeDriver(options);

        // Replace with a URL that triggers download
        driver.get("https://file-examples.com/index.php/sample-documents-download/sample-pdf-
download/");

        // Click the download link/button
        driver.findElement(By.xpath("//a[contains(text(),'Download sample pdf
file')]")).click();

        // Wait for file download (optional logic for checking file existence)
        try { Thread.sleep(5000); } catch (InterruptedException e) {}

        driver.quit();
    }
}
```

---

# 🧠 Extra Tips

### For Uploads:

- If `<input type="file">` is not present and a **custom file picker** is used, Selenium **can't handle OS file picker dialogs**. You'll need:
  - **Robot Class** (for keyboard simulation).
  - **AutoIT** (for Windows GUI automation).

### For Downloads:

- You can verify if the file is downloaded using Java's `File` class:

```java
File file = new File(downloadPath + "/yourFile.pdf");
if(file.exists()) {
    System.out.println("Download Successful");
}
```

## ✅ Summary Table

| Task | Approach | Selenium Handles | Extra Tools Needed |
|---|---|---|---|
| File Upload | **`sendKeys()`** on **`<input>`** | ✅ Yes | ❌ No |
| File Upload | Custom upload button | ❌ No | ✅ Robot / AutoIT |
| File Download | Set browser prefs | ✅ Yes | ❌ No |
| File Download | Handle OS save dialog | ❌ No | ✅ Robot / AutoIT |

If you want examples with **`Robot`** or **`AutoIT`**, or want this setup for Firefox or Edge as well, just ask!

Sure! Let's break down this **Selenium Java** code for file downloading — line by line — so it's beginner-friendly. This program is for **automating file download using Chrome** browser.

---

### 🔰 Purpose:

To **automatically download a file** using Chrome without showing the download dialog.

---

## ✅ Full Code Explained:

```java
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import java.util.HashMap;
import java.util.Map;
```

### 📌 What these do:

- These are **import statements** to bring in Selenium and Java utilities.
    - **`WebDriver`**: Interface to control browser.
    - **`ChromeDriver`**: Controls Chrome browser.
    - **`ChromeOptions`**: Used to customize how Chrome behaves.
    - **`HashMap`**, **`Map`**: Used to store browser settings.

---

```java
public class FileDownloadExample {
    public static void main(String[] args) {
```

### 📌 What this means:

- Starting point of the program. **`main()`** is where Java starts executing code.
- **`FileDownloadExample`** is the class name.

```java
        String downloadFilepath = "/path/to/download/folder";
```

## 📌 What this does:

- This is the **folder path** where you want the file to be downloaded.
- You must **replace this** with a valid path on your computer, like:

  ```java
  String downloadFilepath = "C:/Users/YourName/Downloads";
  ```

---

```java
        Map<String, Object> prefs = new HashMap<>();
        prefs.put("download.default_directory", downloadFilepath);
        prefs.put("download.prompt_for_download", false);
        prefs.put("profile.default_content_settings.popups", 0);
```

## 📌 Explanation:

- Here we create a **map of preferences** for Chrome browser.
- These settings will:
  - 📁 **`download.default_directory`**: Set the download location.
  - ❌ **`download.prompt_for_download`**: Avoid download popups.
  - ⛔ **`popups`**: Disable popups (deprecated but kept as safe option).

---

```java
        ChromeOptions options = new ChromeOptions();
        options.setExperimentalOption("prefs", prefs);
```

## 📌 What this does:

- We create a **`ChromeOptions`** object to **set custom options** for Chrome.
- Attach the **`prefs`** map to tell Chrome how to behave when downloading.

---

```java
        System.setProperty("webdriver.chrome.driver", "/path/to/chromedriver");
        WebDriver driver = new ChromeDriver(options);
```

## 📌 Explanation:

- **`System.setProperty`** tells Selenium **where your ChromeDriver executable is**.

  - Replace **`/path/to/chromedriver`** with the actual path.
- **`new ChromeDriver(options)`** launches Chrome with the download options we set earlier.

---

```java
        driver.get("https://file-examples.com/index.php/sample-documents-download/sample-pdf-download/");
```

### 📌 What this does:

- It opens the browser and **navigates to a website** that has sample PDF files for download.

---

```java
        driver.findElement(By.xpath("//a[contains(text(),'Download sample pdf file')]")).click();
```

### 📌 Explanation:

- It **finds the download link** for the PDF file on the page and clicks on it.
- We use XPath to find an anchor tag (**`<a>`**) that has the text **"Download sample pdf file"**.

---

```java
        try { Thread.sleep(5000); } catch (InterruptedException e) {}
```

### 📌 What this does:

- Pauses the script for 5 seconds to **give time for the file to download**.
- **`Thread.sleep(5000)`** means: "Wait for 5000 milliseconds (5 seconds)".
- Wrapped in **`try-catch`** block to handle interruption errors.

---

```java
        driver.quit();
```

### 📌 Final Step:

- Closes the browser after the download is complete.

---

## 🔁 Summary of the Flow

| Step | What Happens |
|---|---|
| 1 | Set download folder preferences for Chrome |

| Step | What Happens |
|---|---|
| 2 | Launch Chrome with those settings |
| 3 | Open the webpage with the download link |
| 4 | Click the download link |
| 5 | Wait for the download to complete |
| 6 | Close the browser |

---

## ✅ What You Must Customize:

| Variable | What You Need to Do |
|---|---|
| `downloadFilepath` | Replace with your actual folder path |
| `chromedriver` path | Replace with path to your downloaded ChromeDriver |

---

Would you like me to show you how to:

- Verify if the file was actually downloaded?
- Automatically create the folder if it doesn’t exist?

Let me know!

Great question! Let’s break down this piece of code in simple terms, **line by line**, so you understand exactly what it's doing and **why it's important** for file downloads in Selenium using Chrome.

---

## 🔍 Code Block:

```java
Map<String, Object> prefs = new HashMap<>();
prefs.put("download.default_directory", downloadFilepath);
prefs.put("download.prompt_for_download", false);
prefs.put("profile.default_content_settings.popups", 0);
```

This part is **creating a preferences map** (using a **`HashMap`**) to **tell Chrome how to behave** when downloading files.

---

## ✅ Line-by-Line Explanation:

### 🧩 Line 1:

```java
Map<String, Object> prefs = new HashMap<>();
```

- This creates a **Map** called **`prefs`** to store key-value pairs.
- The **`String`** is the **setting name** (e.g., **`"download.default_directory"`**).
- The **`Object`** is the **value for that setting** (e.g., the path to your download folder).

Think of it like a table of browser settings.

---

### 🧩 Line 2:

```java
prefs.put("download.default_directory", downloadFilepath);
```

- This sets the **default folder path where Chrome should save downloaded files**.
- **`downloadFilepath`** is a variable containing a valid folder path like:

  ```java
  String downloadFilepath = "C:/Users/YourName/Downloads";
  ```

- Without this, Chrome would show a "Save As" dialog, which Selenium can't handle.

---

### 🧩 Line 3:

```java
prefs.put("download.prompt_for_download", false);
```

- This tells Chrome **not to ask for confirmation** before downloading.
- Set to `false` so that the browser **automatically starts downloading** the file when the link is clicked.

Otherwise, you'd see a popup like “Do you want to save this file?” — which Selenium can't interact with.

---

### 🧩 Line 4:

```java
prefs.put("profile.default_content_settings.popups", 0);
```

- This disables any **popup windows** that might block downloads.
- Setting it to `0` means: **No popups allowed**.
- Although this setting is old (deprecated in newer Chrome versions), it’s still commonly included as a safety fallback.

---

## 🔁 Final Output

You attach this `prefs` map to Chrome like this:

```java
ChromeOptions options = new ChromeOptions();
options.setExperimentalOption("prefs", prefs);
```

So when you create the browser, it follows these rules and **downloads files quietly and automatically**.

---

## ✅ Real-world Example

Imagine you're testing an HR system where clicking "Download Payslip" generates a PDF file. This code makes sure:

- No dialogs block the process.
- The file goes straight to a known folder.
- You can later write Java code to check if the file exists.

---

Let me know if you'd like an example to **verify the downloaded file using Java** or to use the same setup with **Firefox** or **Edge**.

Here are **practical Selenium Java assignments** on file upload and download to help you master the concepts hands-on. Each one includes a clear **goal, requirements, hints**, and **bonus challenges**.

---

## 📝 Assignment 1: Basic File Upload

### ✅ Goal:

Upload a file using a form that uses `<input type="file">`.

### 🔗 Test Website:

https://www.w3schools.com/howto/howto_html_file_upload_button.asp

### 📌 Tasks:

- Open the website.
- Upload a file (e.g., a `.txt` or `.jpg`) using `sendKeys()`.
- Click the submit button (if present).

### 💡 Hint:

Use:

```java
driver.findElement(By.id("myFile")).sendKeys("/absolute/path/to/your/file.txt");
```

### ⭐ Bonus:

Take a screenshot after uploading the file.

---

## 📝 Assignment 2: File Upload with Hidden Input

### ✅ Goal:

Handle a hidden file input using JavaScript.

### 🔗 Test Website:

Build a small HTML file locally with:

```html
<input type="file" id="hiddenUpload" style="display: none;">
<button onclick="document.getElementById('hiddenUpload').click()">Upload</button>
```

### 📌 Tasks:

- Open the local HTML file.
- Click the "Upload" button using Selenium.
- Upload a file using **`sendKeys()`** after making the hidden input visible via JavaScript.

### 💡 Hint:

```java
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("document.getElementById('hiddenUpload').style.display='block';");
```

---

## 📝 Assignment 3: Basic File Download

### ✅ Goal:

Download a PDF file from a website using ChromeOptions.

### 🔗 Test Website:

https://file-examples.com/index.php/sample-documents-download/sample-pdf-download/

### 📌 Tasks:

- Set up ChromeOptions to download the file automatically.
- Navigate to the site and click the download link.
- Use Java code to check if the file exists in your download folder.

### 💡 Hint:

Use:

```java
File file = new File(downloadPath + "/sample.pdf");
if (file.exists()) {
    System.out.println("Download successful!");
}
```

---

## 📝 Assignment 4: File Download with Timestamp Check

### ✅ Goal:

Ensure the file was downloaded recently (not an old copy).

### 📌 Tasks:

- Reuse Assignment 3.

- After download, verify that the file's last modified time is within the last 1 minute.

### 💡 Hint:

Use:

```java
long lastModified = file.lastModified();
long now = System.currentTimeMillis();
if (now - lastModified < 60000) {
    System.out.println("File downloaded recently.");
}
```

---

## 📝 Assignment 5: Upload and Download in the Same Test

### ✅ Goal:

Combine upload and download in a single automated test.

### 🔗 Suggested Test Site (if none available):

Host your own small form using HTML on your machine, or use two test sites — one for upload, one for download.

### 📌 Tasks:

- First, upload a file.
- Then, download a file from a different page.
- Validate upload success (e.g., check text or alert).
- Validate download by checking the file.

---

## 📝 Assignment 6: Use Robot Class for File Upload

### ✅ Goal:

Simulate keyboard actions to handle OS file dialog (if `<input type="file">` is not available).

### 📌 Tasks:

- Visit a site that opens the OS file dialog (e.g., WordPress media uploader).
- Click the upload button.
- Use Java's `Robot` class to type the file path and hit Enter.

### 💡 Hint:

```java
Robot robot = new Robot();
robot.keyPress(KeyEvent.VK_C);  // For 'C:\'
robot.keyRelease(KeyEvent.VK_C);
// And so on...
```

## 🧠 Optional Bonus Challenges

| Challenge | Description |
|---|---|
| Retry Download | Retry clicking the download link if file is not downloaded in 5 sec |
| Multiple Files | Upload/download multiple files in a loop |
| Use Firefox | Repeat assignments using Firefox and **`FirefoxProfile`** |
| Dynamic Wait | Use **`WebDriverWait`** instead of **`Thread.sleep()`** |
| Headless Mode | Perform upload/download in Chrome Headless |

Would you like the **starter code templates or HTML files** for any of these assignments? I can generate them for you.