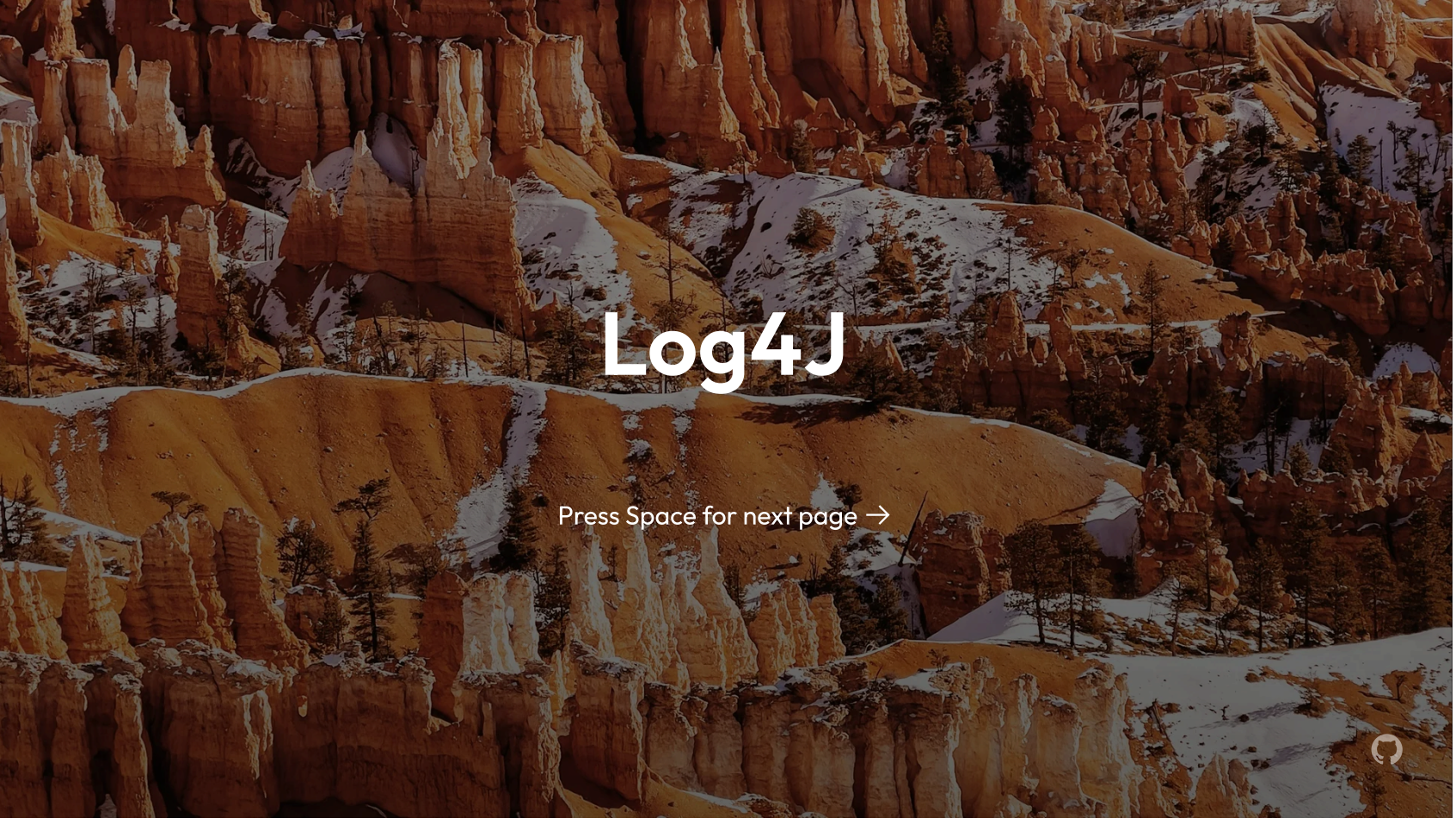

Log4J
Press Space for next page →

Agenda

1. Log4J
2. Adding Log4J to the Project
3. Log4J Setup and Syntax
4. Log4J Configuration
5. Logging Levels
6. Code Example: Log4J

Log4J

> Log4j is a fast, flexible and reliable logging framework written in Java. It is an open-source logging API for Java. It is managed using Apache Software Foundation.

Purpose and Importance

Logging is essential for tracking events that happen when some software runs. Log4j provides a sophisticated and flexible framework to produce logs in a variety of formats and destinations.

Adding Log4J to the Project

Link: https://mvnrepository.com/artifact/org.apache.logging.log4j

add the following dependency to the `pom.xml` file.

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.24.0</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.24.0</version>
</dependency>
```

Log4J Setup and Syntax

Import the following classes in the Java file.

```
import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;
```

Create a logger object using the following syntax.

```
private static final Logger logger = LogManager.getLogger(ClassName.class);
```

For logging, use the following syntax.

```
logger.info("This is an info message");
logger.error("This is an error message");
logger.warn("This is a warning message");
logger.debug("This is a debug message");
```

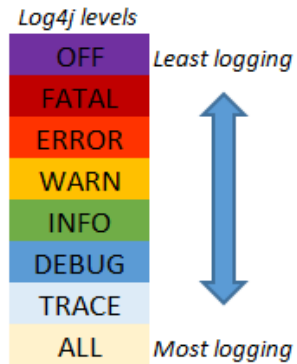
Log4J Configuration

Create a `log4j2.properties` file in the `src/test/resources` directory.

```
# Set the level of internal Log4j events.
status = error

# Define the configuration name
name = PropertiesConfig

# Console appender configuration
appender.console.type = Console
appender.console.name = ConsoleAppender
appender.console.layout.type = PatternLayout
appender.console.layout.pattern = %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n

# File appender configuration
appender.file.type = File
appender.file.name = FileAppender
appender.file.fileName = logs/nexxvali_automation_application.log
appender.file.layout.type = PatternLayout
appender.file.layout.pattern = %d{yyyy-MM-dd HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n

# Root logger configuration
rootLogger.level = info
rootLogger.appenderRefs = console, file
rootLogger.appenderRef.console.ref = ConsoleAppender
rootLogger.appenderRef.file.ref = FileAppender
```

Logging Levels

ALL < TRACE < DEBUG < INFO < WARN < ERROR < FATAL < OFF



Code Example: Log4J

```
class Log4JExample {
    WebDriver driver;
    private static final Logger log = LogManager.getLogger(Log4JExamp
    @BeforeMethod
    public void setup() {
        log.info("*************** Application starting ***************
        driver = new ChromeDriver();
        driver.get("https://www.saucedemo.com/");
        driver.manage().window().maximize();
        log.info("*************** Application started ***************
    }
    @Test(dataProvider ="sauceLabData")
    public void login(String username, String password) throws Interr
        driver.findElement(By.id("user-name")).sendKeys(username);
        log.info("Username entered");

        driver.findElement(By.name("password")).sendKeys(password);
        log.info("Password entered");

        driver.findElement(By.className("submit-button")).click();
        log.info("Clicked on login button");
        Thread.sleep(3000);
        String actual = null;
        try {
            actual = driver.findElement(By.xpath("//*[text()='Product
        } catch (Exception e) {
            log.error(e.getMessage());
```

```
@AfterMethod
public void tearDown() {
    log.info("*************** Application closing ***************");
    driver.quit();
    log.info("*************** Application closed ***************");
}

@DataProvider(name = "sauceLabData")
public Object[][] passData() {

    Object[][] data = new Object[0][0];
    try(FileInputStream inputStream = new FileInputStream(new File("S
        Workbook workbook = new XSSFWorkbook(inputStream);
        Sheet sheet = workbook.getSheet("Sheet1");

        int rowCount = sheet.getLastRowNum() - sheet.getFirstRowNum()
        data = new Object[rowCount][2];

        for (int i = 1; i <= rowCount; i++) {
            Row row = sheet.getRow(i);
            for (int j = 0; j <2; j++) {
                data[i - 1][j] = row.getCell(j).toString();
            }
        }
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
    return data;
}
```