Junit 5

Extension Model

We Cover

- Extension Design
- Creating Extensions
- Registering Extensions
- Test Lifecycle Callbacks
- Types of Extensions
- ExtensionContext

```
BeforeAllCallback (1)

    @BeforeAll (2)
    LifecycleMethodExecutionExceptionHandler
    #handleBeforeAllMethodExecutionException (3)

        BeforeEachCallback (4)

            @BeforeEach (5)
            LifecycleMethodExecutionExceptionHandler
            #handleBeforeEachMethodExecutionException (6)

                BeforeTestExecutionCallback (7)

                    @Test (8)
                    TestExecutionExceptionHandler (9)

                AfterTestExecutionCallback (10)

            @AfterEach (11)
            LifecycleMethodExecutionExceptionHandler
            #handleAfterEachMethodExecutionException (12)

        AfterEachCallback (13)

    @AfterAll (14)
    LifecycleMethodExecutionExceptionHandler
    #handleAfterAllMethodExecutionException (15)

AfterAllCallback (16)
```

Test Life Cycle

Extensions - The Basics

```java
public class ExtensionExample implements BeforeEachCallback {
    @Override
    public void beforeEach(ExtensionContext extensionContext) throws Exception {
        System.out.println("This is an extension example");
    }
}
```

```java
@ExtendWith(ExtensionExample.class)
public class ExtensionTest {

    @Test
    public void testExtension() {
        System.out.println("This is a test with an extension");
    }
}
```

```
This is an extension example
This is a test with an extension
```

Register Extensions

```java
@ExtendWith(ExtensionExample.class)
public class ExtensionTest {
}
```

```java
@ExtendWith(ExtensionExample.class)
@Test
public void testExtension() {
        System.out.println("This is a test with an extension");
}
```

```java
@ExtendWith(ExtensionExample.class)
@ExtendWith(AnotherExtensionExample.class)
public class ExtensionTest {
}
```

```java
@ExtendWith({ExtensionExample.class, AnotherExtensionExample.class})
public class ExtensionTest {
}
```

Registering Extensions

```java
@Target({ ElementType.TYPE, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@ExtendWith({ DatabaseExtension.class, WebServerExtension.class })
public @interface DatabaseAndWebServerExtension {
}
```

```java
@ExtendWith(ExtensionExample.class)
@DatabaseAndWebServerExtension
public class ExtensionTest {

    @Test
    public void testExtension() {
        System.out.println("This is a test with an extension");
    }
}
```

Registering Extensions

```java
public class ExtensionTest {

    @RegisterExtension
    static ExtensionExample extensionExample = new ExtensionExample();

    @Test
    public void testExtension() {
        System.out.println("This is a test with an extension");
    }
}
```

```
This is an extension example
This is a test with an extension
```

Extension Life Cycle

```
public class ExtensionExample implements
        BeforeAllCallback,
        BeforeEachCallback,
        BeforeTestExecutionCallback,
        AfterTestExecutionCallback,
        AfterEachCallback,
        AfterAllCallback {
}
```

Registering Extensions - Programmatically

```java
public class ProgrammingExtensionTest {

    @RegisterExtension
    static ExtensionExample extension = new ExtensionExample();

    @Test
    public void extensionExample(){
        System.out.println("This is a test with an extension");
    }
}
```

```
This is an extension example - beforeAll
This is an extension example - beforeEach
This is an extension example - beforeTestExecution
This is a test with an extension
This is an extension example - afterTestExecution
This is an extension example - afterEach
This is an extension example - afterAll
```

Registering Global Extensions - META-INF

Steps:

(1) Create Extension

(2) Registern Extension in META-INF/services folder

(3) Allow Auto Detection of extensions

META-INF/services file name: `org.junit.jupiter.api.extension.Extension`

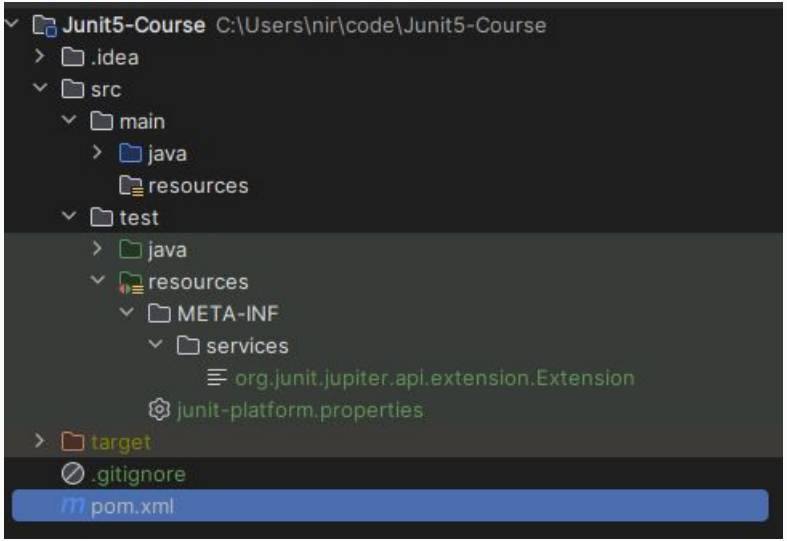
Global configuration file name: `junit-platform.properties`

Registering Global Extensions - META-INF

```
package ai.verisoft;

import org.junit.jupiter.api.extension.*;

public class ExtensionExample implements
        BeforeAllCallback,
        BeforeEachCallback,
        BeforeTestExecutionCallback,
        AfterTestExecutionCallback,
        AfterEachCallback,
        AfterAllCallback {
}
```



Registering Global Extensions - META-INF

junit-platform.properties

```
junit.jupiter.extensions.autodetection.enabled=true
```

Test result processing

```java
public class TestResultProcessingExtension implements AfterEachCallback {

    @Override
    public void afterEach(ExtensionContext context) {
        if(context.getExecutionException().isPresent()) {
            System.out.println("Test failed: " + context.getDisplayName());
        } else {
            System.out.println("Test passed: " + context.getDisplayName());
        }
    }
}
```

Context Object

```java
public class TestResultProcessingExtension implements AfterEachCallback {

    @Override
    public void afterEach(ExtensionContext context) {
        if(context.getExecutionException().isPresent()) {
            System.out.println("Test failed: " + context.getDisplayName());
        } else {
            System.out.println("Test passed: " + context.getDisplayName());
        }
    }
}
```

Context Object

Method Name	Description
getTestInstance()	Retrieves the test instance for the current test or container.
getTestMethod()	Retrieves the test method for the current test or container.
getDisplayName()	Retrieves the display name for the current test or container.
getUniqueId()	Retrieves the unique ID for the current test or container.
getTags()	Retrieves the tags associated with the current test or container.
getParent()	Retrieves the parent ExtensionContext.
getStore(Namespace)	Provides access to the Store for storing and retrieving data associated with the current context.
getElement()	Retrieves the annotated element for the current test or container.
getRequiredTestClass()	Retrieves the test class for the current test or container.
getRequiredTestMethod()	Retrieves the test method for the current test or container.

Native Junit Dependency Injection

What is Dependency Injection?

Native Junit 5 DP Objects

- Native Junit 5 dependency injection
  - TestInfo object
  - RepetitionInfo object
  - TestReporter object
  - Extension Model
    - ExtensionContext object
    - Extension model

Parameter Resolver

```java
public class ParameterResolverDemoExtension implements ParameterResolver {
    @Override
    public boolean supportsParameter(ParameterContext parameterContext,
                                     ExtensionContext extensionContext)
                                     throws ParameterResolutionException {

      // Does the given parameter can be resolved with this extension?
      return true;
    }

    @Override
    public Object resolveParameter(ParameterContext parameterContext,
                                   ExtensionContext extensionContext)
                                   throws ParameterResolutionException {
        // Resolve the parameter
        return null;
    }
}
```

Parameter Resolver - Example

```
public class RandomNumberExtension implements
    BeforeAllCallback,
    BeforeEachCallback,
    ParameterResolver {
}
```

Parameter Resolver - Example

```
@Target({ ElementType.FIELD, ElementType.PARAMETER })
@Retention(RetentionPolicy.RUNTIME)
@ExtendWith(RandomNumberExtension.class)
public @interface Random {
}
```

Parameter Resolver

```java
public class RandomNumberDemoTest {

    @Random
    private static Integer randomNumber0;

    @Random
    private int randomNumber1;

    RandomNumberDemoTest(@Random int randomNumber2) {
        System.out.println("Constructor randomNumber2 = " + randomNumber2);
    }

    @BeforeEach
    void beforeEach(@Random int randomNumber3) {
        System.out.println("BeforeEach randomNumber3 = " + randomNumber3);
    }

    @Test
    void test(@Random int randomNumber4) {
        System.out.println("Test randomNumber4 = " + randomNumber4);
    }
}
```

```
Constructor randomNumber2 = -1422242968
BeforeEach randomNumber3 = 306177365
Test randomNumber4 = -1411721409
```

TestWatcher Extension

```java
public class TestWatcherExtensionExample implements TestWatcher {

    @Override
    public void testSuccessful(ExtensionContext context) {
    }

    @Override
    public void testFailed(ExtensionContext context, Throwable cause) {
    }

    @Override
    public void testAborted(ExtensionContext context, Throwable cause) {
    }

    @Override
    public void testDisabled(ExtensionContext context, Throwable cause) {
    }
}
```

```
BeforeAllCallback (1)

    @BeforeAll (2)
    LifecycleMethodExecutionExceptionHandler
    #handleBeforeAllMethodExecutionException (3)

        BeforeEachCallback (4)

            @BeforeEach (5)
            LifecycleMethodExecutionExceptionHandler
            #handleBeforeEachMethodExecutionException (6)

                BeforeTestExecutionCallback (7)

                    @Test (8)
                    TestExecutionExceptionHandler (9)

                AfterTestExecutionCallback (10)

            @AfterEach (11)
            LifecycleMethodExecutionExceptionHandler
            #handleAfterEachMethodExecutionException (12)

        AfterEachCallback (13)

    @AfterAll (14)
    LifecycleMethodExecutionExceptionHandler
    #handleAfterAllMethodExecutionException (15)

AfterAllCallback (16)
```

Test Life Cycle

TestExecutionExceptionHandler Extension

```java
public class IgnoreIOExceptionExtension implements TestExecutionExceptionHandler {

    @Override
    public void handleTestExecutionException(ExtensionContext context, Throwable throwable)
            throws Throwable {

        if (throwable instanceof IOException) {
            return;
        }
        throw throwable;
    }
}
```

TestExecutionExceptionHandler Extension

```
public class RecordStateOnErrorExtension implements LifecycleMethodExecutionExceptionHandler {

    @Override
    public void handleBeforeAllMethodExecutionException(ExtensionContext context, Throwable ex)
            throws Throwable {
    }
    @Override
    public void handleBeforeEachMethodExecutionException(ExtensionContext context, Throwable ex)
            throws Throwable {
    }
    @Override
    public void handleAfterEachMethodExecutionException(ExtensionContext context, Throwable ex)
            throws Throwable {
    }
    @Override
    public void handleAfterAllMethodExecutionException(ExtensionContext context, Throwable ex)
            throws Throwable {
    }
}
```

Keeping State in Extensions

- Usually, an extension is instantiated only once
- How to keep the state from one invocation to the next?
- The answer is: `Store` object
- `Store` works with name spaces to separate context behvaior

Putting a value in the store

```java
public class TimingExtension implements BeforeTestExecutionCallback, AfterTestExecutionCallback {
    private static final Logger logger = Logger.getLogger(TimingExtension.class.getName());

    private static final String START_TIME = "start time";

    private Store getStore(ExtensionContext context) {
        return context.getStore(Namespace.create(getClass(), context.getRequiredTestMethod()));
    }

    @Override
    public void beforeTestExecution(ExtensionContext context) throws Exception {
        getStore(context).put(START_TIME, System.currentTimeMillis());
    }
}
```

Retrieving a value from the store

```java
@Override
public void afterTestExecution(ExtensionContext context) throws Exception {
    Method testMethod = context.getRequiredTestMethod();
    long startTime = getStore(context).remove(START_TIME, long.class);
    long duration = System.currentTimeMillis() - startTime;

    logger.info(() ->
            String.format("Method [%s] took %s ms.", testMethod.getName(), duration));
}
```

Missing Pieces in the life cycle

- BeforeSuite - running only once
- AfterAll - running only once

Suite Level Extension

```java
public class SuiteLevelExtension implements BeforeAllCallback, ExtensionContext.Store.CloseableResource {

    private static boolean didRun = false;

    @Override
    public void beforeAll(ExtensionContext extensionContext) throws Exception {

        if (!didRun) {
            extensionContext.getRoot().getStore(ExtensionContext.Namespace.GLOBAL).put("ExtensionCallback", this);
            System.out.println("This is a suite level extension - beforeAll");
            didRun = true;
        }
    }

    @Override
    public void close() {
        System.out.println("This is a suite level extension - close");
    }
}
```

Invocation Interception

- Can invoke tests programmatically
- Advanced topic, just know that it's there
- More on the issue: https://junit.org/junit5/docs/current/user-guide/#extensions-intercepting-invocations

Best Practice

Extension	Name	Before Suite	Before All	Before Each	After Each	After All	After Suite
Log	LogExtension	Create log instance		Log - start test	Log - test end. Result is?		Close the log object
Report	ReportExtension	Connect to reporting server	Create a new section for test class			Close section	Close the report object, Disconnect from server
Database	DBExtension	Connect to DB		Retrieve relevant information for test			Close connection to DB
Services	ServiceExtension	Fire up server		Activate service			Stop server

The End