Extending JUnit

Esteban Herrera
JAVA ARCHITECT

@eh3rrera   www.eherrera.net

Overview

Extension points

Parameter resolution

Meta-annotations

Keeping state

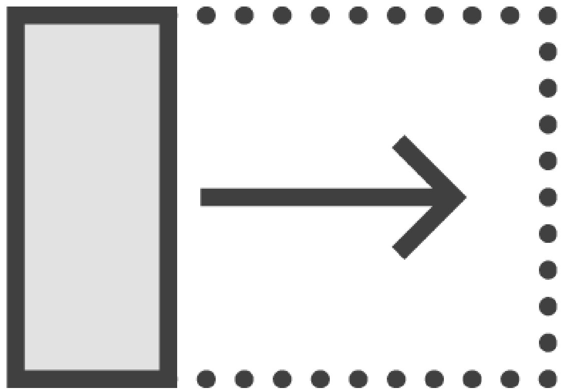
Sample extensions

Extension Points

“Prefer extension points over features.”

JUnit design principles

Extension Points



Marker interface ⟶ Extension

One extension point ⟶ one interface

Called by the JUnit Jupiter engine

General Purpose

TestInstancePostProcessor

ParameterResolver

TestExecutionExceptionHandler

Conditional

ExecutionCondition

Lifecycle Callbacks

- ✓ BeforeAllCallback / AfterAllCallback
- ✓ BeforeEachCallback / AfterEachCallback
- ✓ BeforeTestExecutionCallback / AfterTestExecutionCallback

Extension Registration

Explicit

- @ExtendWith

Global

- java.util.ServiceLoader mechanism
  - /META-INF/services
  - org.junit.jupiter.api.extension.Extension
  - junit.jupiter.extensions.autodetection.enabled

Demo

Implementing an extension

- Lifecycle callbacks

Parameter Injection

Parameter Injection

Test information parameters

- RepetitionInfo
- TestInfo
- TestReporter

ParameterResolver

- RepetitionInfoParameterResolver
- TestInfoParameterResolver
- TestReporterParameterResolver

```
boolean supportsParameter(ParameterContext parameterContext,
                          ExtensionContext extensionContext)
        throws ParameterResolutionException


Object resolveParameter(ParameterContext parameterContext,
                        ExtensionContext extensionContext)
        throws ParameterResolutionException
```

ParameterResolver

Methods That Can Have Parameters Injected

@Test

@TestFactory

Lifecycle Methods

Constructors

Demo

ParameterResolver extension

Meta-annotations

Meta-annotations

```java
@Test
void testRewardProgram() {
  ...
}
```

Meta-annotations

```
@TestWithErrorHandler
void testRewardProgram() {
   ...
}
```

Meta-annotations

```
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Test
@ExtendWith({ExceptionHandler.class})
public @interface TestWithErrorHandler() { }
```

```
@TestWithErrorHandler
void testRewardProgram() {
  // ...
}
```

Demo

Meta-annotations

Keeping State

Extensions have to be stateless

Storing State

Store

- Key-value structure

Namespace

Hierarchy

- MethodExtensionContext
- ClassExtensionContext
- JupiterEngineExtensionContext

```
Object get(Object key)

<K,V> Object getOrComputeIfAbsent(K key, Function<K,V> defaultCreator)

void put(Object key, Object value)

Object remove(Object key)
```

ExtensionContext.Store

Demo

Implementing an extension

- Store
- ExecutionCondition

Sample Extensions

Summary

Extension points

- One extension point ⟶ one interface
- Register explicitly or globally
- Lifecycle callbacks

Parameter resolution

- ParameterResolver

Meta-annotations

Keeping state

- Extensions are stateless
- Store and namespaces
- Hierarchical contexts

Sample extensions