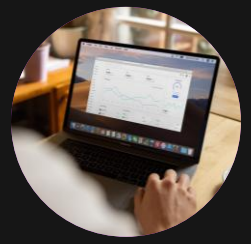
TestContainers (Browser): Navegadores en contenedores para pruebas E2E

Elena Ceinos Abeijón



¿Por qué usar TestContainers?

Ventajas

1. Entornos reproducibles
2. No requiere instalación de navegadores
3. Aislamiento total

1. Ideal para CI/CD
2. Integración sencilla

Introducción

Test Containers

- Librería para lanzar y controlar contenedores Docker desde el código de test.
- Ideal para pruebas de integración y sistema en entornos reproducibles

TestContainers (Browser)

- Permite ejecutar navegadores reales (Chrome, Firefox) en contenedores.
- Facilita pruebas E2E de interfaces web con Selenium o Selenide.

¿Por qué usarlo?

- Aislamiento completo, sin dependencias locales. Entorno reproducible y fácil de integrar en CI/CD

Cómo funcionaTest Containers

Browser

Se lanza el test1

Por ejemplo, con JUnit

TestConainers

Crea un contenedor Docker con navegador (Chrome/Firefox)

Selenium o Selenide

interactúa con el navegador dentro del contenedor

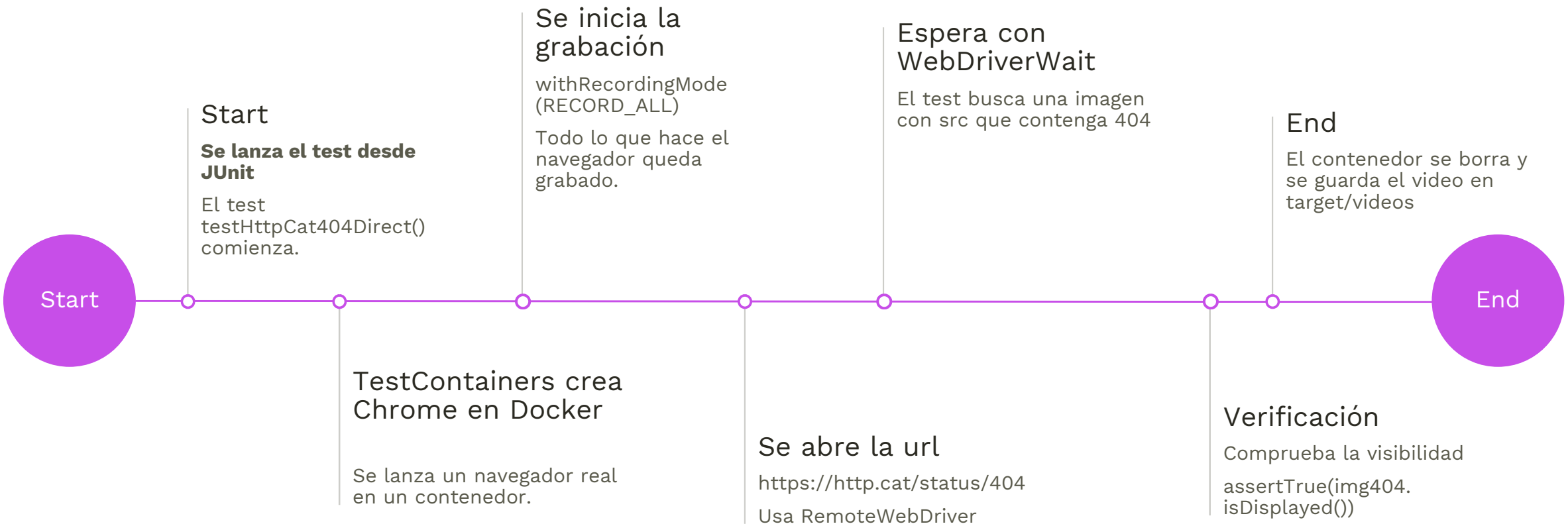
El test realiza acciones

en la interfaz web

El contenedor se destruye

automáticamente al finalizar el test

Timeline


Start
Start
Se lanza el test desde JUnit
El test testHttpCat404Direct() comienza.
TestContainers crea Chrome en Docker
Se lanza un navegador real en un contenedor.
Se inicia la grabación
withRecordingMode (RECORD_ALL)
Todo lo que hace el navegador queda grabado.
Se abre la url
https://http.cat/status/404
Usa RemoteWebDriver
Espera con WebDriverWait
El test busca una imagen con src que contenga 404
Verificación
Comprueba la visibilidad
assertTrue(img404. isDisplayed())
End
El contenedor se borra y se guarda el video en target/videos
End

Video

TestContainers Demo

Repositorio

GitHub

Desglose de código

```
import java.io.File;
import java.net.URL;
import java.time.Duration;

import static org.junit.jupiter.api.Assertions.assertTrue;
import org.junit.jupiter.api.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.testcontainers.containers.BrowserWebDriverContainer;
import org.testcontainers.junit.jupiter.Container;
import org.testcontainers.junit.jupiter.Testcontainers;
```

- Java/JUnit
  - import java.io.File;
  - import java.net.URL;
  - import java.time.Duration;

  - import static org.junit.jupiter.api.Assertions.assertTrue;
  - import org.junit.jupiter.api.Test;

- Selenium
  - import org.openqa.selenium.By;
  - import org.openqa.selenium.WebDriver;
  - import org.openqa.selenium.WebElement;
  - import org.openqa.selenium.chrome.ChromeOptions;
  - import org.openqa.selenium.remote.RemoteWebDriver;
  - import org.openqa.selenium.support.ui.ExpectedConditions;
  - import org.openqa.selenium.support.ui.WebDriverWait;

- Test containers
  - import org.testcontainers.containers.BrowserWebDriverContainer;
  - import org.testcontainers.junit.jupiter.Container;
  - import org.testcontainers.junit.jupiter.Testcontainers;

Desglose de código

```java
@Testcontainers
public class TestContainersBrowserTest {

    static {
        new File(pathname:"target/videos").mkdirs();
    }

    @Container
    public BrowserWebDriverContainer<?> container = new BrowserWebDriverContainer<>()
            .withCapabilities(new ChromeOptions())
            .withRecordingMode(
                BrowserWebDriverContainer.VncRecordingMode.RECORD_ALL,
                new File(pathname:"target/videos")
            );

    @Test
    void testHttpCat404Direct() throws Exception {
        // Crear el WebDriver manualmente usando la URL remota
        WebDriver driver = new RemoteWebDriver(
            new URL(container.getSeleniumAddress().toString()),
            new ChromeOptions()
        );
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(seconds:10));

        driver.get(url:"https://http.cat/status/404");
        WebElement img404 = wait.until(ExpectedConditions.visibilityOfElementLocated(
            By.xpath(xpathExpression:"//img[contains(@src,'404')]")
        ));
        assertTrue(img404.isDisplayed(), "La imagen del gato 404 debe estar visible");
        driver.quit();
    }
}
```

- @Testcontainers
- static { new File("target/videos").mkdirs(); }
- @Container BrowserWebDriverContainer<?> container
- @Test void testHttpCat404Direct()
- RemoteWebDriver driver = new RemoteWebDriver(...)
- driver.get("https://http.cat/status/404")
- WebDriverWait wait = new WebDriverWait(...)
- By.xpath("//img[contains(@src,'404')]")
- assertTrue(img404.isDisplayed())
- driver.quit();