DP2 2021-2022

Planning report

Repositorio:

https://github.com/mpadillatabuenca/Acme-Toolkit.git

Miembros:

- José Manuel Bejarano Pozo (josbezpoz@alum.us.es)
- Mario Espinosa Rodríguez (maresprod5@alum.us.es)
- Andrea Meca Sánchez (andmecsan@alum.us.es)
- Manuel Padilla Tabuenca (maresprod5@alum.us.es)
- Ezequiel Pérez Sosa (ezepersos@alum.us.es)
- Javier Terroba Orozco (javteroro@alum.us.es)

GRUPO G1-E2-05

Versión 0.1

25/02/2022

WIS testing knowledge report, maresprod5

Tabla de contenidos

Tabla de contenidos 2

Historial de versiones 2

Resumen ejecutivo 3

Introducción 3

Contenido 3

Conclusiones 4

Bibliografía 4

Historial de versiones

Fecha	Versión	Descripción de los cambios
27/02/2022	v1	Creación y finalización del documento

Resumen ejecutivo

Dada la brevedad del contenido presentado, estimo poco acertada la realización de un resumen ejecutivo de este documento.

Introducción

En este reporte comentaré mis conocimientos actuales sobre el testeo de los WIS de forma breve y concisa. Todo esto se realizará bajo la asunción de que los conocimientos que he adquirido en asignaturas anteriores sobre arquitecturas de servicios web hagan referencia a estos WIS, acrónimo que he escuchado por primera vez en esta asignatura.

El informe se estructura de la siguiente forma:

- Conocimientos básicos y experiencias previas con frameworks.
- Conocimientos sobre uso de JUnit y realización de tests parametrizados.
- Conocimientos sobre uso de Mockito.

Contenido

En anteriores asignaturas he aprendido sobre los tipos de test que se pueden realizar a la hora de comprobar el funcionamiento de una aplicación. De menor a mayor grado de granularidad, estas pruebas son: unitarias, de integración, de extremo a extremo, de aceptación y exploratorias. A pesar de esto, el único tipo de prueba que he utilizado de forma activa anteriormente son las unitarias, mediante el framework de testeo JUnit y el framework de mocking Mockito.

Respecto a los tests unitarios, la estructura se puede diseccionar en los siguientes elementos:

- Arrange: Preparación de la situación a probar (creación/obtención de datos relevantes, etc.).
- Act: Ejecución del código que se desea probar.
- Assert: Comprobación de la correcta ejecución del código y de la obtención de los resultados esperados.

Respecto a JUnit, he de destacar los conocimientos que he adquirido respecto a las etiquetas más esenciales que ofrece, entre las que destaco:

- @Test: Indica que la función se trata de una prueba.
- @Transactional: Indica que el test realiza consultas o modificaciones al sgbd.
- @Disabled: Indica que el test debe ser ignorado, y proporciona el motivo.
- @BeforeAll: Se ejecuta previo a todos los tests.
- @BeforeEach: Se ejecuta previo a cada test.
- @AfterAll: Tras todos los tests.
- @AfterEach: Tras cada test

Por supuesto, también he realizado anteriormente tests parametrizados, concretamente utilizando las siguientes etiquetas de JUnit:

- @ParameterizedTest: Indica que el test recibe datos parametrizados.
- @ValueSource: Los datos son de un solo tipo y se incluyen en la cabecera.
- @CsvSource: Los datos se proporcionan en un formato csv desde la cabecera.
- @MethodSource: Los datos son proporcionados por un método auxiliar.

En cuanto a Mockito, se trata de un framework que he usado brevemente durante la asignatura DP1, del cual puedo destacar las siguientes anotaciones útiles para el uso de mocks y stubs al realizar las pruebas:

- @MockBean: Indica que el la bean bajo la anotación debe ser simulada por Mockito a la hora de inyectar dependencias.
- @WithMockUser: Indica el usuario simulado que realizará la prueba (para pruebas que requieran autenticación, credenciales, etc.).
- @WebMvcTest: Indica que la capa web debe ser instanciada para una suite de pruebas.

Conclusiones

He realizado un breve informe sobre los conocimientos básicos de testeo de servicios web que he adquirido en asignaturas previas.
Espero poder ampliar durante el curso de esta asignatura los conocimientos aquí expuestos.

Bibliografía

En adición al temario de asignaturas anteriores, la única información que he llegado a aprender por mi cuenta (respecto al uso de @MethodSource), la encontré en el siguiente enlace:
https://www.arhohuttunen.com/junit-5-parameterized-tests/#how-to-convert-multiple-arguments-into-a-single-object