

Considerations

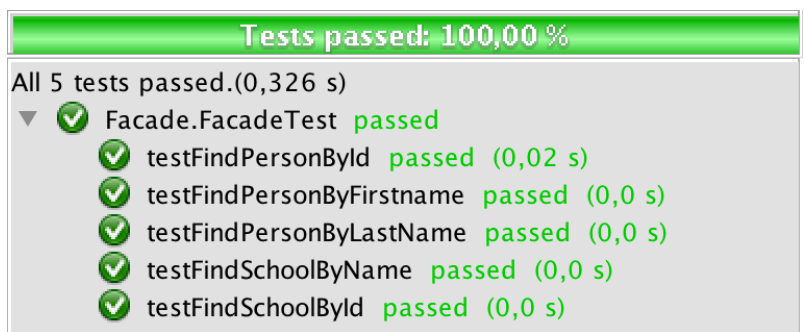
Mocking bruges primært i unit testing. Et objekt under test kan have afhængigheder af andre (komplekse) objekter. For at isolere opførelsen af objektet som man vil teste, kan man så erstatte disse objekter med mocks. Disse mocks simulere opførelsen af de rigtige objekter. Dette kan være meget nyttigt hvis man f.eks. arbejder med objekter fra database, ved at bruge mock kan vi så undgå at dykke ned i databasen, og gøre testing mindre kompleks.

Vi har lavet et JPA Entity Framework projekt med 2 tabeller som vi genere ud fra en eksisterende database. Derefter har vi lavet en facade som har adgang til database. Ved hjælp af Mockito har vi testet vores CRUD metoder fra facaden. Mockito opretter forbindelsen til databasen og vha. mock objekter kan vi teste vores CRUD metoder. Vi har bl.a. testet

"FindPersonById" og "FindSchoolByName".

Billedet til højre viser vores mock test resultater. Vi kunne også have gjort brug af en "test" database i stedet for mock. Men vi mente at det var nemmere at gøre brug

af Mockito, fremfor at lave en test database, fordi i en test database, skal man slette og tilføje data ved test cases. Mockito Frameworket har som sagt til formål at simulere opførelsen af en rigtig database.



The screenshot displays a test runner interface with a green header bar indicating 'Tests passed: 100,00 %'. Below this, it states 'All 5 tests passed.(0,326 s)'. A tree view shows the test class 'Facade.FacadeTest' with a green checkmark and the word 'passed'. Underneath, five individual test methods are listed, each with a green checkmark, the word 'passed', and its execution time in parentheses:

- testFindPersonById passed (0,02 s)
- testFindPersonByFirstname passed (0,0 s)
- testFindPersonByLastName passed (0,0 s)
- testFindSchoolByName passed (0,0 s)
- testFindSchoolById passed (0,0 s)