



Università degli Studi di Salerno
Corso di Ingegneria del Software



Enoteca
Il Gocciolatoio

Bevi poco... Ma bevi bene!

Unit-Test

Data: 19/01/2021

Coordinatore del Progetto:

Nome	Matricola
Francesco Di Palma	0512104586

Partecipanti:

Nome	Matricola
Giovanni Di Mauro	0512104596
Francesco Di Palma	0512104586
Maria Giuseppina Mosca	0512106090
Francesco Saviano	0512104912

Introduzione

Il Testing di unità consiste nel testare le singole unità del software del sistema suddivisi in sottosistemi individuali, con l'obiettivo di testare che ogni sottosistema è stato codificato correttamente. Il test verrà effettuato con il framework JUnit insieme alla libreria Mockito.

Relazione con altri documenti

Per individuare i test da effettuare si utilizzeranno la tecnica Black-Box e White-Box, quindi ci baseremo sui nostri documenti prodotti:

- Test-Plan
- Test-Case

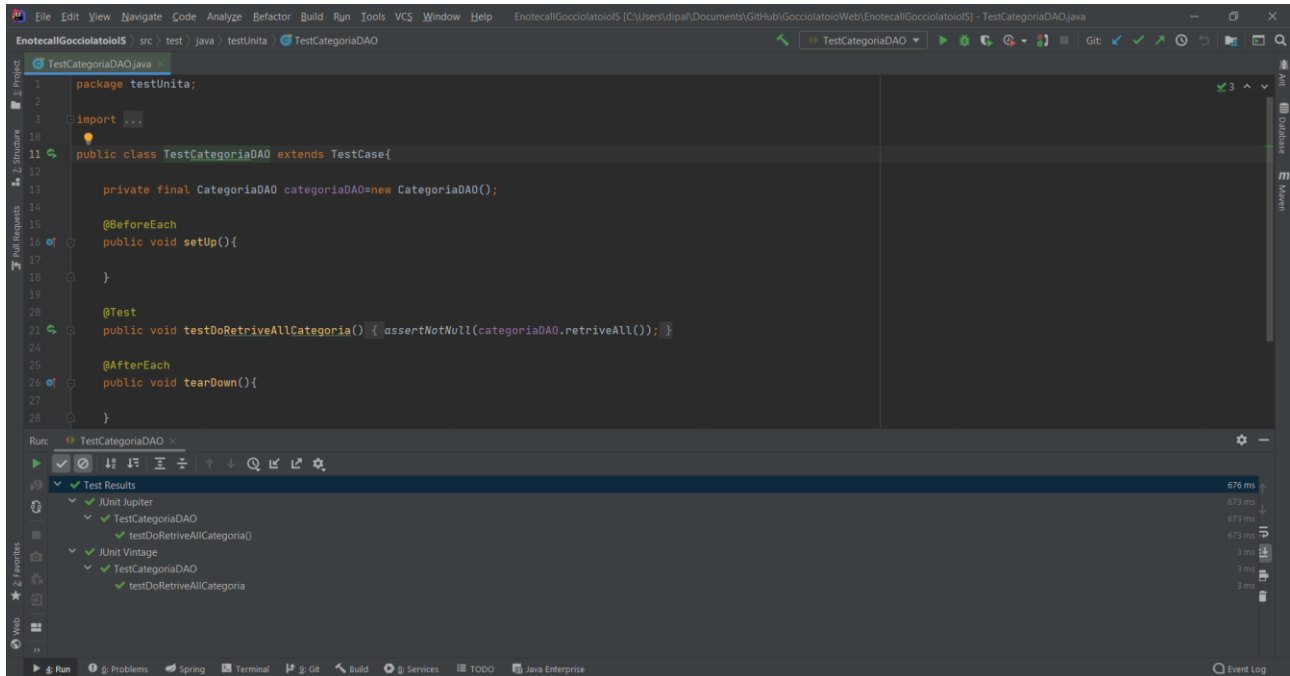
Approccio

Il Testing di Unità verrà diviso in TestingDAO e TestingGestore, e verranno testati singolarmente.

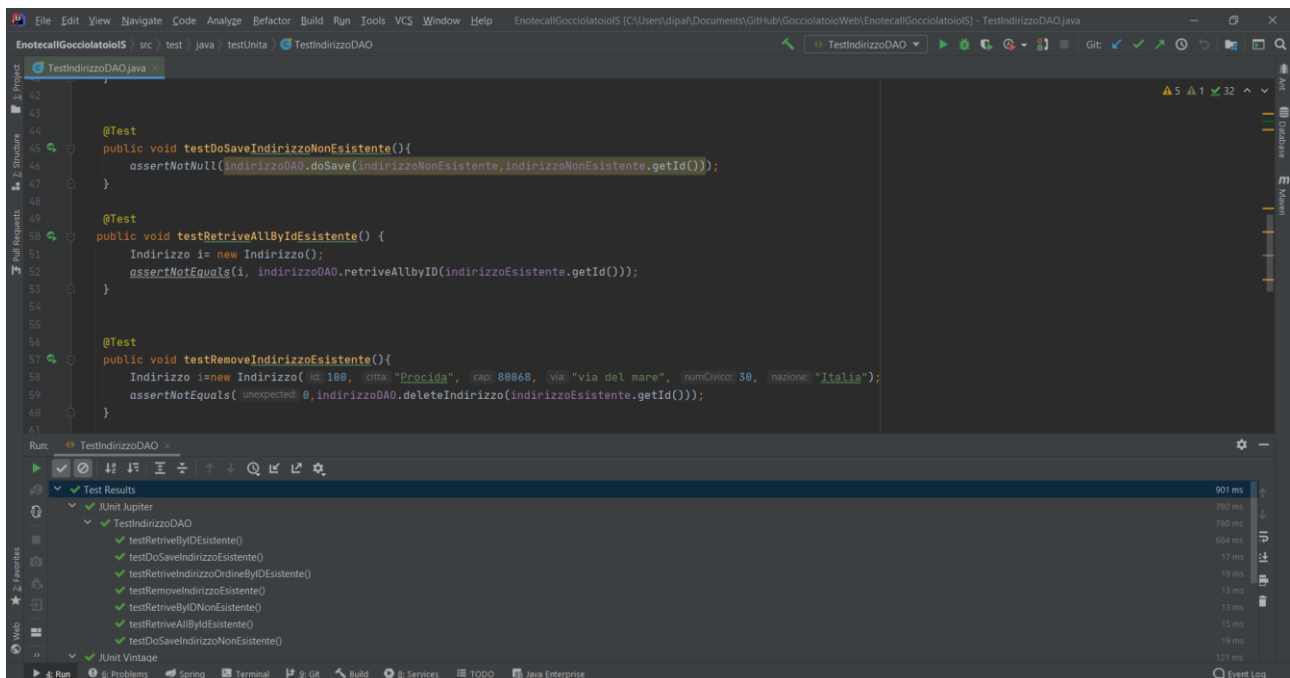
Testing di Unità

- TestingDAO

TestCategoria



TestIndirizzo



TestOrdine

The screenshot shows an IDE window titled "EnotecallGocciolatoioIS" with the file "TestOrdineDao.java" open. The code defines several test methods for the "TestOrdineDao" class:

```
@Test
public void testRetrieveAll() { assertNotNull(ordineDAO.retrieveAll()); }

@Test
public void testRetrieveByIdUser() { assertNotNull(ordineDAO.retrieveByIdUser(2)); }

@Test
public void testSalvataggioAddIndirizzoCancellazioneOrdine(){
    int id= ordineDAO.addAddressToOrder(indirizzoNonEsistente);
    utenteDAO.doSave(utenteNonEsistente);
    indirizzoNonEsistente.setId(id);
    ordineDAO.doSave(ordineNonEsistente,id);
    ordineDAO.deleteOrder(ordineNonEsistente.getId_ordine());
}

@AfterEach
public void tearDown(){
    System.out.println(ordineEsistente.getId_ordine());
}
```

Below the code editor, the "Run" tab displays the test results for "TestOrdineDao". The results show that all tests passed successfully:

- JUnit Jupiter: 896 ms
- TestOrdineDao: 793 ms
- testRetrieveByIdUser: 718 ms
- testSalvataggioAddIndirizzoCancellazioneOrdine: 42 ms
- testRetrieveAll: 23 ms
- JUnit Vintage: 103 ms
- TestOrdineDao: 103 ms
- testRetrieveByIdUser: 25 ms
- testSalvataggioAddIndirizzoCancellazioneOrdine: 23 ms
- testRetrieveAll: 36 ms

TestProdotto

The screenshot shows an IDE window titled "EnotecallGocciolatoioIS" with the file "TestProdottoDAO.java" open. The code defines several test methods for the "TestProdottoDAO" class:

```
@Test
public void testRetrieveOneEsistente(){
    assertNotNull(prodottoDAO.retrieveOne(prodottoEsistente.getId()));
}

@Test
public void testRetrieveOneNonEsistente() { assertNull(prodottoDAO.retrieveOne(prodottoNonEsistente.getId())); }

@Test
public void testRetrieveQuantMagazzinoEsistente(){
    assertEquals(prodottoEsistente.getQuantita_magazzino(),prodottoDAO.retrieveQuant(prodottoEsistente.getId()));
}

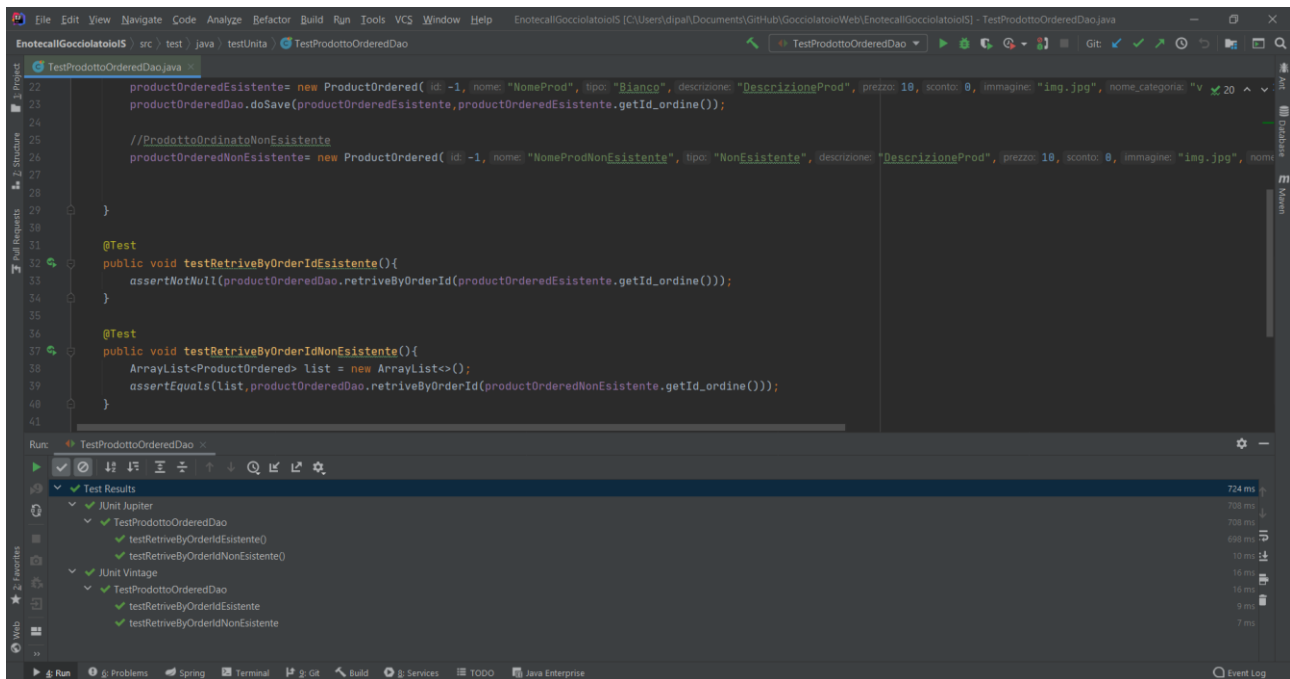
@Test
public void testRetrieveQuantMagazzinoNonEsistente(){
    assertEquals("expected: 0,prodottoDAO.retrieveQuant(prodottoNonEsistente.getId());",
    );
}
```

Below the code editor, the "Run" tab displays the test results for "TestProdottoDAO". The results show that all tests passed successfully:

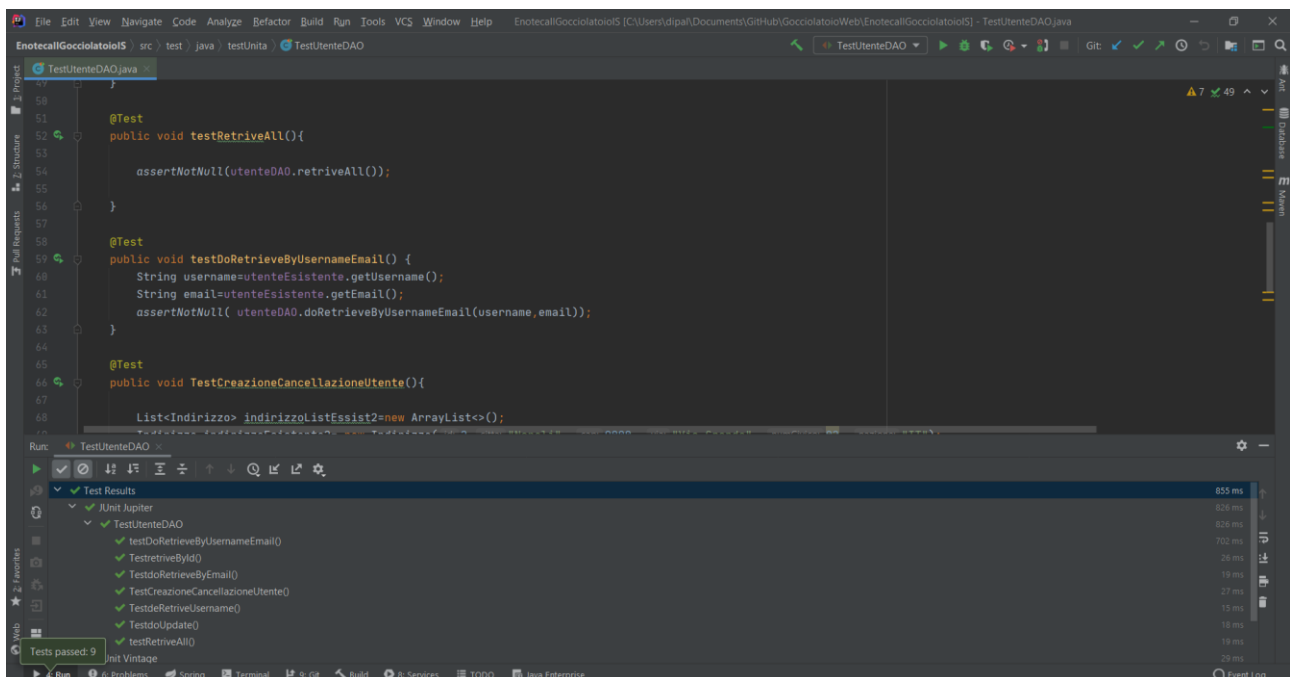
- JUnit Jupiter: 853 ms
- TestProdottoDAO: 758 ms
- testRetrieveCategoryNonEsistente: 657 ms
- testSalvataggioRimozioneProdotto: 17 ms
- testDoUpdateQuantityEsistente: 11 ms
- testRetrieveOneEsistente: 9 ms
- testRetrieveBySearchEsistente: 13 ms
- testRetrieveQuantMagazzinoNonEsistente: 9 ms
- testRetrieveCategoryEsistente: 9 ms
- testRetrieveQuantMagazzinoEsistente: 7 ms

At the bottom of the Run tab, it states "Tests passed: 22".

TestProdottoOrdinato



TestUtente



- TestingGestore

TestGestoreAccount

The screenshot shows the Eclipse IDE with the `TestGestoreAccount.java` file open. The code is a JUnit test for the `TC_GestioneModificaUtenteEmailVuoto()` method. It sets up a mock `UtenteDAO` and a `request` object, then calls `gestoreAccount.gestoreModificaUtente(request, response)`. The test expects an `IllegalArgumentException` with the message "email non corretto".

```
@Test
public void TC_GestioneModificaUtenteEmailVuoto() throws ServletException, IOException {
    UtenteDAO uDAO= new UtenteDAO();
    Utente u=uDAO.retrieveById(2);
    request.getSession().setAttribute("utente",u);

    request.addParameter( "username",u.getUsername());
    request.addParameter( "email", value: "");
    request.addParameter( "nome",u.getNome());
    request.addParameter( "cognome",u.getCognome());
    request.addParameter( "ruolo",u.getRuolo());

    gestoreAccount.gestoreModificaUtente(request,response);
    String exit = (String)request.getAttribute( "errorTest");
    String oracolo="email non corretto";
    assertEquals(oracolo,exit);
    System.out.println("oracolo: "+oracolo);
    System.out.println("servlet: "+exit);
}
```

The Run window shows the test results for `TestGestoreAccount`. All tests passed, with a total of 66 tests in 14.93 ms. The test results table is as follows:

Test Name	Duration
TestGestoreAccount	14.93 ms
TC_GestioneModificaUtenteUsernameVuoto()	788 ms
TC_GestioneModificaUtenteCittaVuoto()	12 ms
TC_GestioneModificaUtenteEmailVuoto()	3 ms
TC_GestioneModificaUtenteCognomeVuoto()	3 ms
TC_AggiungiUtenteNomeVuoto()	3 ms
TC_GestioneModificaUtenteByAdminNomeVuoto()	3 ms
TC_GestioneModificaUtenteByAdminNomeVuoto()	3 ms
TC_GestioneModificaUtenteByAdminCognomeVuoto()	4 ms
TC_GestioneModificaUtenteByAdminIdVuoto()	3 ms

Warnings in the console:

- WARNING: An illegal reflective access operation has occurred
- WARNING: Illegal reflective access by org.mockito.cglib.core.ReflectUtils\$2 (file:/C:/Users/dipal/.m2/repository/org/mockito/mockito-core/2.28.2/mockito-core-2.28.2.jar) to method java.lang.ProcessImpl::start0()
- WARNING: Please consider reporting this to the maintainers of org.mockito.cglib.core.ReflectUtils\$2
- WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
- WARNING: All illegal access operations will be denied in a future release

TestGestoreCarrello

The screenshot shows the Eclipse IDE with the `TestGestoreCarrello.java` file open. The code is a JUnit test for the `TC_AggiungiProdottoCarrello1()` method. It sets up a mock `Carrello` and a `request` object, then calls `gestoreCarrello.gestoreAggiungiProdottoCarrello(request, response)`. The test expects the message "prodotto aggiunto al carrello".

```
@Test
void TC_AggiungiProdottoCarrello1() throws ServletException, IOException {
    //Aggiunta prodotto al carrello singola quantita
    carrello.put(prodotto, quantita);
    request.getSession().setAttribute("carrello",carrello);
    request.addParameter( "prodId", String.valueOf(prodotto.getId()));
    request.addParameter( "addNum", String.valueOf("1"));

    String message = "prodotto aggiunto al carrello";
    System.out.println("oracolo: "+message);
    servlet.gestoreAggiungiProdottoCarrello(request,response);
    String result = (String) request.getAttribute( "errorTest");
    assertEquals(message, result);

    request.getSession().removeAttribute("carrello");
}

@Test
void TC_AggiungiProdottoCarrello2() throws ServletException, IOException {
    //Aggiunta prodotto al carrello singola quantita
}
```

The Run window shows the test results for `TestGestoreCarrello`. All tests passed, with a total of 6 tests in 824 ms. The test results table is as follows:

Test Name	Duration
TestGestoreCarrello	824 ms
TC_GestoreAcquistaUtenteNonLoggato()	750 ms
TC_AggiungiProdottoCarrello1()	28 ms
TC_AggiungiProdottoCarrello2()	2 ms
TC_RimuoviProdottoCarrello()	1 ms
TC_GestoreAcquistaUtentePrivilediDiritto()	3 ms
TC_GestoreAcquistaUtenteLoggato()	39 ms

Output in the console:

```
oracolo: utente non loggato, ritorno alla pagina login
Servlet: utente non loggato, ritorno alla pagina login
Per effettuare un acquisto Registrati oppure effettua il Login:

oracolo: prodotto aggiunto al carrello
Servlet: prodotto aggiunto al carrello
```

TestGestoreOrdine

The screenshot shows the Eclipse IDE with the `TestGestoreOrdine.java` file open. The code defines two test methods: `TC_GestoreOrdine()` and `TC_EliminaOrdine()`. The `TC_GestoreOrdine()` method performs a series of assertions and prints the results of the test.

```
@Test
public void TC_GestoreOrdine() throws ServletException, IOException {
    UtenteDAO uDAO=new UtenteDAO();
    Utente utente=uDAO.retrieveById(2);
    request.getSession().setAttribute("utente", utente);
    gestoreOrdine.gestoreOrdini(request, response);
    String exit=(String)request.getAttribute(name: "errorTest");
    String oracolo="TuttiGliOrdiniUtente";

    assertEquals(oracolo, exit);

    System.out.println("oracolo: "+oracolo);
    System.out.println("servlet: "+exit);
}

@Test
public void TC_EliminaOrdine() throws ServletException, IOException {
    // ...
}
```

The Run window shows the test results for `TestGestoreOrdine`. The tests passed, and the output is as follows:

Test Results	Duration	Output
TC_GestoreOrdine	774 ms	oracolo: ordine aggiunto servlet: ordine aggiunto
TC_AggiungiOrdine	60 ms	ho preso tutti i dati
TC_RestituisceOrdineClienti	677 ms	oracolo: OrdiniIDITuttiIClienti servlet: OrdiniIDITuttiIClienti
TC_EliminaOrdine	13 ms	
TC_GestoreOrdine	10 ms	
TC_RestituisceDettagliOrdine	8 ms	
TC_RestituisceOrdiniUtente	8 ms	

TestGestoreProdotto

The screenshot shows the Eclipse IDE with the `TestGestoreProdotto.java` file open. The code defines a test method `TC_AggiuntaProdotto()` that performs a series of assertions and prints the results of the test.

```
MockitoAnnotations.initMocks(this);

@Test
void TC_AggiuntaProdotto() throws ServletException, IOException {
    request.addParameter(name: "nome", prodotto.getNome());
    request.addParameter(name: "tipo", prodotto.getTipo());
    request.addParameter(name: "descrizione", prodotto.getDescrizione());
    request.addParameter(name: "prezzo", String.valueOf(prodotto.getPrezzo()));
    request.addParameter(name: "sconto", String.valueOf(prodotto.getSconto()));
    request.addParameter(name: "immagine", prodotto.getImmagine());
    request.addParameter(name: "anno", String.valueOf(prodotto.getAnno()));
    request.addParameter(name: "regione", prodotto.getRegione());
    request.addParameter(name: "gradazione", String.valueOf(prodotto.getGradazione()));
    request.addParameter(name: "formato", String.valueOf(prodotto.getFormato()));
    request.addParameter(name: "quantita_magazzino", String.valueOf(prodotto.getQuantita_magazzino()));
    request.addParameter(name: "nome_categoria", prodotto.getNome_categoria());

    String message = "Prodotto inserito con successo";
    System.out.println("oracolo: "+message);
}
```

The Run window shows the test results for `TestGestoreProdotto`. The tests passed, and the output is as follows:

Test Results	Duration	Output
TC_AggiuntaProdotto	861 ms	
TC_ModificaProdottoNomeVuoto	861 ms	
TC_AggiuntaProdottoGradazioneVuoto	125 ms	
TC_AggiuntaProdottoScontoErrato	2 ms	
TC_ModificaProdottoByAdminID0	2 ms	
TC_ModificaProdottoFormatoErrato	627 ms	
TC_AggiuntaProdottoAnnoVuoto	3 ms	
TC_MostraProdottoVuoto	3 ms	
TC_ModificaProdottoTipoErrato	1 ms	
TC_AggiuntaProdottoQuantitaVuoto	2 ms	