GABRIELE ROMANATO

Introduzione al Test-Driven Development (TDD) in Java

Il Test-Driven Development (TDD) è una metodologia di sviluppo del software che prevede la scrittura dei test prima della scrittura del codice di produzione. Questo approccio è particolarmente efficace in Java grazie al supporto di framework come JUnit.

Cos'è il TDD

TDD segue un ciclo molto semplice, spesso chiamato **Red-Green-Refactor**:

- **Red:** Scrivi un test che fallisce.
- **Green:** Scrivi il codice minimo per far passare il test.
- **Refactor:** Migliora il codice mantenendo i test verdi.

Setup per TDD in Java

Per iniziare con TDD in Java, puoi utilizzare JUnit come framework di testing. Di seguito un esempio di configurazione base con Maven:

```
<dependencies>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.10.0</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Esempio pratico

Supponiamo di voler implementare una semplice classe `Calcolatrice` che somma due numeri. Iniziamo scrivendo il test:

```
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class CalcolatriceTest {

    @Test
    public void testSomma() {
        Calcolatrice calc = new Calcolatrice();
        int risultato = calc.somma(2, 3);
        assertEquals(5, risultato);
    }
}
```

Ora scriviamo il codice minimo per far passare il test:

```
public class Calcolatrice {

    public int somma(int a, int b) {
        return a + b;
    }
}
```

Vantaggi del TDD

- Codice meglio testato e più affidabile.
- Miglior design del software grazie al refactoring continuo.
- Maggiore fiducia nei cambiamenti del codice.

Conclusione

Il TDD è una pratica potente per sviluppatori Java che desiderano produrre codice di qualità, facilmente manutenibile e ben testato. Anche se richiede disciplina, i benefici a lungo termine superano ampiamente il costo iniziale.