



JUnit

Programmazione Java - Modulo 9

Livio Pompianu



- Test-driven development (TDD)
- JUnit

Test-Driven Development

Processo di sviluppo del software basato sull'utilizzo di test case scritti prima di implementare il codice dell'applicazione

Test-Driven Development

1. Per ogni nuova specifica si sviluppa un *test case*: porzione di codice che testa la singola funzionalità desiderata
2. Esecuzione di tutti i test e verifica di quali falliscono
3. Scrittura del codice per il superamento del test
4. Esecuzione dei test
5. *Refactor* del codice
6. Ripeti

TDD – Primo approccio

1) Scrittura del test per la specifica: la *classe Persona* ha un metodo *getNome* che rende il nome della persona

```
public class TestAnagrafica{  
    public static void main(String[] args) {  
        System.out.println("Test 1: " + test1());  
    }  
  
    public static boolean test1() {  
        Persona p = new Persona("Nome");  
        if(p.getNome().equals("Nome"))  
            return true;  
        return false;  
    }  
}
```

2) L'esecuzione del test fallisce, la classe *Persona* non esiste

3) Scrittura del codice

```
class Persona{  
    String name;  
  
    Persona(String n) {  
  
        name = n;  
    }  
  
    String getNome() {  
        return nome;  
    }  
}
```

TDD – Primo approccio

4) L'esecuzione del test ha successo

```
public class TestAnagrafica{  
  
    public static void main(String[] args) {  
        System.out.println("Test 1: " + test1());  
    }  
  
    public static boolean test1() {  
        Persona p = new Persona("Nome");  
        if(p.getNome().equals("Nome"))  
            return true;  
        return false;  
    }  
}
```

TDD – Primo approccio

5) Refactor: modificatori di accesso, nomi variabili, documentazione...

```
/**
 * Classe per la gestione di una Persona
 * @author Livio
 */
public class Persona{
    private String nome;
    /**
     * Costruttore per creare una nuova persona con Nome
     * @param nome Nome della persona
     */
    public Persona(String nome) {
        this.nome = nome;
    }

    public String getNome(){
        return nome;
    }
}
```


TDD – Principi cardine

I test superati non devono essere cancellati dalle versioni successive del software di test

Ogni test deve essere il più leggero possibile e mirato a testare una singola funzionalità (*unit testing*)

I test non devono testare dettagli implementativi

I test devono essere indipendenti tra loro (evitare falsi negativi)

L'ordine di esecuzione dei test dovrebbe essere casuale



JUnit

Unit test framework per il linguaggio Java

Mette a disposizione metodi per svolgere diversi tipi di test:



- Uguaglianza, ordine, range di valori e oggetti
- Gestione delle eccezioni
- Tempo di esecuzione impiegato da un metodo
- ...

Dei tag consentono di controllare il flusso di esecuzione dei test:


- Disabilitare un test
- Abilitare / Disabilitare un test in base al sistema operativo
- Eseguire un test prima / dopo un altro
- ...










Se un test fallisce verrà visualizzata una barra rossa (e l'indicazione del test fallito), se è tutto ok viene invece stampata una barra verde

Finished after 0,141 seconds



Runs: 9/9  Errors: 0  Failures: 1




▼  JUnitEsemplioAssertion [Runner: JUnit 5] (0,059 s)

-  exceptionTesting() (0,001 s)
-  groupedAssertions() (0,004 s)
-  timeoutExceededWithPreemptiveTermination() (0,010 s)
-  timeoutExceeded() (0,002 s)
-  dependentAssertions() (0,003 s)
-  timeoutNotExceeded() (0,002 s)
-  timeoutNotExceededWithMethod() (0,002 s)
-  standardAssertions() (0,027 s)
-  timeoutNotExceededWithResult() (0,006 s)

Finished after 0,2 seconds

Runs: 9/9  Errors: 0  Failures: 0



>  JUnitEsemplioAssertion [Runner: JUnit 5] (0,014 s)

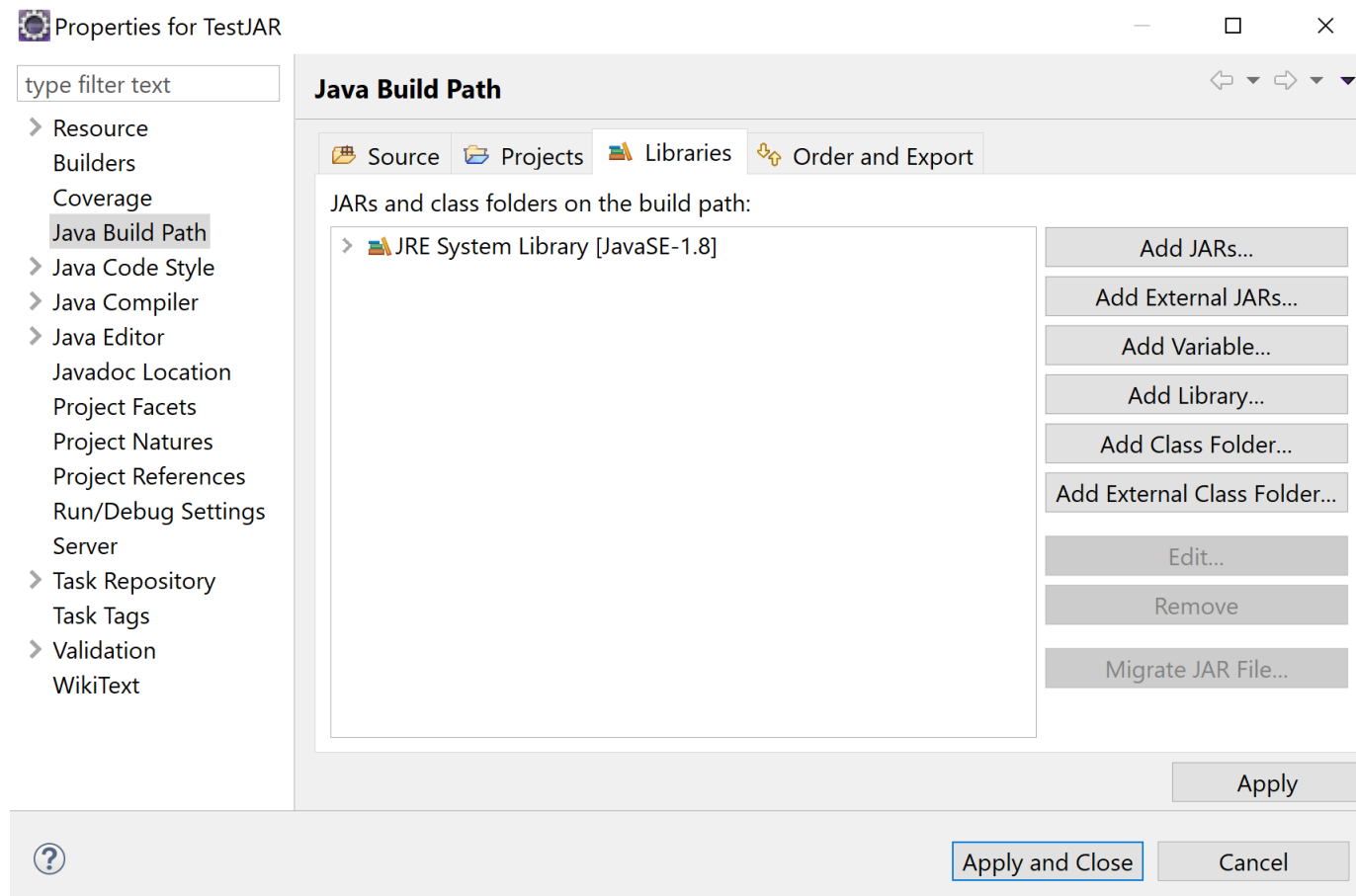
Installazione

Aprire il progetto Anagrafica

Click destro sul progetto e
click su *Properties*

Selezionare *Java Build Path*

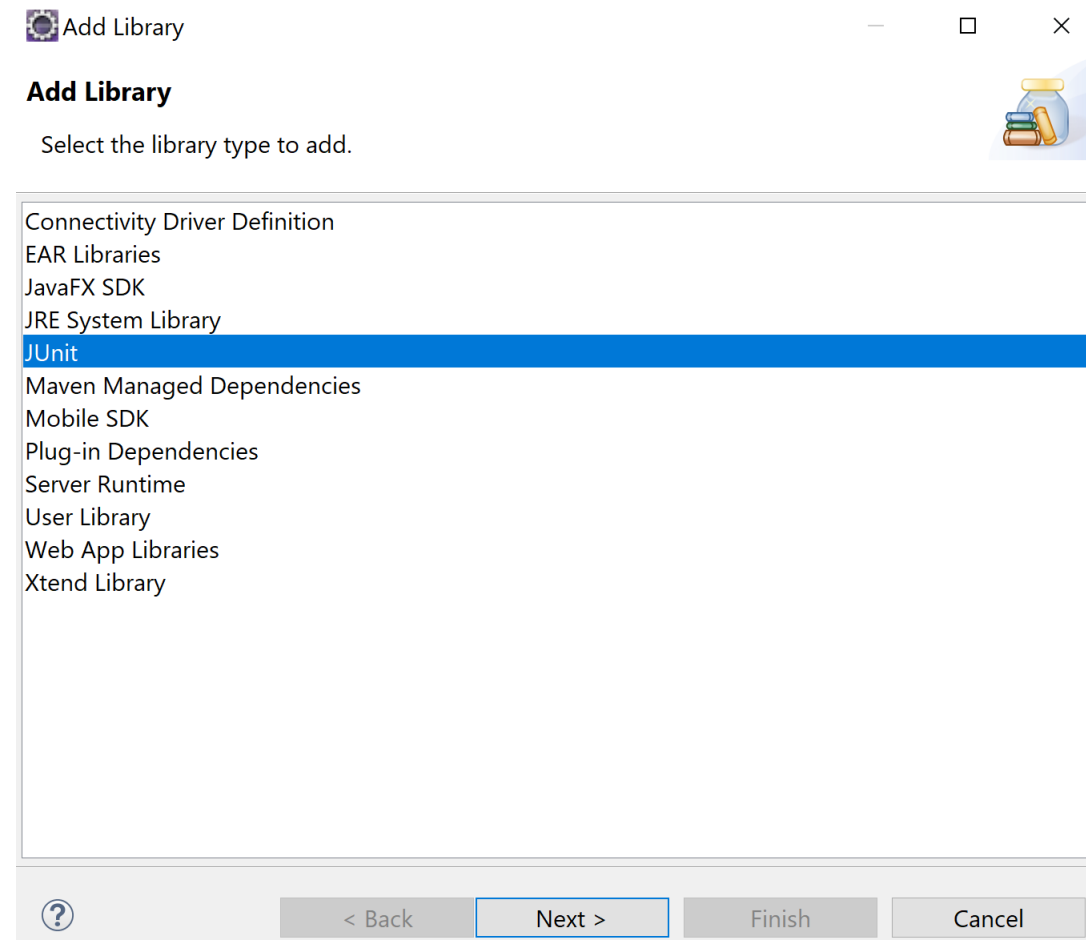
Click su *Add Library...*



Installazione

Selezionare *JUnit*

Premere *Next*

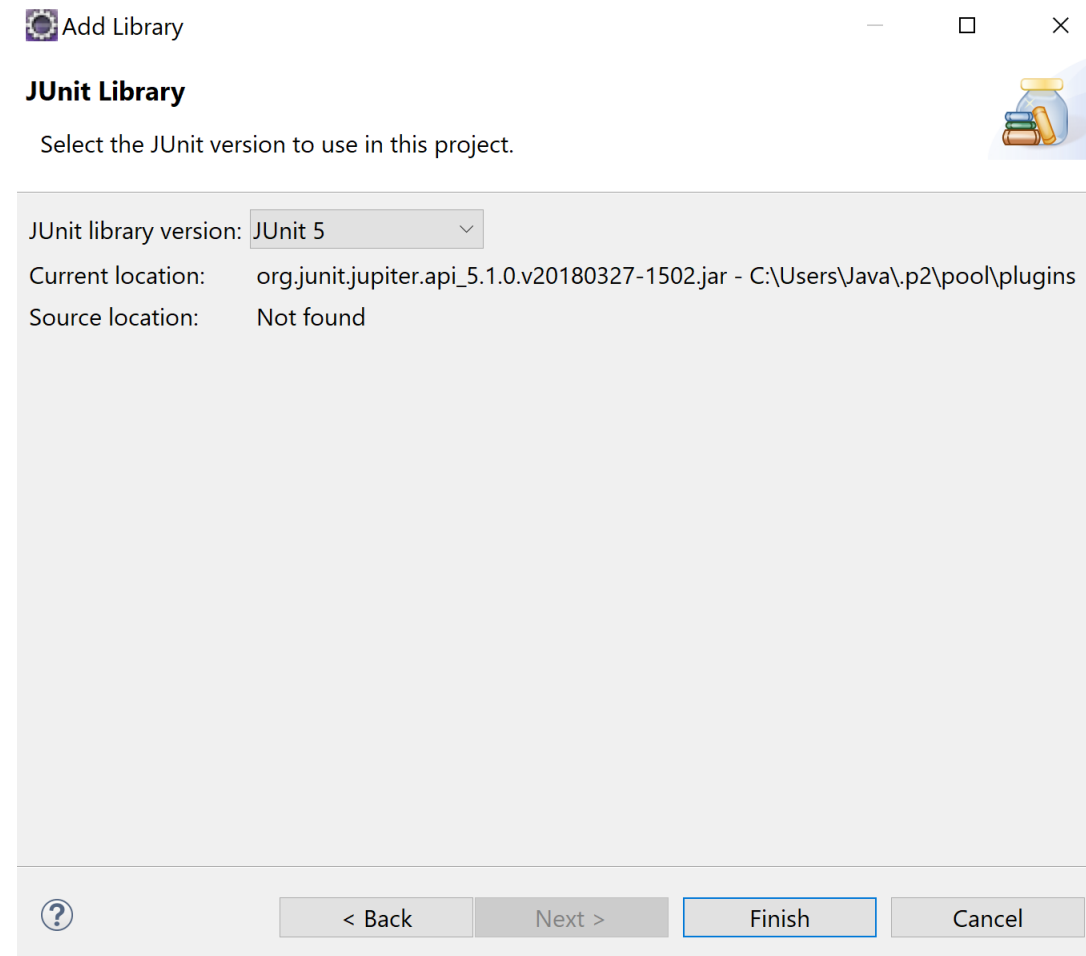


Installazione

Scegliere la versione 5

Premere Finish

JUnit5 è stato aggiunto alle
external libraries



Installazione

Tutti i jar di JUnit5 sono stati aggiunti come libreria

```
> 📁 > src
> 📁 resources
> 📖 Referenced Libraries
> 📖 JRE System Library [JavaSE-1.8]
▼ 📖 JUnit 5
  > 📄 org.junit.jupiter.api_5.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.jupiter.engine_5.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.jupiter.migrationsupport_5.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.jupiter.params_5.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.platform.commons_1.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.platform.engine_1.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.platform.launcher_1.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.platform.runner_1.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.platform.suite.api_1.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.junit.vintage.engine_5.1.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.opentest4j_1.0.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 org.apiguardian_1.0.0.v20180327-1502.jar - C:\Users\Java\.p2\pool\plugins
  > 📄 junit.jar - C:\Users\Java\.p2\pool\plugins\org.junit_4.12.0.v201504281640
  > 📄 org.hamcrest.core_1.3.0.v20180420-1519.jar - C:\Users\Java\.p2\pool\plugins
```


Avviare i seguenti file di esempio presenti nel repository:

- JUnitTestEsempioTag
- JUnitTestEsempioAssertion

Riepilogo

- Nel TDD il test è scritto prima del codice
- I test devono essere verificare singole funzionalità (unit testing)
- I test devono essere indipendenti tra loro
- JUnit è un framework per realizzare unit testing in Java