Panoramica dei test JUnit del progetto "Jam Factory"

1. Introduzione

Questo documento descrive il funzionamento della suite di unit test scritta in JUnit 5 che verifica la correttezza delle classi **DAO** e dei principali **Model** dell'applicazione web per la gestione della filiera di produzione di marmellate e confetture.

Ogni test segue alcune linee guida comuni:

- Importa *org.junit.jupiter.api* ed usa le asserzioni statiche di **JUnit 5** (`assertEquals`, `assertTrue`, `assertFalse`, `assertNull`, `assertNotNull`).
- Inizializza le dipendenze nella fase `@BeforeEach`, così da avere un contesto pulito per ogni metodo di test.
- Usa dati fittizi o generati al volo (es. timestamp nell'email) per garantire l'indipendenza dei test.
- Verifica sia gli scenari **positivi** (input valido) sia quelli **negativi** (input errato o id inesistente).

---

2. Struttura tipica di un test

```
class XxxDaoTest {
    private XxxDao dao;

    @BeforeEach
    void setUp() {
        dao = new XxxDao();
    }

    @Test
    void metodoDaTestare() {
        // Arrange / Act
        // Assert
    }
}
```

---

3. Descrizione dei singoli file di test

3.1 CustomerTest

Verifica il costruttore del modello **Customer**:

- **testCustomerCreation_WithPassword** e **testCustomerCreation_WithoutPassword** controllano che ragione sociale, P IVA, email e password (quando prevista) siano impostati correttamente.
- **testSetId** si assicura che il metodo `setId` aggiorni l'identificativo del cliente.

3.2 EmployeeDaoTest

Copre il DAO degli impiegati:

- **testEmailExists** accerta che una email assente non venga segnalata come presente.
- **testInsertEmployee** crea un impiegato con email unica e verifica che l'id restituito sia > 0 e che l'email risulti poi esistente.
- **testInsertEmployeeWithExistingEmail** controlla il vincolo di unicità dell'email: il secondo inserimento con la stessa email deve fallire (id = 0).
- **testFindAll** e **testInsertAndFindAll** verificano il recupero dell'elenco impiegati e l'aumento di cardinalità dopo un nuovo inserimento.
- **testInsertWithNullValues** prova la gestione di campi obbligatori nulli.

3.3 FeedbackDaoTest

Testa le funzionalità di feedback dei dipendenti sui prodotti:

- Recupero di tutti i feedback, di quelli di un dipendente o di quelli disponibili per un dipendente.
- Verifica che l'API rifiuti id negativi o rating fuori range (p.es. 10 stelle).

3.4 OrderDaoTest

Si concentra sugli ordini dei clienti e sulle statistiche di vendita:

- **testGetRecentOrders**, **testGetTodayOrdersCount**, **testGetTodaySales**, **testGetTotalProductsCount**, **testGetTotalCustomersCount** validano query aggregate.
- **testCreateOrderInvalidCustomer** garantisce che non si possa creare un ordine con id cliente non valido.
- **testGetCustomerOrdersInvalidCustomer** e **testGetOrderItemsInvalidOrder** verificano che il DAO restituisca collezioni vuote per id inesistenti.

3.5 ProductDaoTest

Copre le operazioni CRUD e di magazzino sui prodotti:

- Ricerca di tutti i prodotti, di quelli destinati allo shop e di quelli non inclusi nel Taste List.
- Inserimento di un prodotto di test con eventuale pulizia (delete) a fine test.
- Recupero per id valido/invalidato, calcolo dello stock corrente e test di **addStock** con successivo rollback.

3.6 RawMaterialOrderDaoTest

Gestisce gli ordini di materia prima ai fornitori:

- Verifica che `findAll` restituisca informazioni aggregate (nome fornitore, materia prima) ordinate per data decrescente.
- Inserimento, ricerca, cancellazione di un ordine; controlli su id fornitore inesistente, quantità negative e ordinamento per data.

3.7 SupplierDaoTest

Controlla il DAO dei fornitori:

- Recupero di tutti i fornitori e ricerca per id.
- Inserimento di un nuovo fornitore con successivo delete di cleanup.
- Test di update e delete su id non valido.

3.8 TasteListDaoTest

Copre la logica del programma **Taste Hunter**:

- Individua i prodotti presenti o assenti nella lista di degustazione.
- Aggiunge, rimuove prodotti e verifica la media delle valutazioni, compresi i casi di id non valido.

3.9 StatsDaoTest

Fornisce test sulle statistiche di alto livello:

- Recupera counts globali (prodotti, clienti, dipendenti, fornitori) e verifica che siano ≥ 0.
- Top/Least selling products, top suppliers, vendite mensili, top customers, valore medio ordine e distribuzione rating 1-5.

3.10 UserDaoTest

Verifica l'autenticazione degli utenti:

- Controlla che credenziali valide restituiscano l'istanza **User** con il relativo ruolo (ADMIN, CLIENT, EMPLOYEE).
- Garantisce che credenziali invalide o parametri null restituiscano *null*.

---

4. Conclusioni

La suite di test si focalizza su:

1. **Integrità dei dati** (vincoli di unicità, valori non negativi).
2. **Robustezza** (gestione di input non valido o nullo).
3. **Funzionalità business-critical** (statistiche, stock, autenticazione, feedback).

Eseguendo questi test con *mvn test* o direttamente da Eclipse, il progetto fornisce una rete di regressione che facilita lo sviluppo in sicurezza di nuove funzionalità e refactoring futuro.