



Spring Data

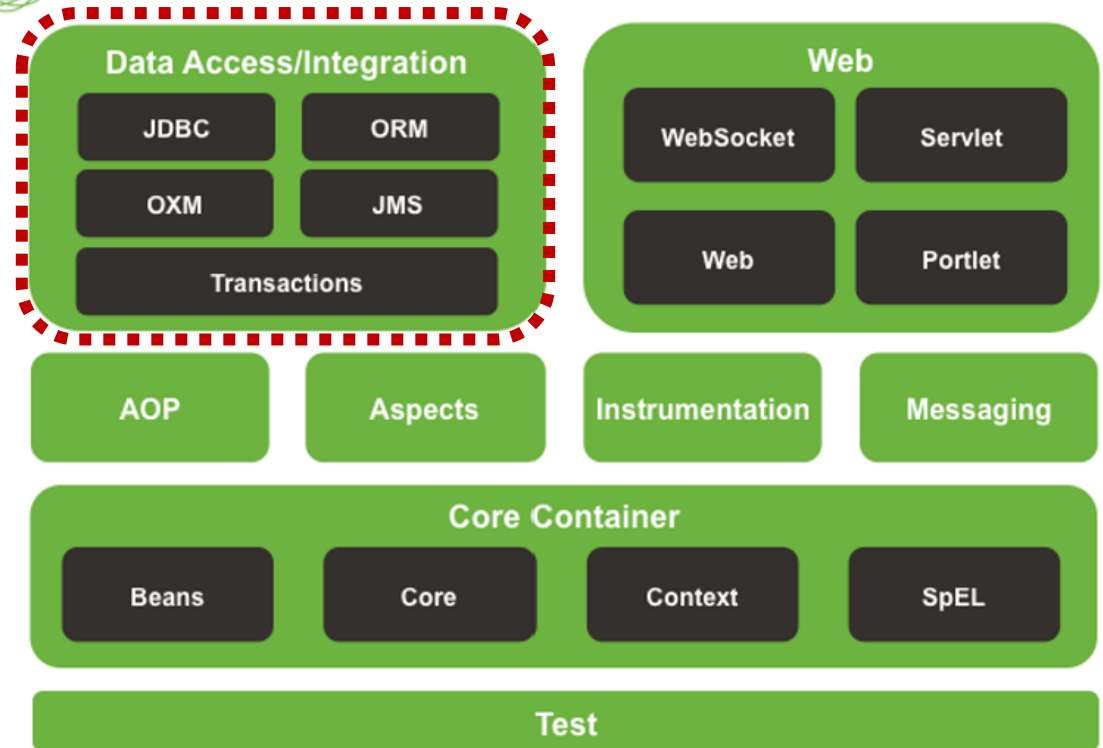
Indice argomenti

- Introduzione a Spring Data
- Configurazione di Spring Data
- Spring Data Repository

Introduzione a Spring Data



Spring Framework Runtime



fonte: <https://docs.spring.io>

Sezione: Spring Data Access

Cosa vedremo:

- ❑ Introduzione a Spring Data.
- ❑ Componenti.
- ❑ Spring Data e JPA.

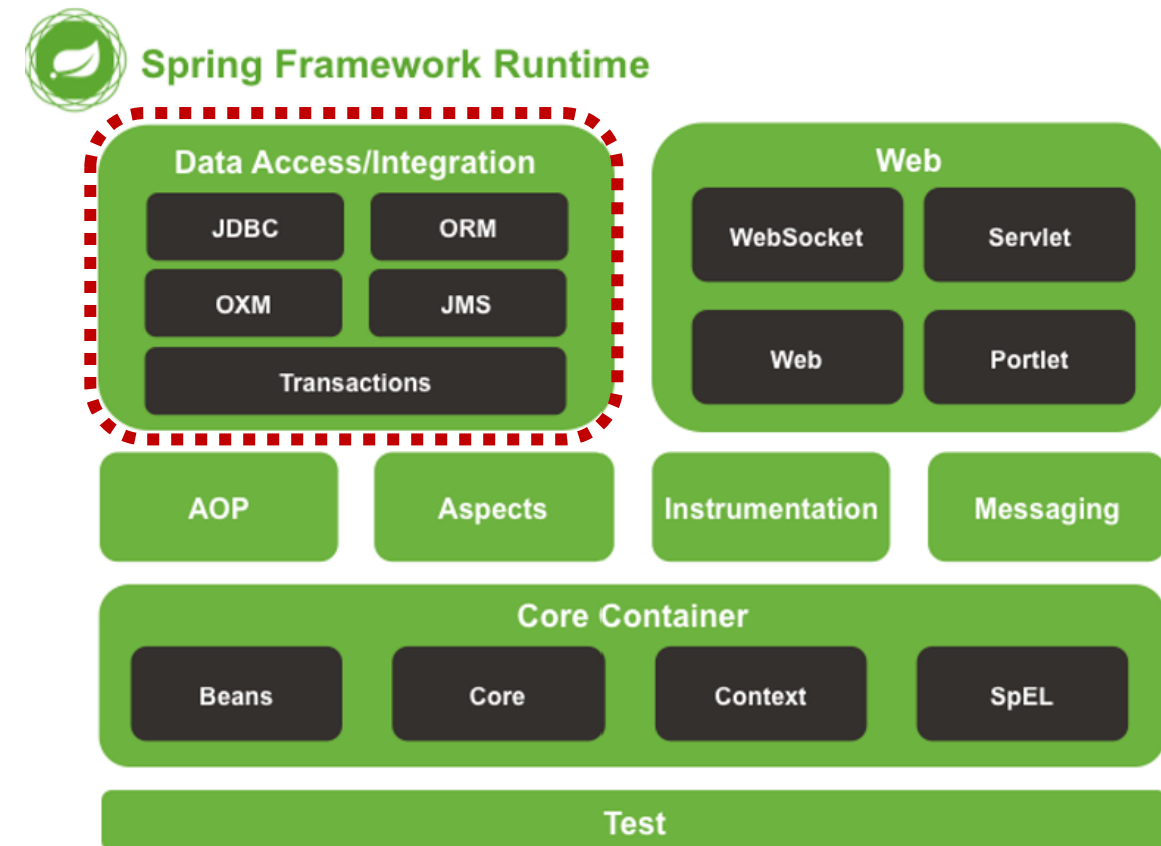
Spring Data

Introduzione

Spring Data Access è la parte di Spring che fornisce gli strumenti necessari all'applicazione per accedere ai dati.

Spring Data Access è composto da diversi moduli che sono specializzati per:

- Gestire l'accesso a uno specifico database
 - Oracle
 - MySql
 - PostgreSQL
 - ...
- Gestire l'accesso ai dati con una specifica tecnologia:
 - JDBC
 - ORM
 - ...



fonte: <https://docs.spring.io>

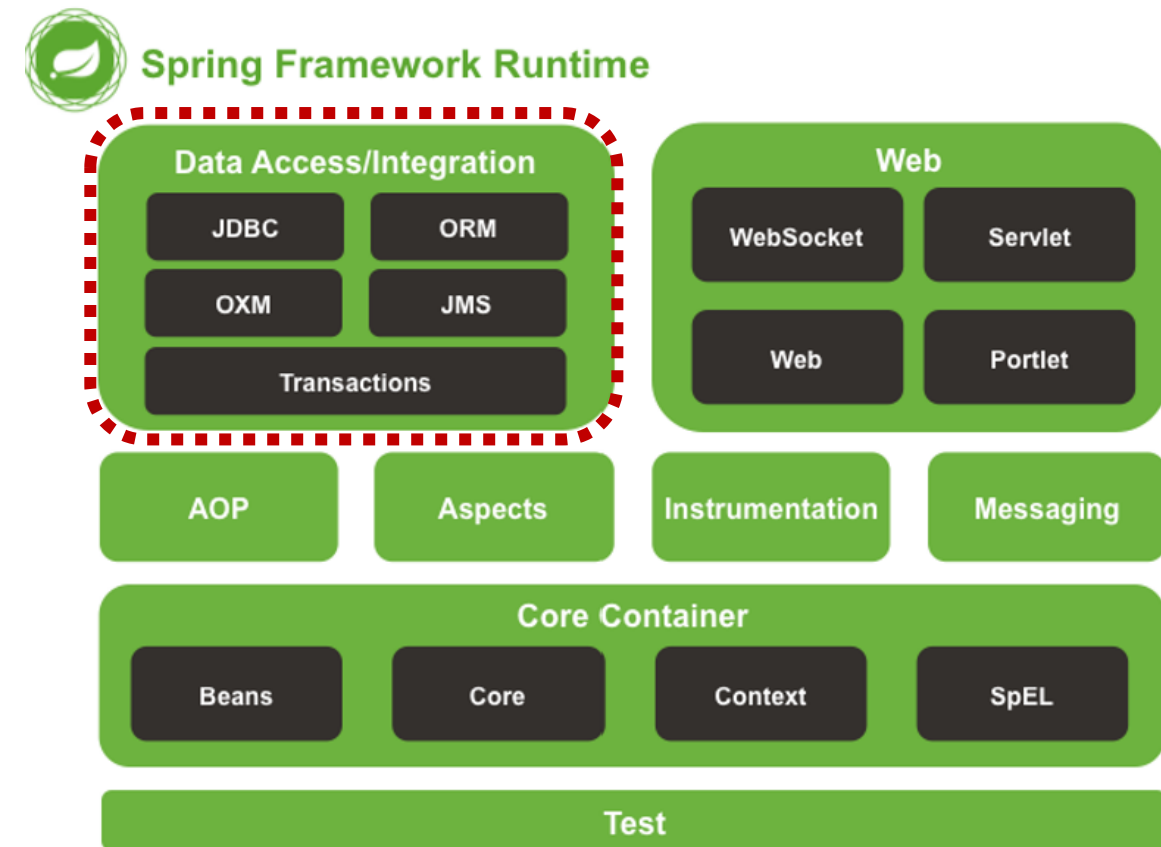
Spring Data

Introduzione

Spring Data Access è composto da moduli specifici che vengono utilizzati dall'applicazione in base alle proprie esigenze.

Principali moduli di Spring Data Access

- Spring Data Commons
 - Contiene i moduli base per utilizzare Spring Data e sono utilizzati da tutti gli altri moduli.
- Spring Data JDBC
 - Componenti per l'accesso al database tramite JDBC.
- Spring Data JPA
 - Componenti per l'accesso al database tramite JPA,



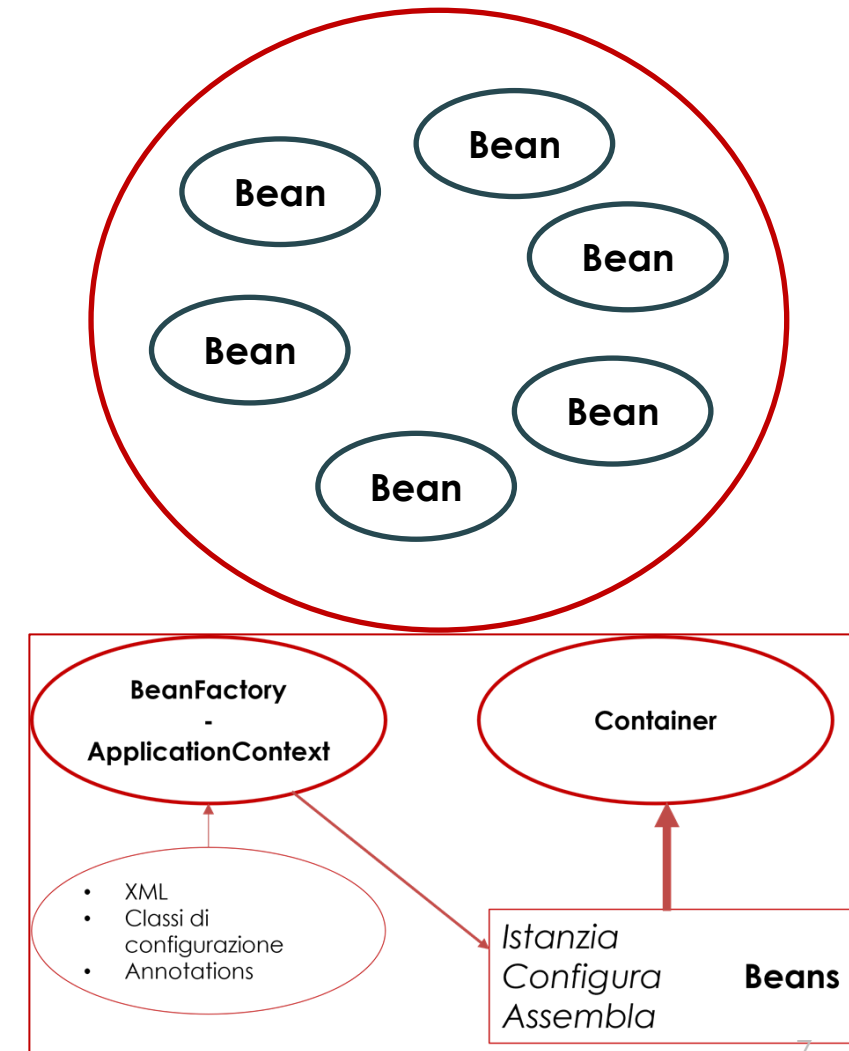
fonte: <https://docs.spring.io>

Ripasso. Spring Core

I Beans

Il container IoC di Spring e i Bean. Creazione tramite **Annotation**.

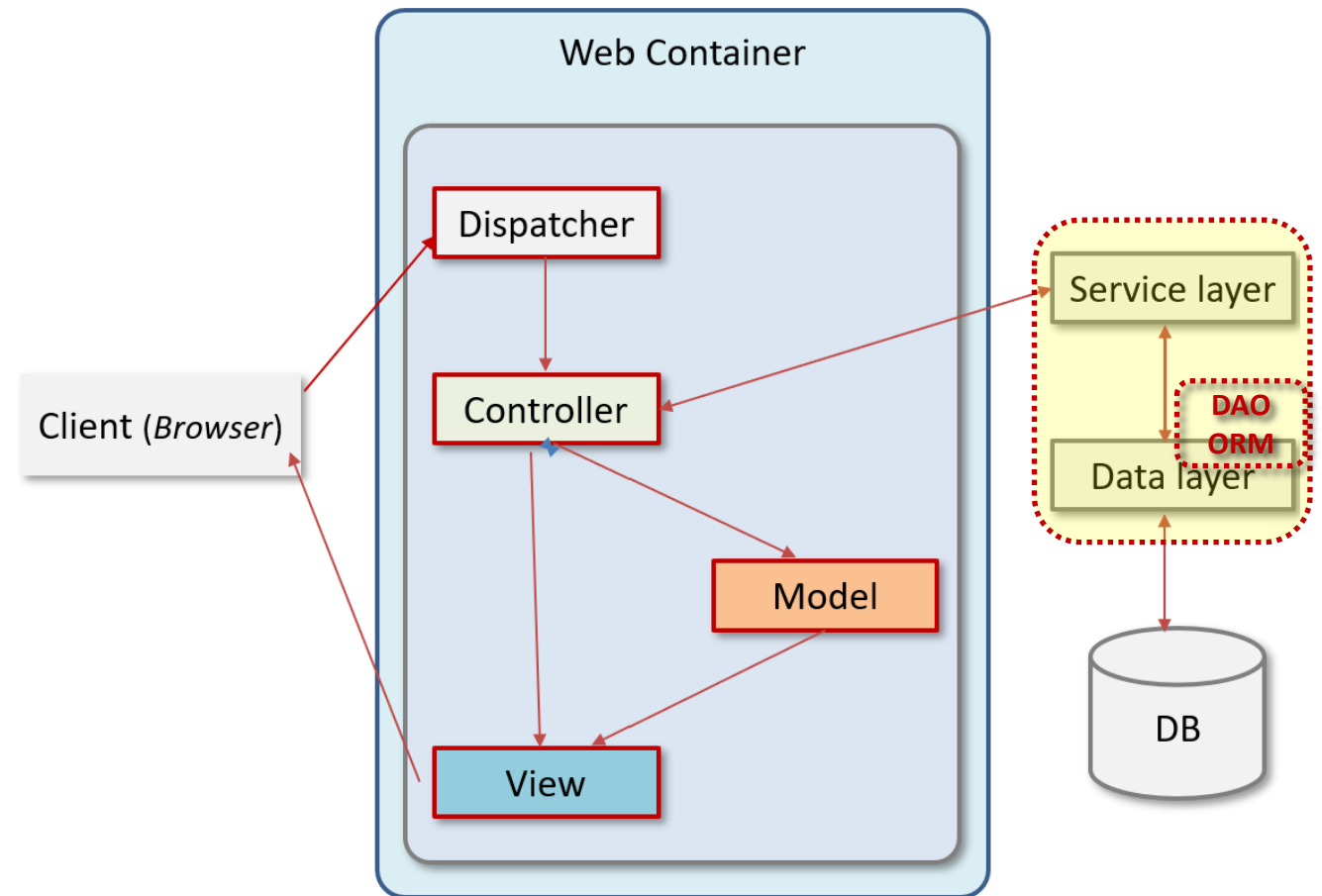
- La creazione di un bean tramite Annotation prevede:
 - Un file con annotation @Configuration
 - Contenente almeno un metodo annotato come @Bean.
 - Un pojo che rappresenta il bean.
 - L'invocazione del metodo getBean().



Spring Data JPA

Spring Data supporta le specifiche JPA per l'accesso al database.

- Spring supporta:
 - L'ORM Hibernate.
 - Le implementazioni DAO.
 - La gestione delle transazioni.
- Il supporto ai componenti JPA è fornito tramite
 - Le funzionalità **IoC** messe a disposizione dal framework.
 - La **Dependency Injection**.
- Una volta configurati i componenti dell'applicazione, è possibile utilizzarli all'interno del **repository** Spring.

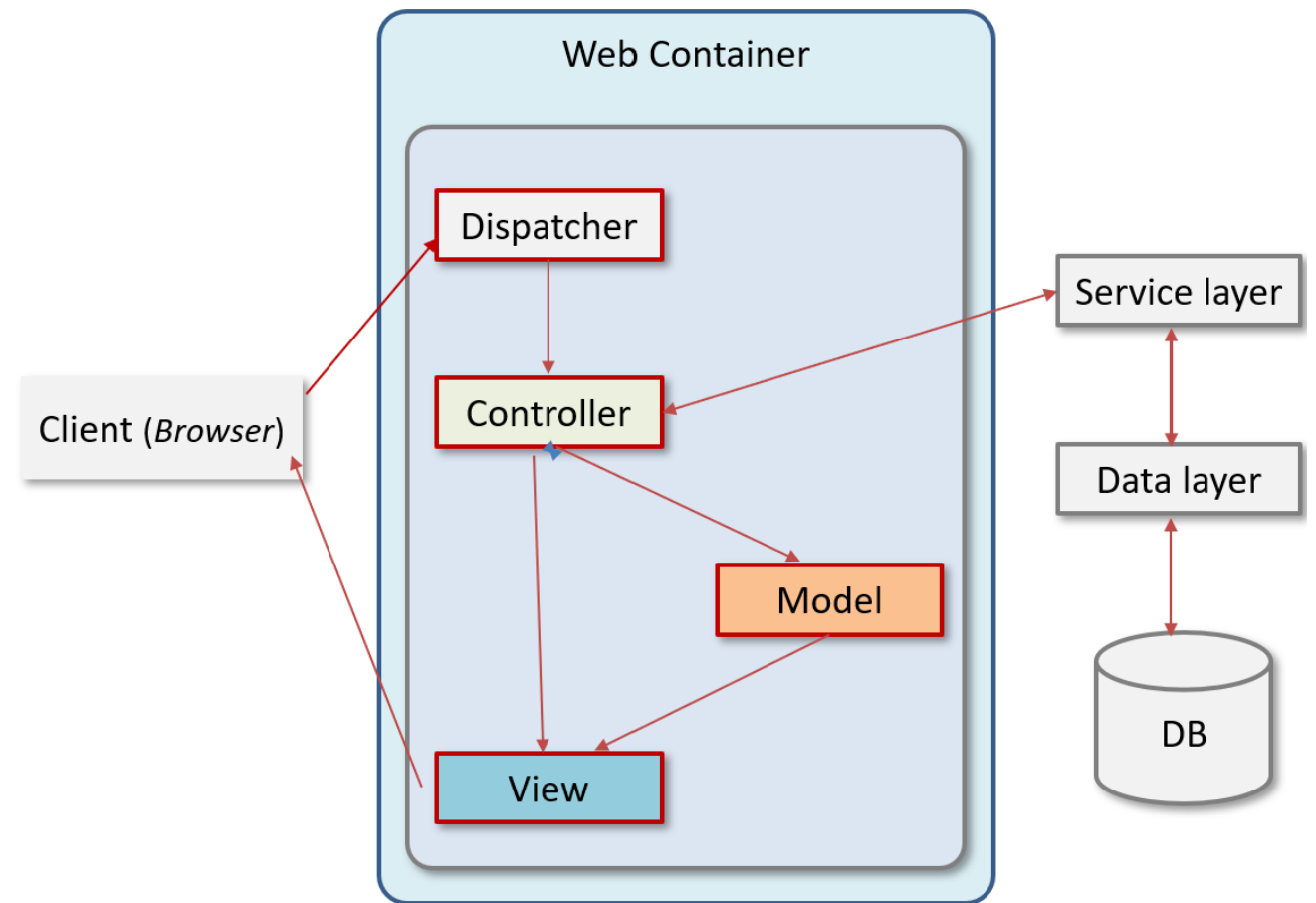


Sezione: Spring Data Access

Abbiamo visto:

- ✓ Introduzione a Spring Data.
- ✓ Componenti.
- ✓ Spring Data e JPA.

Spring Data - Configurazione



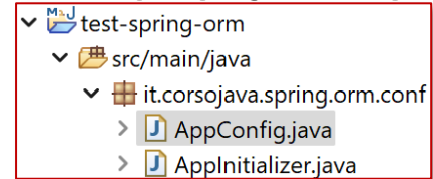
Sezione: Configurazione di Spring Data

Cosa vedremo:

- ☐ Configurazione Spring Data.
- ☐ Configurazione Entity.
- ☐ Configurazione DAO.
- ☐ Configurazione Controller.

Spring Data viene impostato definendo **tre componenti** base per utilizzare JPA e Hibernate nel file di configurazione dell'applicazione.

- **Datasource:**
 - Il bean di configurazione del DataSource imposta i parametri di connessione al database.
 - Con JPA queste informazioni erano nel persistence.xml.
- **LocalContainerEntityManager:**
 - Crea un **EntityManager JPA**.
 - Qui vanno impostate le informazioni riguardanti:
 - Implementazione di JPA (Hibernate in questo caso).
 - Il database utilizzato (dialetto SQL).
 - Il package di partenza da cui iniziare a scansionare i bean.
 - In questo modo forniamo le informazioni del persistence.xml e Spring crea un proprio file persistence interno.



```
// Bean per la connessione al database. Restituisce un DataSource
@Bean
public DataSource getDbConn() {
    DriverManagerDataSource ds = new DriverManagerDataSource();
    ds.setDriverClassName("oracle.jdbc.OracleDriver");
    ds.setUsername("corso");
    ds.setPassword("root");
    ds.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

    return ds;
}
```

```
// Bean per la definizione e creazione del repository JPA
@Bean
public LocalContainerEntityManagerFactoryBean getEntityMan() {
    HibernateJpaVendorAdapter hJpaVendorAdapter = new HibernateJpaVendorAdapter();
    hJpaVendorAdapter.setDatabase(Database.ORACLE);
    // opzione per la creazione automatica delle tabelle
    // hJpaVendorAdapter.setGenerateDdl(true);

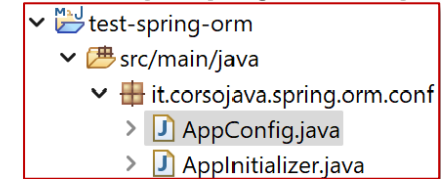
    LocalContainerEntityManagerFactoryBean factoryBean =
        new LocalContainerEntityManagerFactoryBean();
    factoryBean.setDataSource(getDbConn());
    factoryBean.setJpaVendorAdapter(hJpaVendorAdapter);
    // L'istruzione successiva funziona solamente se il file
    // è sulla root del progetto
    // factoryBean.setPackagesToScan(getClass().getPackage().getName());
    factoryBean.setPackagesToScan("it.corsojava.spring.orm");

    return factoryBean;
}
```

Configurazione di Spring Data

Spring Data viene impostato definendo **tre componenti** base per utilizzare JPA e Hibernate nel file di configurazione dell'applicazione.

- **JpaTransactionManager:**
 - Crea un **TransactionManager JPA**.
 - Gestisce le transazioni per le operazioni di scrittura sul database.

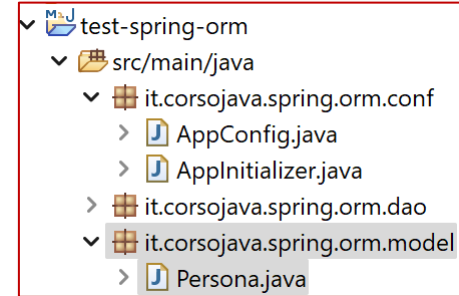


```
// Bean per la gestione delle transazioni
@Bean
public PlatformTransactionManager getTransactionManager() {
    JpaTransactionManager jtm = new JpaTransactionManager();
    jtm.setEntityManagerFactory(getEntityManager().getEntityManagerFactory());
    return jtm;
}
```

Configurazione

Configurazione delle Entity

Utilizzando Spring, le classi **Entity** vengono configurate come visto per JPA.



```
@Entity
public class Persona implements Serializable {
    private static final long serialVersionUID = 1L;

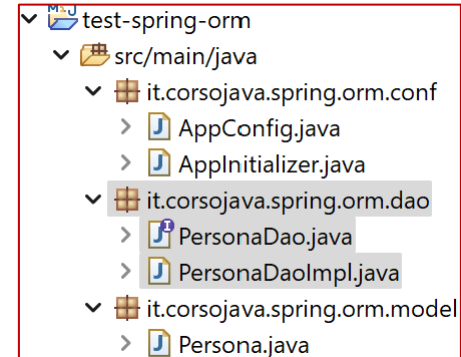
    @Id
    @SequenceGenerator(name="pers_seq", sequenceName = "persona_seq",
        allocationSize = 1)
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
        generator = "pers_seq")
    private Long id;
    @Column
    private String nome;
    @Column
    private String indirizzo;
    @Column
    private String email;

    public Persona() {
    }
```

Configurazione

Configurazione del DAO

Lo strato DAO prevede la dichiarazione di un'interfaccia e di un'implementazione per ciascun DAO e utilizza il model.



```
public interface PersonaDao {
    public void inserisci(Persona p);
    public Persona seleziona(Long id);
    public void aggiorna(Persona p);
    public void elimina(Long id);
}
```

```
public class PersonaDaoImpl implements PersonaDao{
    // Definizione EntityManager
    @PersistenceContext()
    private EntityManager em;

    @Override
    @Transactional
    public void inserisci(Persona p) {
        em.persist(p);
    }

    @Override
    public Persona seleziona(Long id) {
        Persona p = em.find(Persona.class, id);
        return p;
    }
}
```

```
    @Override
    @Transactional
    public void aggiorna(Persona p) {
        em.merge(p);
    }

    @Override
    @Transactional
    public void elimina(Long id) {
        Persona p = em.find(Persona.class, id);
        em.remove(p);
    }
}
```

Configurazione

Configurazione del Controller

Il Controller dichiara il DAO – configurato per l'applicazione - e lo utilizza richiamando i metodi esposti.

```
@Controller
@RequestMapping("/persona")
public class PersonaController {
    @Autowired
    private PersonaDao persDao;

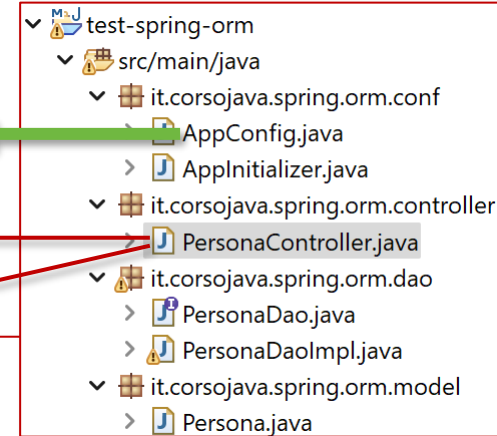
    @ResponseBody
    @RequestMapping("/seleziona")
    public String seleziona() {
        // return "Ok!";
        String risultato=persDao.seleziona(Long.valueOf(5)).toString();
        return risultato;
    }

    @ResponseBody
    @RequestMapping("/inserisci")
    public String inserisci() {
        Persona p=new Persona(null, "Piero", "Pavia", "pi@ero.it");
        persDao.inserisci(p);
        return "Inserimento Ok!";
    }
}
```

```
@Bean
public PersonaDao getPersonaDao() {
    return new PersonaDaoImpl();
}
```

```
@ResponseBody
@RequestMapping("/aggiorna")
public String aggiorna() {
    Persona p=persDao.seleziona(215L);
    p.setIndirizzo("Messina");
    p.setEmail("pie@ro.com");
    persDao.aggiorna(p);
    return "Aggiornamento Ok!";
}

@ResponseBody
@RequestMapping("/elimina")
public String elimina() {
    persDao.elimina(192L);
    return "Eliminazione Ok!";
}
```



Esempio di file di configurazione dell'applicazione completo.

```
@EnableWebMvc
@Configuration
@ComponentScan(basePackages = "it.corsojava.spring.orm")
@EnableTransactionManagement

public class AppConfig {

    @Bean
    public ViewResolver appResolver() {
        InternalResourceViewResolver resolver =
            new InternalResourceViewResolver();
        resolver.setViewClass(JstlView.class);
        resolver.setPrefix("/WEB-INF/view/");
        resolver.setSuffix(".jsp");
        return resolver;
    }

    // Bean per la connessione al database. Restituisce un DataSource
    @Bean
    public DataSource getDbConn() {
        DriverManagerDataSource ds = new DriverManagerDataSource();
        ds.setDriverClassName("oracle.jdbc.OracleDriver");
        ds.setUsername("corso");
        ds.setPassword("root");
        ds.setUrl("jdbc:oracle:thin:@localhost:1521:xe");

        return ds;
    }
}
```

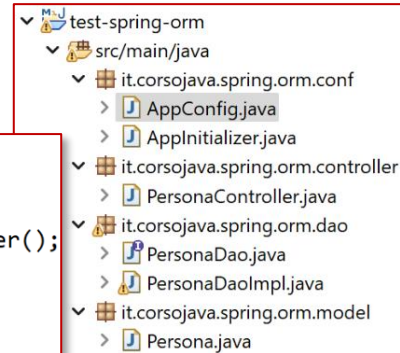
```
// Bean per la definizione e creazione dell'EntityManager
@Bean
public LocalContainerEntityManagerFactoryBean getEntityMan() {
    HibernateJpaVendorAdapter hJpaVendAdapter = new HibernateJpaVendorAdapter();
    hJpaVendAdapter.setDatabase(Database.ORACLE);
    // opzione per la creazione automatica delle tabelle
    // hJpaVendAdapter.setGenerateDdl(true);

    LocalContainerEntityManagerFactoryBean factoryBean =
        new LocalContainerEntityManagerFactoryBean();
    factoryBean.setDataSource(getDbConn());
    factoryBean.setJpaVendorAdapter(hJpaVendAdapter);
    factoryBean.setPackagesToScan("it.corsojava.spring.orm");

    return factoryBean;
}

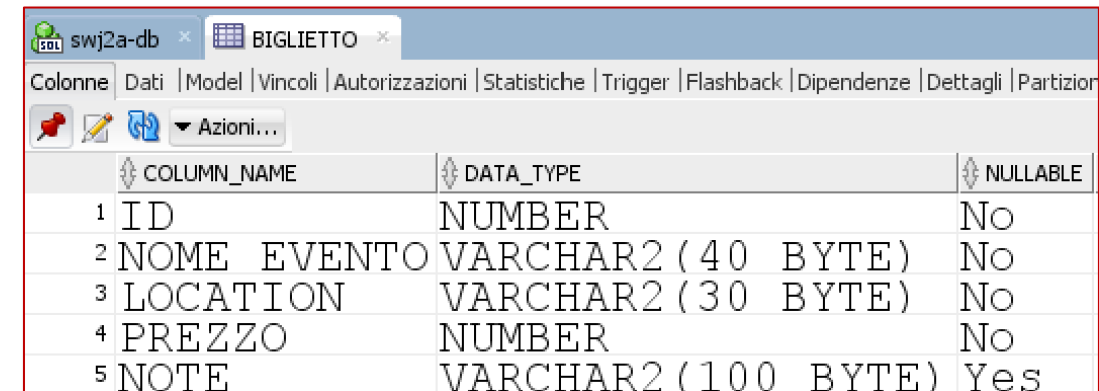
// Bean per la gestione delle transazioni
@Bean
public PlatformTransactionManager getTransactionManager() {
    JpaTransactionManager jtm = new JpaTransactionManager();
    jtm.setEntityManagerFactory(getEntityMan().getObject());
    return jtm;
}

@Bean
public PersonaDao getPersonaDao() {
    return new PersonaDaoImpl();
}
```



Esercizio: Creazione progetto Spring Data

- Riprendere il progetto '**esercizio-spring-mvc**' e gestire la persistenza utilizzando i componenti di Spring Data.
- Gestire l'inserimento, la ricerca per chiave primaria, la visualizzazione della lista delle Entity **Biglietto**.
- Creare le relative view.



The screenshot shows the SQL Developer interface with the 'BIGLIETTO' table selected. The table structure is displayed in a grid with columns: COLUMN_NAME, DATA_TYPE, and NULLABLE. The table has five columns: ID (NUMBER, No), NOME EVENTO (VARCHAR2(40 BYTE), No), LOCATION (VARCHAR2(30 BYTE), No), PREZZO (NUMBER, No), and NOTE (VARCHAR2(100 BYTE), Yes).

	COLUMN_NAME	DATA_TYPE	NULLABLE
1	ID	NUMBER	No
2	NOME EVENTO	VARCHAR2 (40 BYTE)	No
3	LOCATION	VARCHAR2 (30 BYTE)	No
4	PREZZO	NUMBER	No
5	NOTE	VARCHAR2 (100 BYTE)	Yes

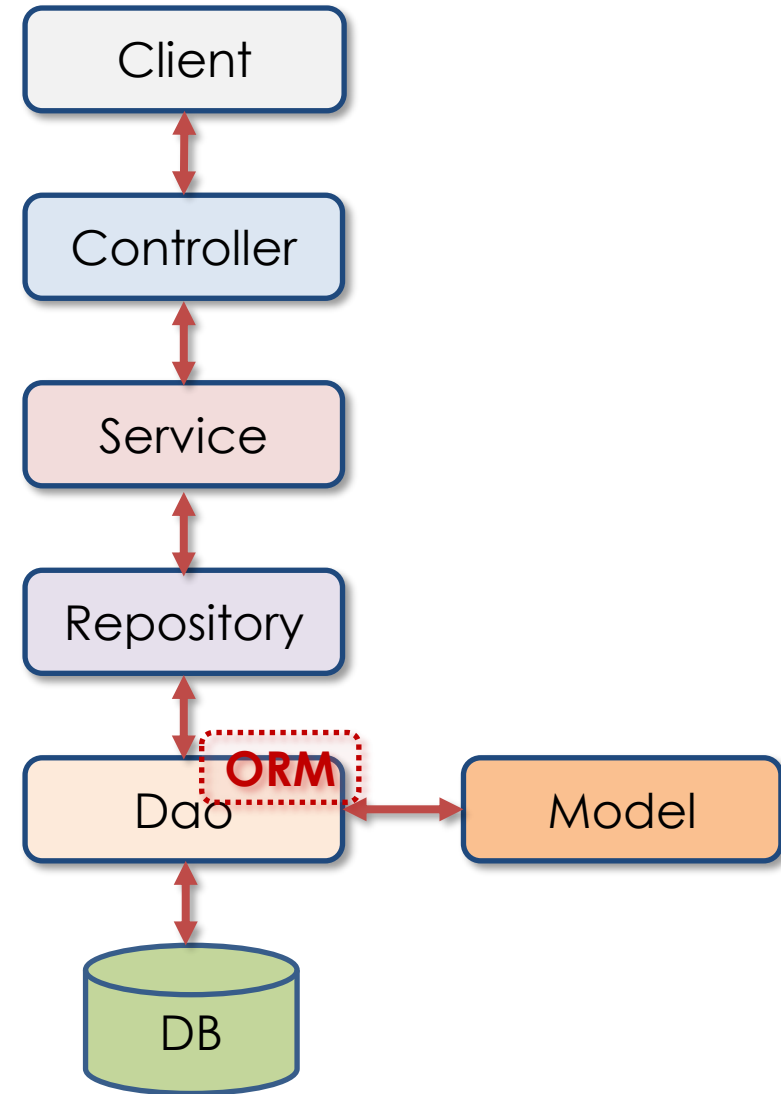


Sezione: Configurazione di Spring Data

Abbiamo visto:

- ✓ Configurazione Spring Data.
- ✓ Configurazione Entity.
- ✓ Configurazione DAO.
- ✓ Configurazione Controller.

Spring Data Repository



Sezione: Spring Data Repository

Cosa vedremo:

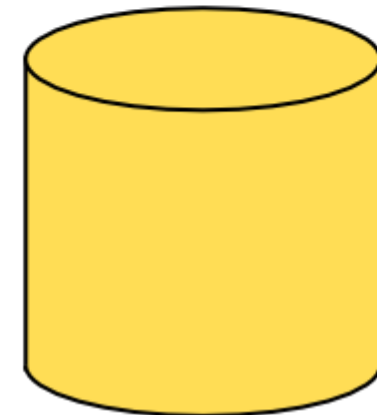
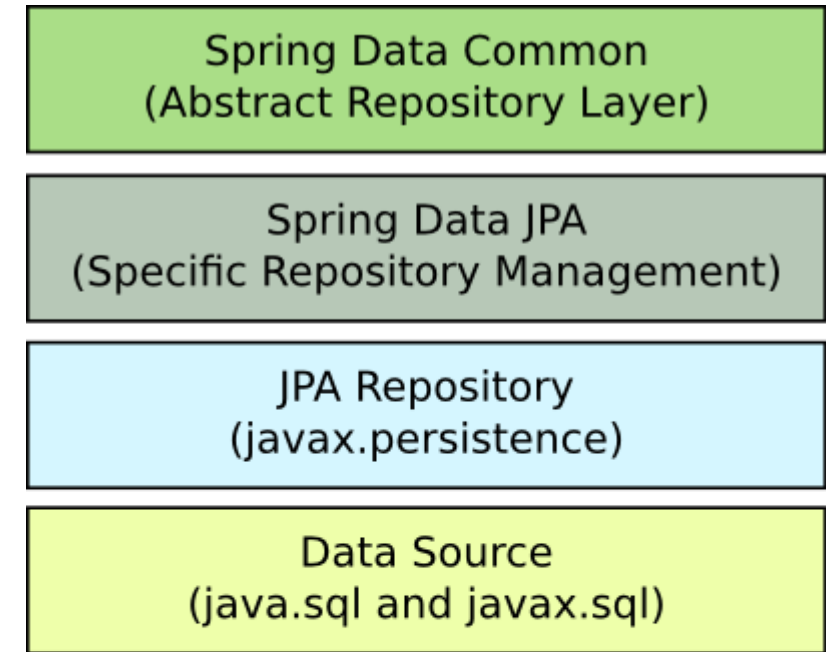
- ❑ Introduzione.
- ❑ Componenti principali.
- ❑ Utilizzo.
- ❑ Sintassi.
- ❑ Esempio pratico.

Spring Data Repository

Introduzione

Strumenti per semplificare la gestione dei metodi di lettura e scrittura su DB.

- Consente di sfruttare un livello di automatismo nelle operazioni di accesso ai dati.
 - Un ulteriore livello di astrazione gestito da Spring Data.
 - Consente di automatizzare la creazione dei metodi DAO.
 - Semplificando ulteriormente la creazione di questi metodi.



Spring Data Repository

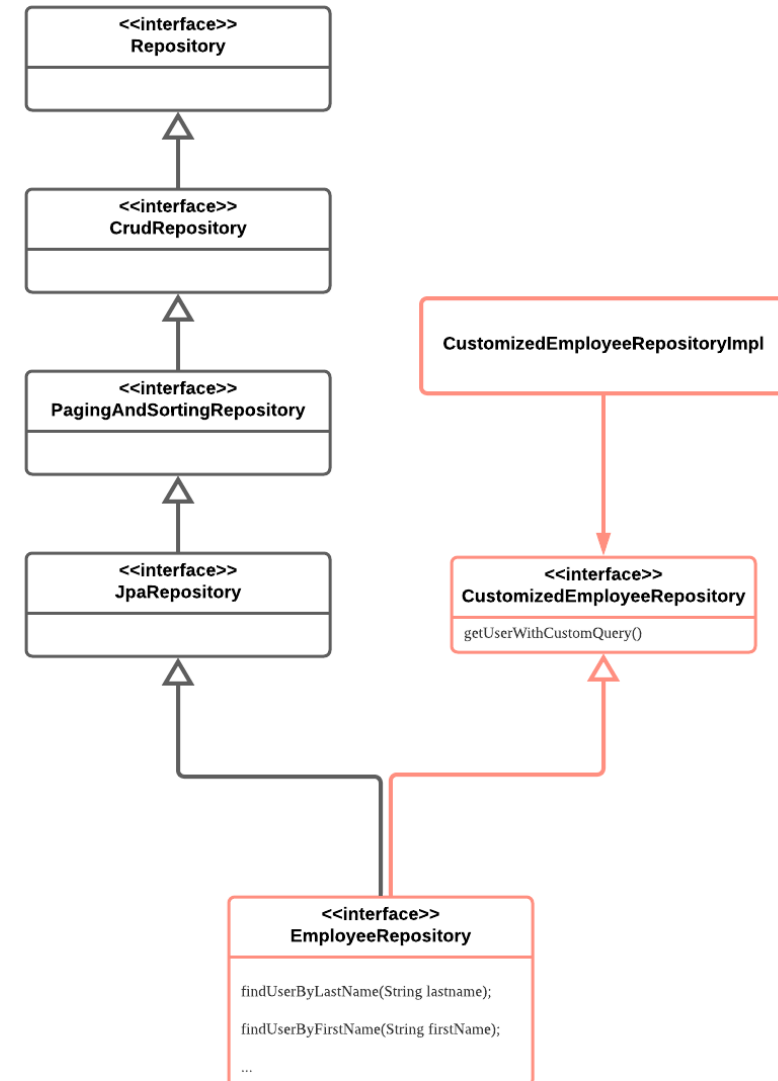
Componenti principali

Strumenti per semplificare la gestione dei metodi di lettura e scrittura su DB.

- La base di Spring Data Repository è costituita da due interfacce
 - **Repository**
 - Identifica le interfacce predisposte per l'accesso ai dati.
 - **CrudRepository**
 - Definisce le principali CRUD.
- **Query Method**

Spring Data consente inoltre di definire metodi in un'interfaccia che:

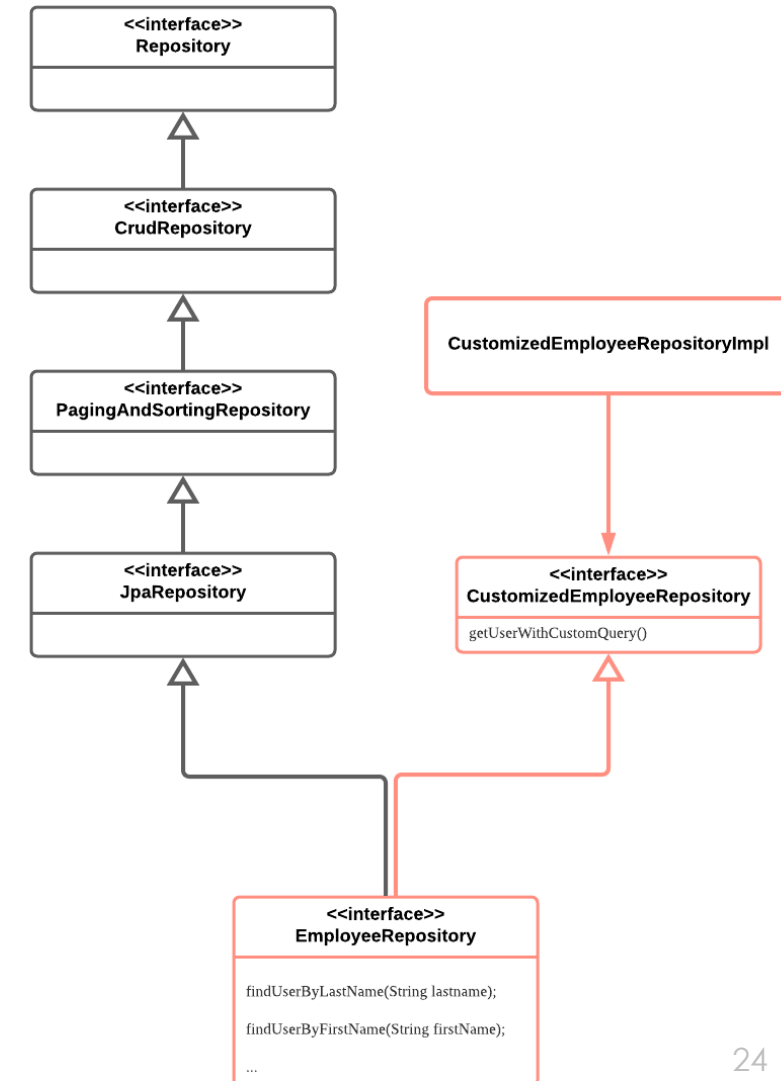
 - Estende CrudRepository
 - Utilizzano una sintassi predefinita
 - ***Verranno implementati automaticamente da Spring.***



Utilizzo di Spring Data Repository

Per utilizzare Spring Data Repository è necessario effettuare le seguenti operazioni.

1. Creare un'interfaccia che estende CrudRepository.
2. Definire i metodi all'interno dell'interfaccia.
3. Utilizzare l'annotation **@EnableJpaRepositories** nella classe di configurazione dell'applicazione
 - Per abilitare l'utilizzo dei Repository
 - Specifica il/i package/s di partenza per la ricerca delle interfacce da implementare.



Spring Data Repository

Sintassi di Spring Data Repository

Spring Data gestisce una serie di parole chiave con le quali è possibile definire i nomi dei metodi di accesso al db.

Esempio di nomi di metodi:

- **find<...>By, get<...>By.**

Es:

- **find**Persona**By**Nome, **get**Persona**By**Indirizzo.

È possibile definire metodi in grado di effettuare query complesse utilizzando keywords Spring Data:

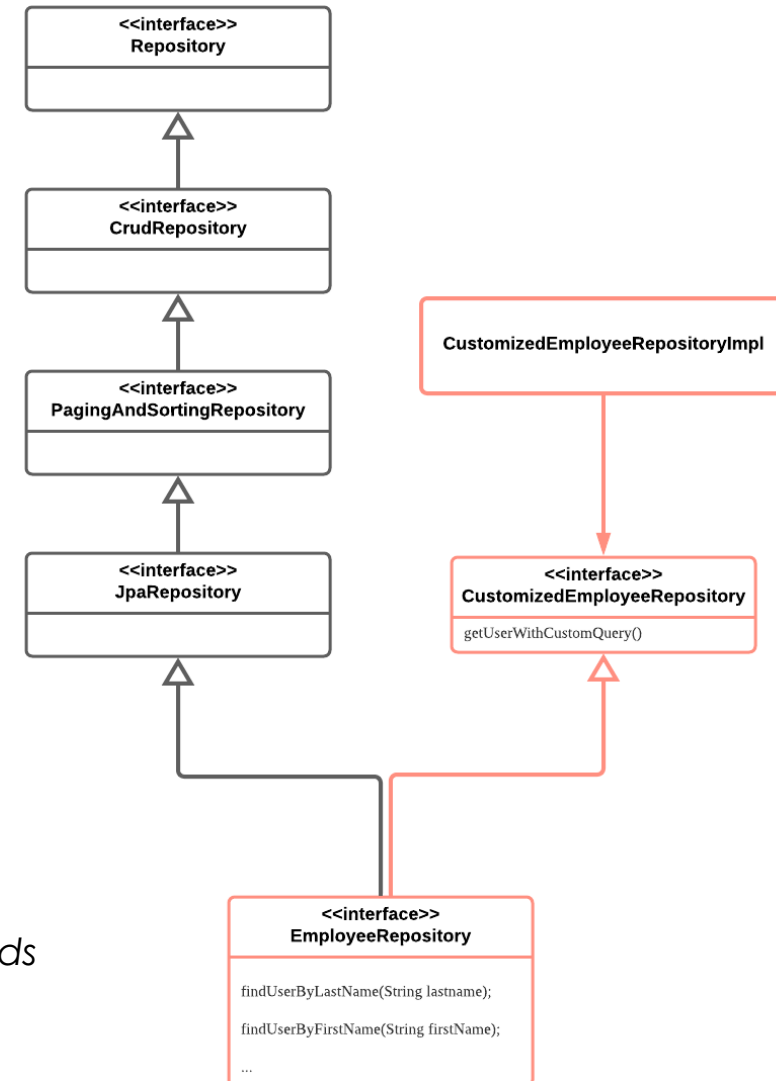
- And, Or, In, ...

Es:

- **findBy**Nome**And**Indirizzo, **findBy**Indirizzo**Or**Email.

Elenco delle keyword:

<https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repository-query-keywords>



Spring Data Repository

Esempio pratico

Controller.

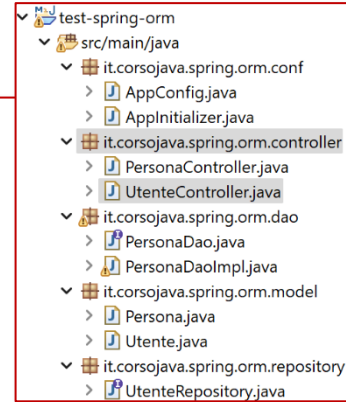
Come visto precedentemente, nel controller andiamo a definire i metodi per le CRUD,

```
@Controller
@RequestMapping("/utente")
public class UtenteController {
    @Autowired
    private UtenteRepository repository;

    @ResponseBody
    @GetMapping("/ins")
    public String inserisci() {
        Utente ut = new Utente("test01", "ped01", "utente di test", "te@st.com");
        repository.save(ut);
        return "inserimento ok";
    }

    @ResponseBody
    @GetMapping("/selectAll")
    public String seleziona() {
        Iterable<Utente> utenti=repository.findAll();
        String ritorno="";
        for(Utente u : utenti) {
            ritorno += u + "<br>";
        }
        return "selezione ok<br>" + ritorno;
    }

    @ResponseBody
    @GetMapping("/selectById")
    public String selezionaDaId() {
        Optional<Utente> utente=repository.findById(2L);
        return "selezione ok<br>" + utente.get();
    }
}
```

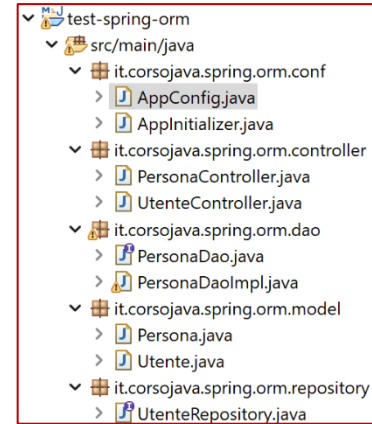


Spring Data Repository

Esempio pratico

Abilitare l'utilizzo del repository nel file di configurazione dell'applicazione.

- Inserire l'annotation **@EnableJpaRepositories**
 - Questa annotation prevede che nella classe di configurazione siano presenti i metodi (Bean) per la gestione di:
 - EntityManager
 - TransactionManager



```
@EnableWebMvc
@Configuration
@ComponentScan(basePackages = "it.corsojava.spring.orm")
@EnableTransactionManagement
@EnableJpaRepositories(basePackages="it.corsojava.spring.orm.repository",
                      entityManagerFactoryRef = "emf",
                      transactionManagerRef = "tmf")
public class AppConfig {
```

```
// Bean per la definizione e creazione dell'EntityManager
@Bean(name="emf")
public LocalContainerEntityManagerFactoryBean getEntityManager() {
    HibernateJpaVendorAdapter hJpaVendorAdapter = new HibernateJpaVendorAdapter();
```

```
// Bean per la gestione delle transazioni
@Bean(name="tmf")
public PlatformTransactionManager getTransactionManager() {
    JpaTransactionManager jtm = new JpaTransactionManager();
```

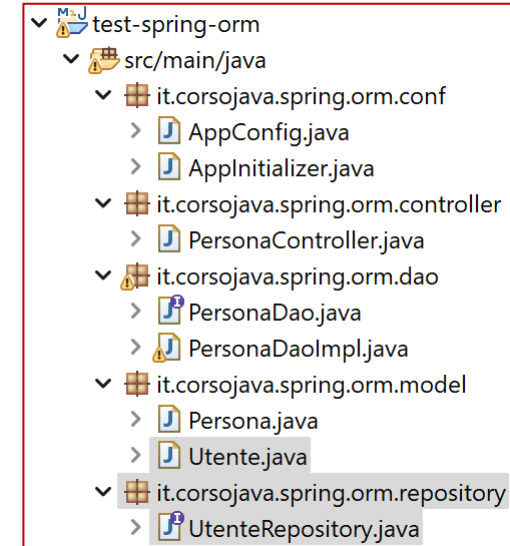
Spring Data Repository

Esempio pratico

Creazione del Repository.

Nel repository è sufficiente definire l'interfaccia che estende CrudRepository

- **I metodi di default vengono implementati da Spring Data.**



```
package it.corsojava.spring.orm.repository;

import org.springframework.data.repository.CrudRepository;
import it.corsojava.spring.orm.model.Utente;

public interface UtenteRepository extends CrudRepository<Utente, Long>{

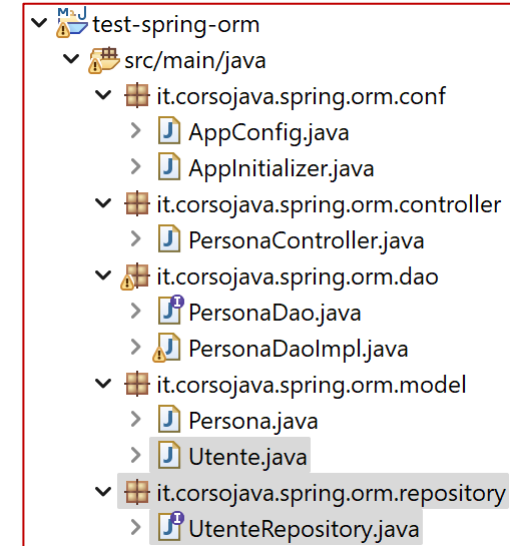
}
```

Spring Data Repository

Esempio pratico

Definizione di query personalizzate.

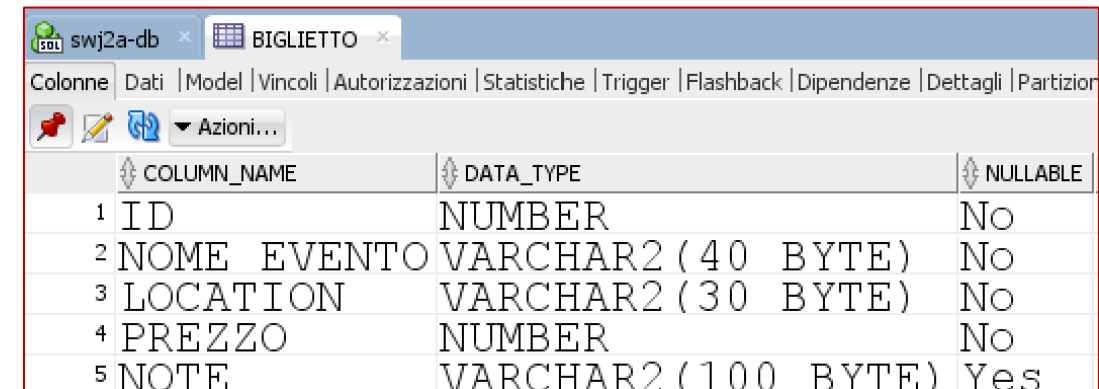
Nel repository è sufficiente definire solamente i metodi per le query personalizzate.



```
public interface UtenteRepository extends CrudRepository<Utente, Long>{  
    public Utente findByName(String nome);  
    // public Utente findUtenteByName(String nome);  
    public Utente getByEmail(String email);  
    public List<Utente> findByNameLike(String nomeLike);  
}
```

Esercizio: Creazione progetto Spring Data

- Riprendere il progetto '**esercizio-spring-mvc**' e gestire la persistenza utilizzando i componenti di Spring Data.
- Gestire l'inserimento, la ricerca per chiave primaria, la visualizzazione della lista, la modifica e l'eliminazione delle Entity **Biglietto**.
- Creare le relative view.



The screenshot shows the SQL Developer interface with the 'BIGLIETTO' table selected. The table structure is as follows:

	COLUMN_NAME	DATA_TYPE	NULLABLE
1	ID	NUMBER	No
2	NOME EVENTO	VARCHAR2 (40 BYTE)	No
3	LOCATION	VARCHAR2 (30 BYTE)	No
4	PREZZO	NUMBER	No
5	NOTE	VARCHAR2 (100 BYTE)	Yes



Sezione: Spring Data Repository

Abbiamo visto:

- ✓ Introduzione.
- ✓ Componenti principali.
- ✓ Utilizzo.
- ✓ Sintassi.
- ✓ Esempio pratico.

Spring Data

Indice argomenti

- Introduzione a Spring Data
- Configurazione di Spring Data
- Spring Data Repository