Spring Data JPA

Arnaud Cogoluègnes

Zenika

27-28 mars 2012

Plan

Plan

`LocalContainerEntityManagerFactoryBean`

- Pour la configuration de JPA dans Spring
- Fait partie du Spring Framework (module ORM)
- Flexible :
  - Injection de dépendances (ex. : `DataSource`)
  - Compatible avec les différentes implémentations JPA
  - Pas besoin de `persistence.xml`
  - Scanning de packages pour trouver les entités

Configuration de JPA avec Spring

```
<bean id="entityManagerFactory"
      class="o.s.orm.jpa.LocalContainerEntityManagerFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="jpaVendorAdapter">
    <bean class="o.s.orm.jpa.vendor.HibernateJpaVendorAdapter">
      <property name="showSql" value="true" />
      <property name="generateDdl" value="true" />
      <property name="database" value="H2" />
    </bean>
  </property>
  <property name="packagesToScan" value="com.zenika.nordnet.model" />
</bean>
```

Entité JPA

```java
package com.zenika.nordnet.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Contact {

  @Id
  @GeneratedValue(strategy = GenerationType.SEQUENCE)
  private Long id;

  private String firstname, lastname;

  (...)
}
```

TestContext framework

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/spring-jpa.xml")
public class SpringJpaTest {

  @Autowired private EntityManagerFactory emf;

  @Test public void springJpa() {
    (...)
  }

}
```

Méthode de test

```
@Test public void springJpa() {
  EntityManager em = emf.createEntityManager();
  em.getTransaction().begin();
  try {
    Contact contact = new Contact();
    contact.setFirstname("Mickey");
    contact.setLastname("Mouse");
    int initialCount = em.createQuery("from Contact")
      .getResultList().size();
    em.persist(contact);
    Assert.assertEquals(
      initialCount+1,
      em.createQuery("from Contact").getResultList().size()
    );
    em.getTransaction().commit();
  } finally {
    em.close();
  }
}
```

Configuration Java (alternative)

```java
@Configuration
public class SpringJpaConfiguration {

  @Bean public EntityManagerFactory emf() {
    LocalContainerEntityManagerFactoryBean emf =
      new LocalContainerEntityManagerFactoryBean();
    emf.setDataSource(ds());
    HibernateJpaVendorAdapter adapter = new HibernateJpaVendorAdapter();
    adapter.setGenerateDdl(true);
    emf.setJpaVendorAdapter(adapter);
    emf.setPackagesToScan(Contact.class.getPackage().getName());
    emf.afterPropertiesSet();
    return emf.getObject();
  }

  @Bean public DataSource ds() {
    return new EmbeddedDatabaseBuilder()
      .setType(EmbeddedDatabaseType.H2)
      .build();
  }

}
```

Configuration Java dans le test

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(classes=SpringJpaConfiguration.class)
public class SpringJpaJavaConfigurationTest {

  @Autowired private EntityManagerFactory emf;

  @Test public void springJpa() {
    (...)
  }

}
```

Plan

Interface de repository

- Utiliser `JpaRepository` comme interface de base
- Typer l'interface avec la classe d'entité et la classe de l'identifiant

```
package com.zenika.nordnet.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.zenika.nordnet.model.Contact;

public interface ContactRepository
       extends JpaRepository<Contact,Long> {

}
```

Déclarer les repositories

- Indiquer à Spring Data JPA les packages des repositories
- Spring détecte les repositories et crée les beans correspondants

```
<jpa:repositories base-package="com.zenika.nordnet.repository" />
```

- NB : utilisation du namespace `jpa`

Infrastructure

- Dépendances de `<jpa:repositories />` :
  - `EntityManagerFactory`
  - `PlatformTransactionManager`

```
<bean id="entityManagerFactory"
      class="o.s.orm.jpa.LocalContainerEntityManagerFactoryBean">
  (...)
</bean>

<bean id="transactionManager"
      class="org.springframework.orm.jpa.JpaTransactionManager">
  <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>
```

Test !

```java
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration("/spring-data-jpa-hello-world.xml")
public class SpringDataJpaHelloWorldTest {

  @Autowired ContactRepository repo;

  @Test public void springJpa() {
    Contact contact = new Contact();
    contact.setFirstname("Mickey");
    contact.setLastname("Mouse");
    long initialCount = repo.count();
    repo.save(contact);
    Assert.assertEquals(initialCount + 1, repo.count());
  }

}
```

Plan

Interface `Repository`

- Interface “marker” (ne définit pas de méthode)
- Définie dans Spring Data Commons

Interface `CrudRepository`

- Hérite de `Repository`
- Définit les méthodes CRUD de base
  - `save`, `findOne`, `count`, `findAll`, etc.
- Définie dans Spring Data Commons

Interface `PagingAndSortingRepository`

- Hérite de `CrudRepository`
- Définit les méthodes de pagination et de tri
- Définie dans Spring Data Commons

Interface `JpaRepository`

- Hérite de `PagingAndSortingRepository`
- Permet d'utiliser des `List` plutôt que des `Iterable`
- Rajoute la notion de `flush`
- Définie dans Spring Data JPA

Classe `SimpleJpaRepository`

- Implémentation de `JpaRepository`
- Utilisée par défaut via `<jpa:repositories />`
- Le développeur ne manipule généralement pas cette classe

Plan

Ajout de méthodes sur une interface

- Ajouter une méthode en suivant des conventions de nommage
- Spring Data JPA construit la requête à l'exécution

```
public interface ContactRepository extends Repository<Contact,Long> {

  List<Contact> findByLastname(String lastname);

}
```

Convention pour les opérateurs

- Support pour and et or

```
public interface ContactRepository extends Repository<Contact,Long> {

  List<Contact> findByFirstnameAndLastname(String firstname,
                                           String lastname);

}
```

Conventions

- `and : findByLastnameAndFirstname`
- `or : findByLastnameOrFirstname`
- `between : findByStartDateBetween`
- `less than : findByAgeLessThan`
- `greater than : findByAgeGreaterThan`

Conventions

- `is null` : `findByAgeIsNull`
- `is not null`, `not null` : `findByAge(Is)NotNull`
- `like` : `findByFirstnameLike`
- `not like` : `findByFirstnameNotLike`
- `order by` : `findByAgeOrderByLastnameDesc`

Conventions

- `not` : `findByLastnameNot`
- `in` : `findByAgeIn`
- `not in` : `findByAgeNotIn`

Un seul enregistrement retourné

- Déclarer le type de l'entité en retour si un seul résultat est attendu
- Retourne `null` si rien n'est trouvé
- Lance une exception si plus d'un résultat est trouvé

```
public interface ContactRepository extends Repository<Contact,Long> {

  Contact findByFirstnameAndLastname(String firstname,
                                     String lastname);

}
```

Repository CRUD + méthodes par convention

```
public interface ContactRepository extends JpaRepository<Contact,Long> {

  List<Contact> findByLastname(String lastname);

}
```

Repository avec seulement des méthodes par convention

```
public interface ContactRepository extends Repository<Contact,Long> {

  List<Contact> findByLastname(String lastname);

}
```

Plan

Pourquoi un repository custom ?

- Quand les méthodes CRUD ne suffisent pas
- Quand les méthodes par convention de nommage ne suffisent pas
- On peut alors implémenter notre propre repository
- Spring Data JPA le combinera avec un repository dynamique

Déclaration des interfaces

```
public interface ContactRepositoryCustom {

  List<Contact> findByExample(Contact contact);

}

public interface ContactRepository extends ContactRepositoryCustom,
                                           Repository<Contact,Long> {

  List<Contact> findByLastname(String lastname);

}
```

Implémentation custom

```
public class ContactRepositoryImpl implements ContactRepositoryCustom {

  @PersistenceContext private EntityManager em;

  @Override
  public List<Contact> findByExample(Contact contact) {
    (...)
  }

}
```

Conventions

- Interfaces et implémentation dans le même package
- Classe d'implémentation doit finir par `Impl`
- Le repository est détecté et combiné automatiquement

Si le postfix par défaut ne convient pas

- Postfix `Jpa` plutôt que `Impl`

```
<jpa:repositories base-package="com.zenika.nordnet.repository"
                  repository-impl-postfix="Jpa" />
```