Repository les bases

Démonstration 3 du module 5

Les objectifs de cette démonstration sont

- De créer notre premier Repository de Spring Data JPA
- De voir les contraintes du cycle de vie des entités

Contexte

- Compléter le projet de démonstration précédent

Déroulement

Couche DAL

- Placer l'annotation @Profile(«dev») sur la classe CourseDAOMock ; ainsi ce bean ne sera plus utilisé par Spring

```
package fr.eni.demo.dal.mock;

import java.util.ArrayList;
import java.util.List;

import org.springframework.context.annotation.Profile;
import org.springframework.stereotype.Repository;

import fr.eni.demo.bo.Course;
import fr.eni.demo.dal.CourseDAO;

@Repository
@Profile("dev")
public class CourseDAOMock implements CourseDAO {
```

- Créer une nouvelle interface CourseRepository
    - Elle hérite de JpaRepository

```
package fr.eni.demo.dal;

import org.springframework.data.jpa.repository.JpaRepository;

import fr.eni.demo.bo.Course;
//Création de l'interface pour utiliser ORM JPA
public interface CourseRepository extends JpaRepository<Course, Long>{

}
```

- Rien de plus à faire au niveau de la couche DAL.

Couche BLL

- Modifier la classe TrainerServiceImpl pour lui injecter CourseRepository à la place de CourseDAO
    - Remplacer l'appel de la méthode read par getById

```
package fr.eni.demo.bll;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import fr.eni.demo.bo.Course;
import fr.eni.demo.dal.CourseRepository;

@Service
public class CourseServiceImpl implements CourseService {
	//Injection du Repository de Spring Data
	private CourseRepository courseDAO;

	@Autowired
	public CourseServiceImpl(CourseRepository courseDAO) {
		this.courseDAO = courseDAO;
	}

	@Override
	public List<Course> findAll() {
		return courseDAO.findAll();
	}

	@Override
	public Course findById(long id) {
		return courseDAO.getById(id);
	}
}
```

- Voilà au niveau métier ce qu'il faut faire.

Configuration d'un bean pour avoir des cours par défaut

- Dans notre application de démonstration, nous avions une liste de cours chargée par la CourseDAOMock.
- Remplaçons là par un bean qui va nous injecter par le repository la liste des cours par défaut
- Ajout d'une nouvelle méthode dans la classe WebConfiguration pour créer ce bean
    - Il y a plus de cours pour faire la différence avec la liste précédente

```
//Pour injecter par défaut des cours en base.
@Bean
public CommandLineRunner demo(CourseRepository repository) {
	return (args) -> {
		// ajouter des cours
		repository.save(new Course(10, "Algorithmique", 5));
		repository.save(new Course(20, "Initiation à la programmation", 5));
		repository.save(new Course(30, "POO", 10));
		repository.save(new Course(50, "SQL", 5));
		repository.save(new Course(60, "PL-SQL", 5));
		repository.save(new Course(130, "Web Client", 5));
		repository.save(new Course(140, "Symfony", 10));
		repository.save(new Course(220, "frameworks Java EE", 10));
```

```
            // Tracer la liste créée
            System.out.println("Liste des cours : ");
            repository.findAll().forEach(course -> System.out.println(course));
        };
}
```

Exécution :

- Constater dans les traces :
- Création de la table COURSES

```
Hibernate:

    drop table if exists courses
Hibernate:

    create table courses (
       id bigint not null auto_increment,
        duration integer not null,
        title varchar(250) not null,
        primary key (id)
    ) engine=InnoDB
```

- Insertion des cours, pour chaque insert :

```
Hibernate:
    select
        course0_.id as id1_0_0_,
        course0_.duration as duration2_0_0_,
        course0_.title as title3_0_0_
    from
        courses course0_
    where
        course0_.id=?
Hibernate:
    insert
    into
        courses
        (duration, title)
    values
        (?, ?)
```

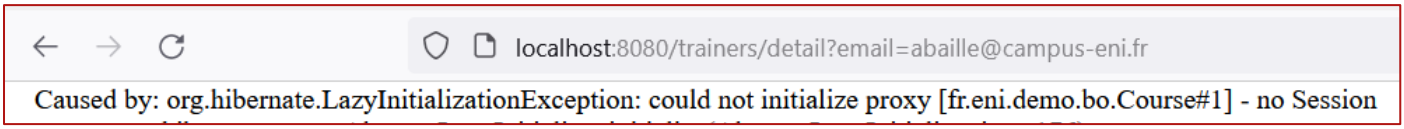
  - Il valide la clef et les valeurs avant l'insertion effectif.
  - Il faudra ne pas laisser les options de traces en production.

- Et la liste des cours ainsi générée :

```
Liste des cours :
Hibernate:
    select
        course0_.id as id1_0_,
        course0_.duration as duration2_0_,
        course0_.title as title3_0_
    from
        courses course0_
course [id=1, title=Algorithmique, duration=5]
course [id=2, title=Initiation à la programmation, duration=5]
course [id=3, title=POO, duration=10]
course [id=4, title=SQL, duration=5]
course [id=5, title=PL-SQL, duration=5]
course [id=6, title=Web Client, duration=5]
course [id=7, title=Symfony, duration=10]
course [id=8, title=frameworks Java EE, duration=10]
course [id=9, title=Ionic, duration=5]
course [id=10, title=JavaScript, duration=10]
```

Cycle de vie des entités :

- Si vous tester l'insertion d'un cours sur un formateur existant ; vous obtiendrez une erreur :



- org.hibernate.LazyInitializationException : could not initialize proxy – no Session.
- Le fait de gérer l'entité Course et pas pour le moment l'entité Trainer.
    - Induit que l'objet Course qui est associé à Trainer n'est pas conserver dans le contexte de l'ORM
    - Il lui ai donc impossible de retourner l'objet correctement pour l'affichage
- La session est un contexte de persistance qui représente une conversation entre une application et la base de données
    - Le Lazy Loading signifie que l'objet ne sera pas chargé dans le contexte Session tant qu'il n'aura pas été accédé dans le code par une méthode de l'ORM
    - Pour le moment, seul Course est manipulé par Spring Data JPA et pas Trainer. Dans la vue, c'est bien à travers l'association que l'on affiche Course.
- Pour pallier à cela temporairement, nous pouvons déclarer dans application.properties :

```
spring.jpa.properties.hibernate.enable_lazy_load_no_trans=true
```

    - Cette propriété Hibernate est utilisée pour déclarer une politique globale pour la récupération d'objets en lazy.
    - Par défaut, cette propriété est à fausse.
    - L'activer signifie que chaque accès à une entité sera dans une nouvelle session au travers d'une transaction

> Evidemment cette solution ne sera pas à conserver. Elle ralentit l'exécution totale en créant autant de transaction qu'il y a d'entités à afficher dans l'association.

- Exécuter de nouveau, tout fonctionne.