CS544 EA

Applications

Spring Data: Advanced Methods

Property Expressions

- Similar to HQL you can traverse properties

```
List<Student> findByAddressZipCode(ZipCode zipCode);// x.address.zipCode
```

- What if a student has a AddressZipCode?

```
List<Student> findByAddress_ZipCode(ZipCode zipCode);// x.address.zipCode
```

Optionally underscores can be used to separate properties

Java Methods should use CamelCase there should not be any conflicts caused by this

@Query

- What if you don’t like typing long names or need to have a complicated query?

```
public interface UserRepository extends JpaRepository<User, Long>{
    @Query("Select u from User u where u.emailAddress = ?1")
    User findByEmailAddrss(String emailAddress);
}
```

- Or want to write SQL instead of HQL?

```
public interface UserRepository extends JpaRepository<User, Long>{
    @Query(value = "SELECT * FROM USERS WHERE EMAIL_ADDRESS = ?0", nativeQuery = true)
    User findByEmailAddrss(String emailAddress);
}
```

Custom Functionality

- You can also write your own custom methods
  - Allowing you to do whatever you want

Step1: create an interface

```
public interface UserDaoCustom {
    void someCustomMethod(User user);
}
```

Step2: create an implementation

```
public class UserDaoImpl implements UserDaoCustom {
    void someCustomMethod(User user) {
        // your custom implementation
    }
}
```

Step3: add the interface to where you want to add the functionality

```
public interface UserRepository extends JpaRepository<User, Long>, UserDaoCustom {
}
```