***Writing JPQL with @Query annotation***

* Similar syntax to SQL query. Reference to entity class and fields inside it.
* Why need to write JPQL? (Write queries that you cannot achieve by method proxy that is provided by JPA)
* JPQL Select query
* JPQL Update query
* JPQL Delete query

***Select query with JPQL***

@Repository

@Query(“From Student where firstName = :firstName and lastName = : lastName”)

Student getByLastNameAndFirstName (String lastName, @Param(“firstname”) String firstName)

@Service

Public Student getByFirstNameAndLastName (String firstName, String lastName){

Return studentRepository.getByLastNameAndFirstName(lastName, firstName);

}

@RestController

…

***Update Query with JPQL***

@Repository

@Modifying

@Transactional

@Query(“Update Student set firstName = :firstName where id = : id”)

Integer updateFirstName (Long id, String firstName);

@Service

Public Integer updateStudentWithJpql (Long id, String firstName){

Return studentRepository.updateFirstName(id, firstname);

}

@RestController

@PutMapping(“updateFirstName/{id}/{firstName}”)

Public String updateStudentWithJpql (@PathVariable Long id, @PathVariable String firstName){

Return studentService.updateStudentWithJpql(id, firstName) + “ Student(s) updated”;

}

***Delete query with JPQL***

@Repository

@Modifying

@Transactional

@Query(“Delete From Student where firstName = : firstName”)

Integer deleteByFirstName (String firstName)

@Service

Public Integer deleteStudent (String firstName){

Return studentRepository.deleteByFirstName(firstName)”;

}

@RestController

@DeleteMapping(“deleteByFirstName/{firstName}”)

Public String deleteStudent (@PathVariable String firstName){

Return studentService.deleteStudent(firstName) + “ Student(s) deleted”;

}