

# Spring Data JPA

---

# Sadržaj

---

- Konfigurisanje Spring Data JPA aplikacije
- Spring Data JPA repozitorijumi
- Klase servisa

# Spring Data JPA projekat

---

- Omogućava efikasan razvoj *data* sloja aplikacije
- Koristi Hibernate implementaciju JPA
- Obezbeđuje spregu između JPA klasa i drugih komponenti srednjeg sloja

# Spring Data JPA zavisnost

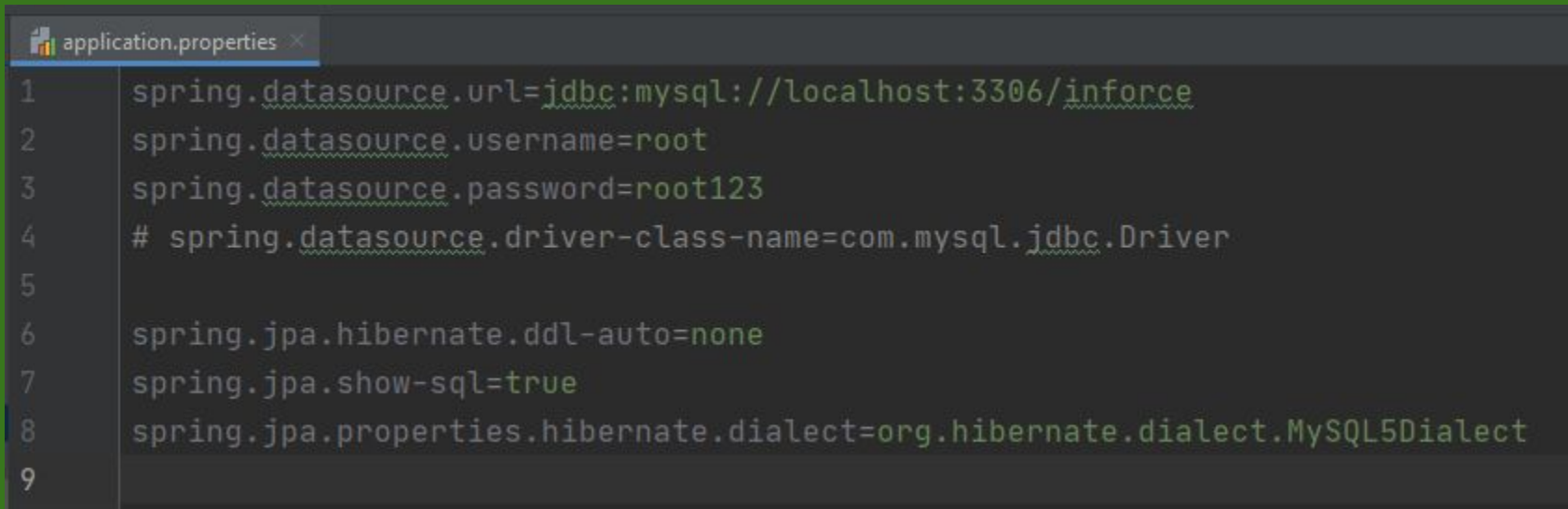
---

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>
```

- Ažurirati Maven (*update* ili *reimport*) projekat kako bi se skinule odgovarajuće biblioteke

# Konfigurisanje Spring Data JPA aplikacije

---



```
application.properties x
1  spring.datasource.url=jdbc:mysql://localhost:3306/inforce
2  spring.datasource.username=root
3  spring.datasource.password=root123
4  # spring.datasource.driver-class-name=com.mysql.jdbc.Driver
5
6  spring.jpa.hibernate.ddl-auto=none
7  spring.jpa.show-sql=true
8  spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
9
```

# Interfejs CrudRepository

## Method Summary

All Methods	Instance Methods	Abstract Methods
Modifier and Type	Method and Description	
long		<code>count()</code> Returns the number of entities available.
void		<code>delete(T entity)</code> Deletes a given entity.
void		<code>deleteAll()</code> Deletes all entities managed by the repository.
void		<code>deleteAll(Iterable&lt;? extends T&gt; entities)</code> Deletes the given entities.
void		<code>deleteById(ID id)</code> Deletes the entity with the given id.
boolean		<code>existsById(ID id)</code> Returns whether an entity with the given id exists.
Iterable<T>		<code>findAll()</code> Returns all instances of the type.
Iterable<T>		<code>findAllById(Iterable&lt;ID&gt; ids)</code> Returns all instances of the type T with the given IDs.
Optional<T>		<code>findById(ID id)</code> Retrieves an entity by its id.
<S extends T> S		<code>save(S entity)</code> Saves a given entity.
<S extends T> Iterable<S>		<code>saveAll(Iterable&lt;S&gt; entities)</code> Saves all given entities.

# Interface JpaRepository

- Izvršavanje CRUD operacija obezbeđuje `CrudRepository`
- Proširuje ga `PagingAndSortingRepository` koji obezbeđuje straničenje i sortiranje objekata
- `JpaRepository` proširuje interfejses
  - `PagingAndSortingRepository` i
  - `QueryByExampleExecutor`



# Definisanje repozitorijuma

```
TStoreRepository.java
1 package rs.inforce.dataapp.repositories;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4 import rs.inforce.dataapp.domain.TStore;
5
6 public interface TStoreRepository extends JpaRepository<TStore, Integer> {
7 }
8
```

```
TStore.java
5 @Entity
6 @Table(name="t_store")
7 public class TStore {
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.AUTO)
11    private Integer id;
12
13    @Column(nullable = false)
14    private String name;
15
16    public TStore() {
17    }
18
19    public Integer getId() { return id; }
22
23    public void setId(Integer id) { this.id = id; }
26
27    public String getName() { return name; }
30
31    public void setName(String name) { this.name = name; }
34 }
```



# REST kontroler klasa *service* interfejs i klasa

```
TStoreController.java
1 package rs.inforce.dataapp.controllers;
2
3 import org.springframework.web.bind.annotation.RequestMapping;
4 import org.springframework.web.bind.annotation.RestController;
5 import rs.inforce.dataapp.domain.TStore;
6 import rs.inforce.dataapp.services.TStoreService;
7
8 @RestController
9 public class TStoreController {
10
11     private TStoreService tStoreService;
12
13     public TStoreController(TStoreService tStoreService) {
14         this.tStoreService = tStoreService;
15     }
16
17     @RequestMapping("/stores")
18     public Iterable<TStore> findAll() {
19         return this.tStoreService.findAll();
20     }
21 }
```

```
TStoreServiceImpl.java
1 package rs.inforce.dataapp.services;
2
3 import org.springframework.stereotype.Service;
4 import rs.inforce.dataapp.domain.TStore;
5 import rs.inforce.dataapp.repositories.TStoreRepository;
6
7 @Service
8 public class TStoreServiceImpl implements TStoreService {
9
10     private TStoreRepository tStoreRepository;
11
12     public TStoreServiceImpl(TStoreRepository tStoreRepository) {
13         this.tStoreRepository = tStoreRepository;
14     }
15
16     @Override
17     public Iterable<TStore> findAll() {
18         return this.tStoreRepository.findAll();
19     }
20 }
21
```

# Interfejs PagingAndSortingRepository

- Obezbeđuje straničenje (engl. *paging*) objekata, što je neophodno kada upit vraća veliku količinu podataka
- Obezbeđuje sortiranje objekata po zadatom kriterijumu sortiranja

## Method Summary

All Methods

Instance Methods

Abstract Methods

Modifier and Type

Method and Description

Page<T>

`findAll(Pageable pageable)`

Returns a Page of entities meeting the paging restriction provided in the Pageable object.

Iterable<T>

`findAll(Sort sort)`

Returns all entities sorted by the given options.

Methods inherited from interface `org.springframework.data.repository.CrudRepository`

`count, delete, deleteAll, deleteAll, deleteById, existsById, findAll, findAllById, findById, save, saveAll`

# Primer sortiranja i straničenja

```
@GetMapping("/storesPagedOrdered")  
public Page<TStore> getPagedOrderedStores(Pageable pageable) {  
    return this.tStoreService.getPagedOrderedStores(pageable);  
}
```

```
public Page<TStore> getPagedOrderedStores(Pageable pageable) {  
    return this.tStoreRepository.findAll(pageable);  
}
```

- Prva stranica dužine 5, sortirano po nazivu u rastućem redosledu  
.../storesPagedOrdered?page=0&size=5&sort=name
- Druga stranica dužine 3, sortirano po nazivu u rastućem redosledu, pa po id u opadajućem redosledu  
.../storesPagedOrdered?page=1&size=3&sort=name,asc&sort=id,desc

# Deklarisanje metoda upita

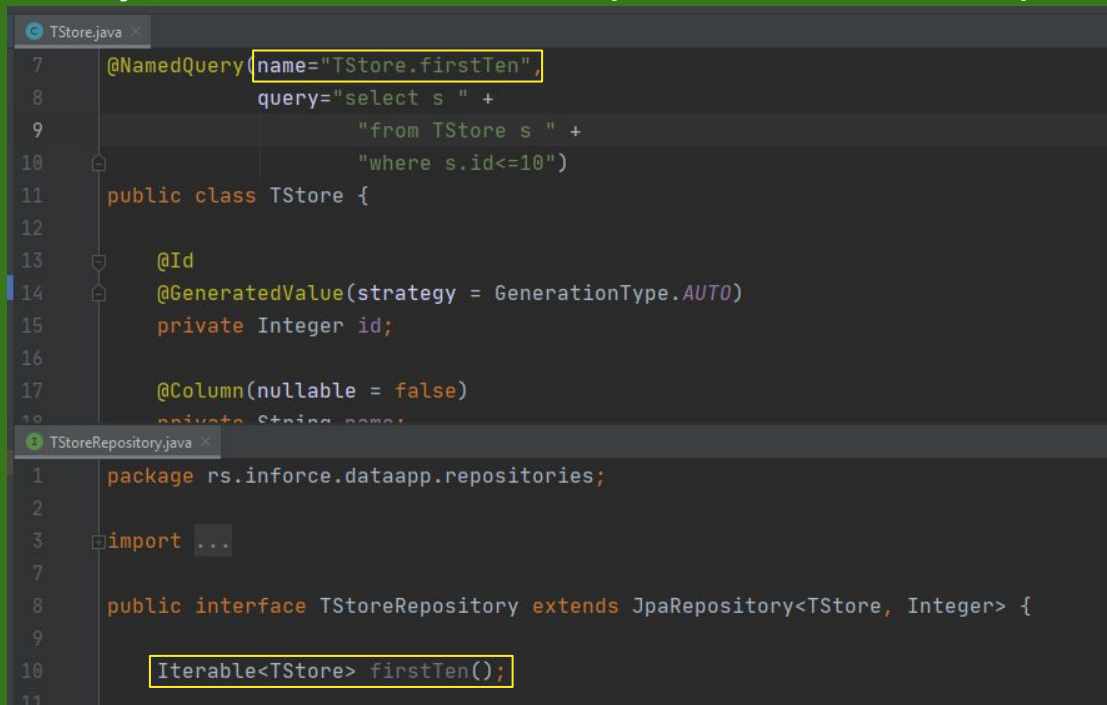
---

- Na osnovu naziva i parametara metode Spring izvršava odgovarajući upit

```
public interface TStoreRepository extends JpaRepository<TStore, Integer> {  
    Iterable<TStore> findByName(String name);  
    Iterable<TStore> findIdLessThan(Integer id);  
    Iterable<TStore> findByNameLike(String likeCriteria);  
    Iterable<TStore> findByNameContaining(String n);  
    Page<TStore> findByNameContainingIgnoreCase(String n, Pageable p);  
}
```

# Imenovani JPQL upiti

- Upit u JPA *entity* klasi može se izvršiti pozivom metode repozitorijuma



The screenshot shows two Java files in an IDE. The top file, `TStore.java`, contains a JPQL query named `TStore.firstTen` and the `TStore` entity class. The bottom file, `TStoreRepository.java`, contains the `TStoreRepository` interface. Both the query name and the repository method name are highlighted with yellow boxes.

```
TStore.java
7  @NamedQuery(name="TStore.firstTen",
8      query="select s " +
9          "from TStore s " +
10         "where s.id<=10")
11  public class TStore {
12
13      @Id
14      @GeneratedValue(strategy = GenerationType.AUTO)
15      private Integer id;
16
17      @Column(nullable = false)
18      private String name;
19  }

TStoreRepository.java
1  package rs.inforce.dataapp.repositories;
2
3  import ...
4
5
6
7
8  public interface TStoreRepository extends JpaRepository<TStore, Integer> {
9
10     Iterable<TStore> firstTen();
11 }
```

# JPQL upiti definisani u repozitorijumu

- JPQL upit moguće je definisati u samom repozitorijumu

```
public interface TStoreRepository extends JpaRepository<TStore, Integer> {  
  
    @Query("select s " +  
           "from TStore s " +  
           "where s.id<6 or s.id=?1")  
    Iterable<TStore> firstFiveOrId(Integer id);  
  
    @Query("select s " +  
           "from TStore s " +  
           "where s.name like ?1%")  
    Iterable<TStore> findByNameStartsWith(String criteria);  
}
```

# JPQL upiti sa imenovanim parametrima

---

- JPQL upiti mogu imati imenovane parametre, pri čemu poredak navođenja parametara u upitu i prosleđivanja parametara u metodi nije bitan

```
public interface TStoreRepository extends JpaRepository<TStore, Integer> {  
  
    @Query("select s " +  
           "from TStore s " +  
           "where s.name=:name or s.id=:id")  
    Iterable<TStore> namedParams(@Param("id") Integer id, @Param("name") String name);  
}
```

# *Service* klase u Spring aplikaciji

---

- Označavaju se anotacijom `@Service`
- Namenjena da sadrži poslovnu logiku
- Omogućava da REST kontroler ne bude kompleksan i da servisu delegira izvršenje poslovne logike ili validaciju
- Može da obezbedi upravljanje transakcijom
  - Prihvatanje objekta koji sadrži podatke iz više JPA klasa (Data Transfer Object - DTO)
  - Izvršavanje operacija nad bazom podataka preko repozitorijuma
- Po pravilu implementira interfejs koji se injektuje i može biti zamenjena moc klasom u fazi testiranja



# Rezime

---

- Spring Data JPA projekat omogućava efikasan razvoj dela aplikacije koji je zadužen za interakciju sa bazom podataka
- *Repository* interfejsi obezbeđuju izvršavanje CRUD operacija kao i definisanje i izvršavanje kompleksnih upita
- *Service* klase oslobađaju REST kontrolere od implementacije poslovne logike i omogućavaju upravljanje transakcijama