Spring Data

- abstrakcja dostępu do danych, max redukcja boiler plate
- implementacje dla różnych baz danych

jpa | mongo | redis | ldap | solr | rest

- community

elastic search | neo4j | couchdb | dynamo

Spring Data - Repository

```
@NoRepositoryBean
public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID> {
  <S extends T> S save(S id);

  <S extends T> Iterable<S> save(Iterable<S> ids);

  T findOne(ID id);

  boolean exists(ID id);

  Iterable<T> findAll();

  Iterable<T> findAll(Iterable<ID>id);

  long count();

  void delete(ID id);

  void delete(T entity);

  void delete(Iterable<? extends T>entities);

  void deleteAll();
}
```

Spring Data - Repository

```
@NoRepositoryBean
public interface PagingAndSortingRepository<T, ID extends Serializable> extends CrudRepository<T, ID> {
   Iterable<T> findAll(Sort sort);

   Page<T> findAll(Pageable pagable);
}
```

Spring Data - własne repozytorium bazowe

```
@NoRepositoryBean
public interface MyBaseRepo<T, ID extends Serializable> extends Repository<T, ID> {

        <S extends T> S save(S var1);

        T findOne(ID var1);

        default T get(ID id) throws NotFoundException {...}

}
```

Spring Data - definiowanie kwerend

- strategie
    - CREATE
        - find...By, read...By, count...By, query...By, get...By
    - USE_DECLARED_QUERY
        - @Query, @NamedQuery lub inne w zależności od impl
    - CREATE_IF_NOT_FOUND
- @EnableJpaRepositories(...)

Spring Data - przykłady kwerend

```
public interface UserRepository extends Repository<User, Long> {

List<Person> findByEmailAddressAndLastname(EmailAddress emailAddress, String lastname);

List<Person> findDistinctPeopleByLastnameOrFirstname(String lastname, String firstname);
List<Person> findPeopleDistinctByLastnameOrFirstname(String lastname, String firstname);

// Enabling ignoring case for an individual property
List<Person> findByLastnameIgnoreCase(String lastname);
// Enabling ignoring case for all suitable properties
List<Person> findByLastnameAndFirstnameAllIgnoreCase(String lastname, String firstname);

// Enabling static ORDER BY for a query
List<Person> findByLastnameOrderByFirstnameAsc(String lastname);
List<Person> findByLastnameOrderByFirstnameDesc(String lastname);

@Query(“SELECT p FROM Person p WHERE p.some = :some”)
List<Person> findBySomeCustomQuery(@Param(“some”) String some);

}
```

Spring Data - przykłady kwerend

```
public interface UserRepository extends Repository<User, Long> {

User findFirstByOrderByLastnameAsc();

User findTopByOrderByAgeDesc();

Page<User> queryFirst10ByLastname(String lastname, Pageable pageable);

Slice<User> findTop3ByLastname(String lastname, Pageable pageable);

List<User> findFirst10ByLastname(String lastname, Sort sort);

List<User> findTop10ByLastname(String lastname, Pageable pageable);

}
```

Spring Data - custom functionality

```
public interface UserRepositoryCustomFunc {

        void comeCustomMethod();

}

public interface UserRepository extends CrudRepository<User, Long>, UserRepositoryCustomFunc {
        ...
}

public class UserRepositoryCustomFuncImpl implements UserRepositoryCustomFunc  {

        public void comeCustomMethod() { … }

}
```

Spring Data - custom functionality

```java
public interface CustomRepository<T, ID extends Serializable> extends Repository<T, ID> {

   void comeCustomMethod1();
   void comeCustomMethod2();

}

public class CustomRepositoryImpl<T, ID extends Serializable>extends SimpleJpaRepository<T, ID> implements CustomRepository<T, ID>  {

   private final EntityManager entityManager;

   public MyRepositoryImpl(JpaEntityInformation entityInformation, EntityManager entityManager) {
      super(entityInformation, entityManager);
      this.entityManager = entityManager;
   }

   @Override
   public void comeCustomMethod1() {...}

   @Override
   public void comeCustomMethod2() {...}

}
```

```java
@EnableJpaRepositories(repositoryBaseClass = CustomRepositoryImpl.class)
```

Spring Boot - profile

```
@Configuration
@Profile("production")
public class ProductionConfiguration {

   @Bean
   SmsSender smsSender() {return new RealSmsSender();}

}

@Configuration
@Profile("development")
public class DevelopmentConfiguration {

   @Bean
   SmsSender smsSender() {return new FakeSmsSender();}

}
```