Spring erzeugt Standard-"Repositories" (DAOs)

- Bei Spring heißen DAOs Repositories
- Ein Interface pro Entity-Klasse wird von JpaRepository abgeleitet
  - Interface JpaRepository<T,ID extends Serializable>
  - T ist der Entity-Typ
  - ID ist der Typ des Schlüssel-Attributs
- Automatisch werden 18 CRUD Methoden durch Spring generiert
- Das Interface kann um eigene Standardmethoden und spezielle Queries erweitert werden
  - z.B. find Methoden für jedes Attribut

```
public interface StudentDao extends JpaRepository<Student, Long> {
    List<Student> findByFirstName(String firstName);
    List<Student> findByLastName(String lastName);
}
```

Standard-Methoden von JpaRepository

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
void	`deleteAllInBatch()` Deletes all entites in a batch call.
void	`deleteInBatch(Iterable<T> entities)` Deletes the given entities in a batch which means it will create a single `Query`.
`List<T>`	`findAll()`
`List<T>`	`findAll(Iterable<ID> ids)`
`List<T>`	`findAll(Sort sort)`
void	`flush()` Flushes all pending changes to the database.
T	`getOne(ID id)` Returns a reference to the entity with the given identifier.
`<S extends T>` `List<S>`	`save(Iterable<S> entities)`
`<S extends T>` S	`saveAndFlush(S entity)` Saves an entity and flushes changes instantly.

Methods inherited from interface org.springframework.data.repository.PagingAndSortingRepository

findAll

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

count, delete, delete, delete, deleteAll, exists, findOne, save

Erweiterte Query-Methoden für JPA Repositories

- Die 18 Standard-Methoden können durch eigene Methoden erweitert werden
  - Es gibt mehrere Möglichkeiten
  - Unterschiedlich komplex
  - Unterschiedlich leistungsfähig
- Query wird durch den Methodennamen bestimmt

```
List<Student> findByFirstName(String firstName);
```

- Query wird in einer @Query-Annotation angegeben

```
@Query("select u from User u where u.firstname like %?1%")
List<User> findByFirstnameContains(String firstname);
```

- Query wird durch ein Suchkriterium bestimmt (Criteria Query)
  - Criteria Objekte enthalten spezielle Java-Ausdrücke, mit denen komplexe Suchkriterien formuliert werden können

Methodennamen Queries

- Methodennamen im Repository Interface werden beim Programmstart analysiert und in SQL übersetzt
- Damit können komplexe Abfragen mit einem oder mehreren Parametern aufgebaut werden

```
public interface UserRepository extends Repository<User, Long> {
  List<User> findByEmailAddressAndLastname
                              (String emailAddress, String lastname);
}
```

- Wird übersetzt zu

```
select u from User u where u.emailAddress = ?1 and u.lastname = ?2
```

- Queries werden aus einer Liste von Schlüsselworten zusammen mit Entitätsnamen und Attributnamen zusammengesetzt

SQL
↔
Query

Logical keyword (SQL)	Keyword expressions (Query Method)
AND	And
OR	Or
NOT	Not, IsNot
GREATER_THAN	GreaterThan, IsGreaterThan
GREATER_THAN_EQUALS	GreaterThanEqual, IsGreaterThanEqual
LESS_THAN	LessThan, IsLessThan
LESS_THAN_EQUAL	LessThanEqual, IsLessThanEqual
TRUE	True, IsTrue
FALSE	False, IsFalse
IS	Is, Equals, (or no keyword)
LIKE	Like, IsLike
NOT_LIKE	NotLike, IsNotLike
IS_NULL	Null, IsNull
IS_NOT_NULL	NotNull, IsNotNull
EXISTS	Exists

Keyword	Sample	JPQL snippet
And	findByLastnameAndFirstname	… where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	… where x.lastname = ?1 or x.firstname = ?2
Not	findByLastnameNot	… where x.lastname <> ?1
GreaterThan	findByAgeGreaterThan	… where x.age > ?1
GreaterThanEqual	findByAgeGreaterThanEqual	… where x.age >= ?1
LessThan	findByAgeLessThan	… where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	… where x.age <= ?1
True	findByActiveTrue()	… where x.active = true
False	findByActiveFalse()	… where x.active = false
Is,Equals	FindByFirstname, findByFirstnameIs, findByFirstnameEquals	… where x.firstname = ?1

Keyword	Sample	JPQL snippet
`Like`	`findByFirstnameLike`	`… where x.firstname like ?1`
`NotLike`	`findByFirstnameNotLike`	`… where x.firstname not like ?1`
`IsNull`	`findByAgeIsNull`	`… where x.age is null`
`IsNotNull, NotNull`	`findByAge(Is)NotNull`	`… where x.age not null`

SQL
↔
Query

Logical keyword (SQL)	Keyword expressions (Query Method)
STARTING_WITH	StartingWith, IsStartingWith, StartsWith
ENDING_WITH	EndingWith, IsEndingWith, EndsWith
AFTER	After, IsAfter
BEFORE	Before, IsBefore
BETWEEN	Between, IsBetween
IN	In, IsIn
NOT_IN	NotIn, IsNotIn
NEAR	Near, IsNear
CONTAINING	Containing, IsContaining, Contains
WITHIN	Within, IsWithin
REGEX	Regex, MatchesRegex, Matches

Keyword	Sample	JPQL snippet
StartingWith	findByFirstnameStartingWith	… where x.firstname like ?1 (parameter bound with appended %)
EndingWith	findByFirstnameEndingWith	… where x.firstname like ?1 (parameter bound with prepended %)
After	findByStartDateAfter	… where x.startDate > ?1
Before	findByStartDateBefore	… where x.startDate < ?1
Between	findByStartDateBetween	… where x.startDate between 1? and ?2
In	findByAgeIn(Collection<Age> ages)	… where x.age in ?1
NotIn	FindByAgeNotIn(Collection<Age> age)	… where x.age not in ?1
Containing	findByFirstnameContaining	… where x.firstname like ?1 (parameter bound wrapped in %)

Keyword	Sample	JPQL snippet
`OrderBy`	`findByAgeOrderByLastnameDesc`	`… where x.age = ?1 order by x.lastname desc`
`IgnoreCase`	`findByFirstnameIgnoreCase`	`… where UPPER(x.firstame) = UPPER(?1)`