Skillbox

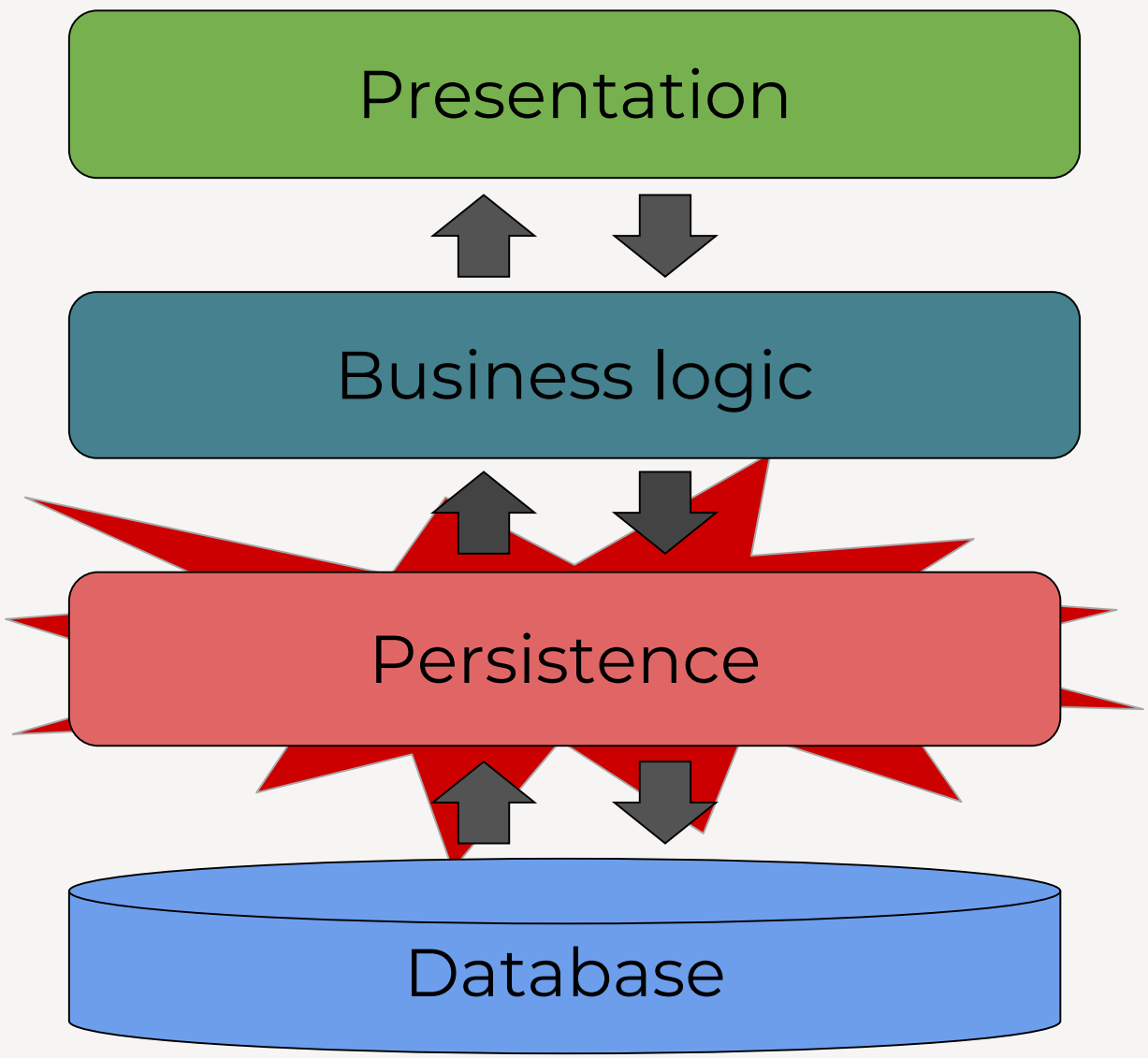
Spring Data JPA

Кошаев К. А.

Java-разработчик

Persistence layer

Вся логика приложения, которая использует интерфейсы JPA и Hibernate, формирует persistence layer приложения



DAO

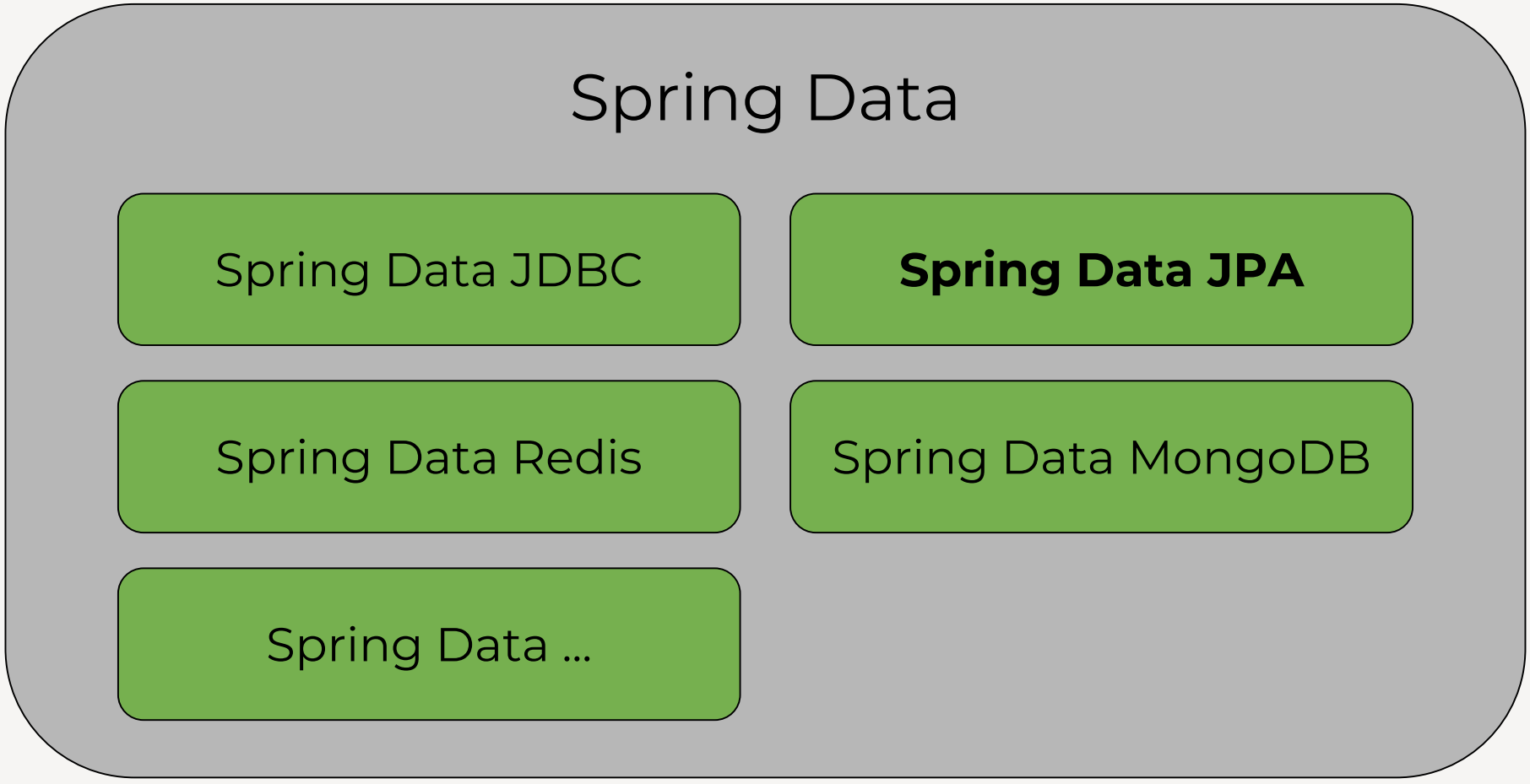
Data Access Object (DAO) — это структурный шаблон, который позволяет нам изолировать уровень бизнес логики от persistence-логики с помощью абстрактного API

```
public interface Dao<T> {

    T get(long id);

    List<T> getAll();

    void save(T t);

    void update(T t, String[] params);

    void delete(T t);
}
```

Skillbox

Spring Data JPA

Spring Data JPA является частью семейства проектов Spring Data и упрощает создание репозиториев на основе JPA. Этот модуль имеет дело с расширенной поддержкой уровней доступа к данным на основе JPA



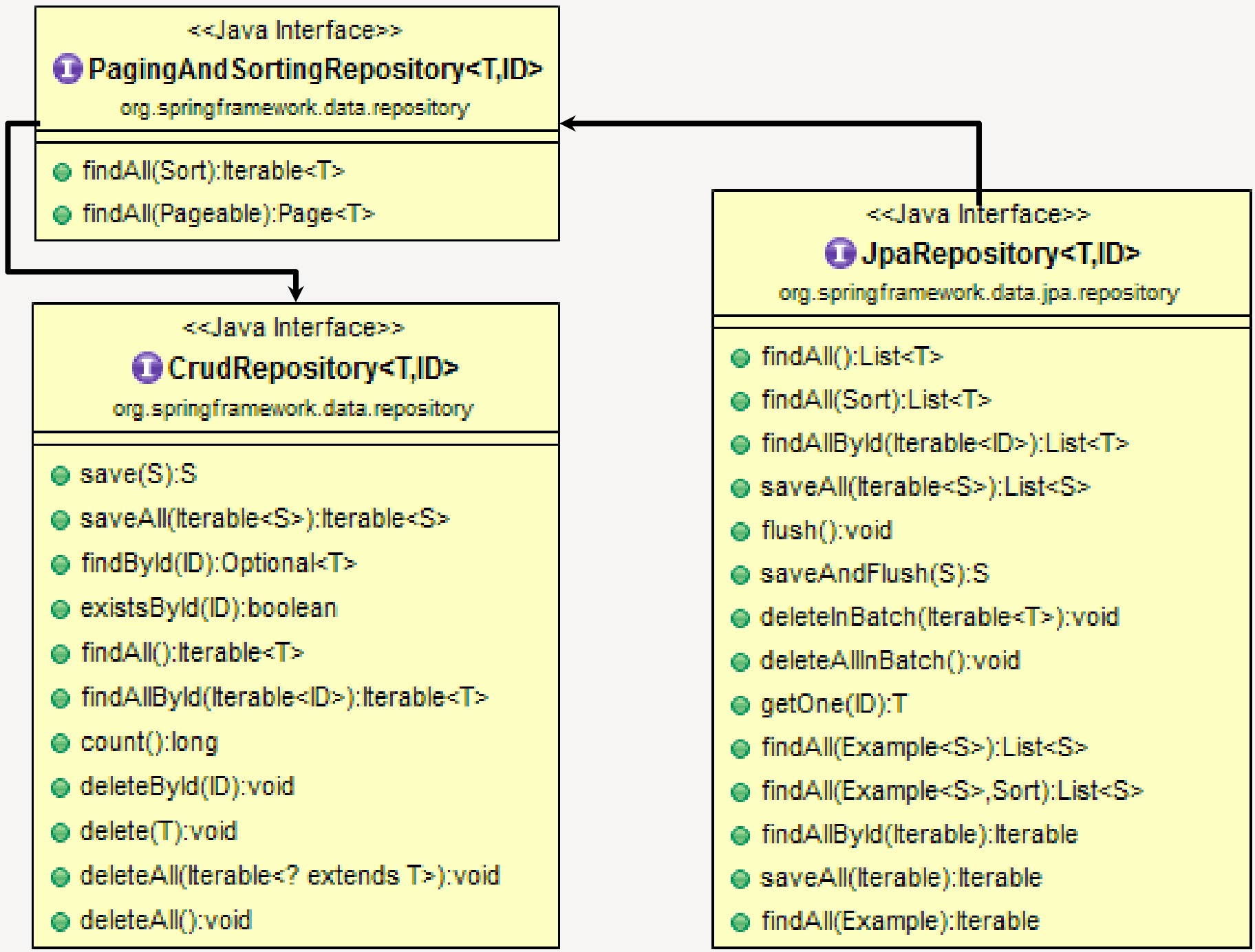
Skillbox

Repository

Шаблон репозиторий — один из самых популярных шаблонов, связанных с темой persistence. Он инкапсулирует конкретные детали реализации хранилища данных и позволяет создавать бизнес-код на ещё более высоком уровне абстракции

```
public interface UserRepository {
    User get(Long id);
    void add(User user);
    void update(User user);
    void remove(User user);
}
```

Repository


<<Java Interface>>
PagingAndSortingRepository<T,ID>
org.springframework.data.repository
findAll(Sort):Iterable<T>
findAll(Pageable):Page<T>
<<Java Interface>>
CrudRepository<T,ID>
org.springframework.data.repository
save(S):S
saveAll(Iterable<S>):Iterable<S>
findById(ID):Optional<T>
existsById(ID):boolean
findAll():Iterable<T>
findAllById(Iterable<ID>):Iterable<T>
count():long
deleteById(ID):void
delete(T):void
deleteAll(Iterable<? extends T>):void
deleteAll():void
<<Java Interface>>
JpaRepository<T,ID>
org.springframework.data.jpa.repository
findAll():List<T>
findAll(Sort):List<T>
findAllById(Iterable<ID>):List<T>
saveAll(Iterable<S>):List<S>
flush():void
saveAndFlush(S):S
deleteInBatch(Iterable<T>):void
deleteAllInBatch():void
getOne(ID):T
findAll(Example<S>):List<S>
findAll(Example<S>,Sort):List<S>
findAllById(Iterable):Iterable
saveAll(Iterable):Iterable
findAll(Example):Iterable

Методы-запросы

Ещё одна удобная функция Spring Data JPA — создание запросов к базе данных на основе имён методов. За кулисами Spring генерирует запрос JPQL — сильно вдохновлённое подмножество HQL на основе имени метода, устанавливает предоставленные параметры метода выполняет binding, выполняет запрос и возвращает результат

```
List<User> findByAgeLessThan(int age);
List<User> findByAgeLessThanEqual(int age);
List<User> findByAgeGreaterThan(int age);
List<User> findByAgeGreaterThanEqual(int age);
...
List<User> findByStartDateAfter(Date date);
List<User> findByFirstnameLike(String firstname);
List<User> findByFirstnameNotLike(String firstname);
```

Методы-запросы

- Имя метода запроса должно начинаться с одного из следующих префиксов: find… By, read… By, query… By, count… By и get… By
- Если необходимо получить более одного результата, нужно добавить необязательное числовое значение к ключевым словам First и Top
- Если необходимо выбрать уникальные результаты, нужно добавить ключевое слово Distinct перед первым By
- Если метод запроса определяет *x* условий поиска , необходимо добавить *x* параметров метода

Методы-запросы
Недостатки

- Возможности парсера имён методов определяют, какие запросы можно создавать
- Если синтаксический анализатор имени метода не поддерживает требуемое ключевое слово, то запрос составить не удастся
- Имена сложных методов запросов выглядят громоздко и некрасиво
- Нет поддержки динамических запросов

@Query @NamedQuery

Использование именованных запросов для объявления запросов сущностей является допустимым подходом и отлично работает для небольшого числа запросов

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

  @Query("select u from User u where u.emailAddress = ?1")
  User findByEmailAddress(String emailAddress);
}
```



```
@Entity
@Table(name = "users")
@NamedQuery(name = "User.findByEmailAddress", query =
"select u from User u where u.emailAddress = ?1")
public class User {
        ...
}
public interface UserRepository extends JpaRepository<User, Long> {

   User findByEmailAddress(String emailAddress);
}
```

Skillbox

Spring Boot & Spring Data JPA

Для того, чтобы использовать этот инструмент в Spring Boot приложении достаточно только добавить артефакт spring-boot-starter-data-jpa, который включает все необходимые зависимости и активирует конфигурацию по умолчанию

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

Skillbox

Спасибо за внимание!