扩展 Spring Data JPA

在编写 Spring Data JPA 的 DAO 时，只需在接口中按规约提供方法的声明即可。而有些业务实现无法通过声明方法或编写简单 SQL 实现，这就需要扩展 Spring Data JPA。

本文主要探讨 2 个问题：

1. 如何为某一个特定 Repository 添加自定义方法。
2. 如何为所有的 Repository 添加自定义的方法。

问题 1：为某一个特定 Repository 添加自定义方法。具体步骤：

1) 定义一个接口: 声明要添加的方法

2) 提供该接口的实现类: 类名需符合 **EntityNameRepositoryImpl** 格式, 并提供方法的实现

3) 声明一个 Repository 接口, 并继承 1) 声明的接口

4) 使用

5) 注意：默认情况下, Spring Data 会在 base-package 中查找 "接口名 Impl" 作为实现类. 也可以通过 repository-impl-postfix 声明后缀

示例代码：

1) 定义一个接口: 声明要添加的方法

```
package com.atguigu.jpa.repository;

public interface EmployeeDao {
    void method();
}
```

2) 提供该接口的实现类: 类名需符合 **EntityNameRepositoryImpl** 格式, 并提供方法的实现

```
package com.atguigu.jpa.repository;

import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

public class EmployeeRepositoryImpl implements EmployeeDao{

    //获取当前线程的 EntityManager 实例
    @PersistenceContext
    private EntityManager entityManager;
```

```
    @Override
    public void method() {
        System.out.println("method..." + entityManager);
    }
}
```

3) 声明一个 Repository 接口，并继承 1) 声明的接口

```
package com.atguigu.jpa.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import com.atguigu.jpa.beans.Employee;

public interface EmployeeRepository
        extends JpaRepository<Employee, Integer>, EmployeeDao{
}
```

4) 使用

```
EmployeeRepository employeeRepository =
        ctx.getBean(EmployeeRepository.class);
employeeRepository.method();
```

问题 2：为所有的 Repository 添加自定义的方法。具体步骤：

1) 声明一个接口，在该接口中声明需要自定义的方法，该接口需要继承 Spring Data 的 Repository 接口或其子接口.
2) 提供 1) 所声明的接口的实现类. 且继承 SimpleJpaRepository, 并提供方法的实现。注意: 全局的扩展实现类不要用 RepositoryImp 作为后缀名, 或为全局扩展接口添加 @NoRepositoryBean 注解告知 Spring Data: 该实现类不是一个 Repository
3) 定义 RepositoryFactoryBean 的实现类, 使其生成 1) 定义的接口实现类的对象
4) 修改 <jpa:repositories /> 节点的 factory-class 属性指向 3) 的全类名
5) 使用

示例代码：

1) 声明一个接口，在该接口中声明需要自定义的方法，该接口需要继承 Spring Data 的 Repository 接口或其子接口.

```
package com.atguigu.jpa.repository;

import java.io.Serializable;
```

```
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.NoRepositoryBean;

@NoRepositoryBean
public interface BaseDao<T, ID extends Serializable>
    extends JpaRepository<T, ID>{

    void method2();

}
```

2) 提供 1) 所声明的接口的实现类. 且继承 SimpleJpaRepository, 并提供方法的实现。注意: 全局的扩展实现类不要用 RepositoryImp 作为后缀名, 或为全局扩展接口添加 @NoRepositoryBean 注解告知 Spring Data: 该实现类不是一个 Repository

```
package com.atguigu.jpa.repository;

import java.io.Serializable;
import javax.persistence.EntityManager;
import
        org.springframework.data.jpa.repository.support.SimpleJpaRepository;

public class BaseDaoImpl<T, ID extends Serializable>
        extends SimpleJpaRepository<T, ID> implements BaseDao<T, ID> {

    private EntityManager entityManager;

    public BaseDaoImpl(Class<T> domainClass, EntityManager em) {
        super(domainClass, em);
        this.entityManager = em;
    }

    @Override
    public void method2() {
        System.out.println(">>>method2..." + entityManager);
    }

}
```

3) 定义 RepositoryFactoryBean 的实现类, 使其生成 1) 定义的接口实现类

的对象

```
package com.atguigu.jpa.repository;

import java.io.Serializable;

import javax.persistence.EntityManager;

import org.springframework.data.jpa.repository.JpaRepository;
import
 org.springframework.data.jpa.repository.support.JpaRepositoryFactory;
import
 org.springframework.data.jpa.repository.support.JpaRepositoryFactoryBean;
import org.springframework.data.repository.core.RepositoryMetadata;
import
org.springframework.data.repository.core.support.RepositoryFactorySupport;

public class BaseDaoRepositoryFactoryBean<R extends JpaRepository<S, ID>,
    S, ID extends Serializable>
    extends JpaRepositoryFactoryBean<R, S, ID> {

    @Override
    protected RepositoryFactorySupport createRepositoryFactory(
            EntityManager entityManager) {
        return new BaseDaoFactory(entityManager);
    }

    private static class BaseDaoFactory<S, ID extends Serializable>
            extends JpaRepositoryFactory{

        public BaseDaoFactory(EntityManager entityManager) {
            super(entityManager);
        }

        @Override
        protected <T, ID extends Serializable> JpaRepository<?, ?>
        getTargetRepository(RepositoryMetadata metadata,
                EntityManager entityManager) {
            return new
                    BaseDaoImpl<>(metadata.getDomainType(), entityManager);
        }
```

```
        @Override
        protected Class<?> getRepositoryBaseClass(
                        RepositoryMetadata metadata) {
            return BaseDao.class;
        }
    }

}
```

4) 修改 <jpa:repositories /> 节点的 factory-class 属性指向 3) 的全类名

```
<jpa:repositories base-package="com.atguigu.jpa.repository"
    entity-manager-factory-ref="entityManagerFactory"
    transaction-manager-ref="transactionManager"
    factory-class="com.atguigu.jpa.repository.BaseDaoRepositoryFactoryBean"/>
```

5) 使用

```
DepartmentRepository departmentRepository =
        (DepartmentRepository) ctx.getBean("departmentRepository");
departmentRepository.method2();
```