第十四节 Spring 数据存储与校验

知识回顾

JDBC

核心 API

数据源 - javax.sql.DataSource

生成 SQL 连接对象（会话管理器）java.sql.Connection

SQL 连接对象（会话管理器）- java.sql.Connection

- 有状态 - 缓存部分查询等结果
- 无状态 - 每次都是和数据直接交互

SQL 命令执行器（会话）- java.sql.Statement

- 普通 - java.sql.Statement
  - DML - executeUpdate 或者 executeBatch

    - DDL - execute
- 预编译 - java.sql.PreparedStatement
- 存储过程 - java.sql.CallableStatement

SQL 查询结果集 - java.sql.ResultSet

getXXX() 方法，参数：通过 column index 或 name

SQL 异常 - java.sql.SQLException

JPA

核心 API

实体管理器（会话）工厂 - javax.persistence.EntityManagerFactory

实体管理器（会话） - javax.persistence.EntityManager

依赖注入注解 - javax.persistence.PersistenceUnit

```
@PersistenceUnit
private EntityManagerFactory
entityManagerFactory;
```

实体生命周期

- javax.persistence.PostLoad
- javax.persistence.PostPersist

Bean Validation

核心 API

校验器工厂 - javax.validation.ValidatorFactory

校验器 - javax.validation.Validator

校验结果 - javax.validation.ConstraintViolation

Spring JDBC

核心 API

主门面接口 - org.springframework.jdbc.core.JdbcTemplate

并非模板模式的实现，而是门面模式的实现，也实现“命令”模式。

设计原则

- 面向数据表行列和面向对象混合编程（手动）
- 通过 Callback 接口无需显示地捕获 SQLException，而是 DataAccessException
- 通过操作方法屏蔽 DDL 和 DML 在 JDBC API 中的差异
- 减少一些重复操作，如关闭 Connection、关闭 Statement 以及关闭 ResultSet

Spring ORM

Spring ORM - Hibernate

org.springframework.orm.hibernate5.LocalSessionFactoryBean

= Local(Spring 容器创建) + SessionFactory + Bean(FactoryBean)

- 实现 Hibernate SessionFactory FactoryBean
- 实现 Spring IoC Aware 回调
  - ResourceLoaderAware - 注入 ResourceLoader（读取 ClassLoader 资源）
  - BeanFactoryAware - 注入 BeanFactory（依赖查找）
- 实现 Spring Bean 生命周期回调
  - InitializatingBean
  - DisposableBean
- 实现 HibernateExceptionTranslator - 将存储相关的 RuntimeException 转换为 Spring JDBC DataAccessException

属性语义

- dataSource - javax.sql.DataSource 对象
- configLocations - Hiberante XML 配置文件路径，依赖 ResourceLoader，如 classpath:/META-INF/hibernate-orm.xml
- mappingResources - Hibernate Mapping XML 配置文件

Spring ORM - JPA

org.springframework.orm.jpa.LocalEntityManagerFactoryBean

= Local(Spring 容器创建) + EntityManagerFactory + Bean(FactoryBean)

- 实现 JPA EntityManagerFactory FactoryBean
- 实现 Spring IoC Aware 回调
  - BeanClassLoaderAware - 注入 Bean 所用到 ClassLoader
  - BeanNameAware - 注入当前 Bean 名称
  - ResourceLoaderAware - 注入 ResourceLoader（读取 ClassLoader 资源）
  - BeanFactoryAware - 注入 BeanFactory（依赖查找）
- 实现 Spring Bean 生命周期回调
  - InitializatingBean
  - DisposableBean
- 实现 PersistenceExceptionTranslator - 将存储相关的 RuntimeException 转换为 Spring JDBC DataAccessException

Spring Data JPA

Spring Data Commons

Spring Data 允许开发人员定义数据仓储接口，可以继承 CrudRepository 等，并且只要符合 Spring Data 规约，就能自动地操作数据库，比如

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

    List<User> findByIdAndName(Long id,String
name);  // WHERE id = ? and name = ?

    List<User> findByIdAndNameAndPassword(Long
id,String name,String password);  // WHERE id =
? and name = ? and password = ?
}
```

Spring Data 利用自定义接口形成动态代理

核心 API

数据仓储标记接口 - org.springframework.data.repository.Repository<T,ID>

T 代表 Entity 类型

ID 代表 ID 主键类型

CRUD 数据仓储接口 - org.springframework.data.repository.CrudRepository

非数据仓储 Bean 注解 - NoRepositoryBean

用于区分 Repository Bean 接口和非 Repository Bean 接口

Spring Data JPA

Enable 模块驱动 - org.springframework.data.jpa.repository.config.EnableJpaRepositories

Spring Validator

核心 API

IoC 整合 Bean Validation - org.springframework.validation.beanvalidation.LocalValidatorFactoryBean

= Local(Spring 容器创建) + ValidatorFactory(Bean Validation) + Bean(FactoryBean)

- 扩展 org.springframework.validation.beanvalidation.SpringValidatorAdapter
- 实现 Spring IoC Aware 回调
  - ApplicationContextAware - 注入 ApplicationContext（BeanFactory）
- 实现 Spring Bean 生命周期回调
  - InitializatingBean
  - DisposableBean
- 实现 Bean Validation 标准接口 - ValidatorFactory

其他相关

契约编程

Hiberate shards

Spring FactoryBean

通过编程方式来确定 Bean 类型和 Bean 实例，而非 Bean 定义

- 比较复杂的 Bean 过程

```
<bean id="xxx" class="com.acme.XXX" ></bean>
```