1.Spring Data JPA

1.1 什么是Spring Data JPA

JPA即Java Persistence API.JPA是一个基于O/R映射的标准规范.所谓规范即只定义标准规则,不提供实现,软件提供商可以按照标准规范来实现,而使用者只需要按照规范中定义的方式来使用,而不用和软件打交道.

Spring Data JPA是Spring Data的一个子项目,它通过基于JPA的Repository极大地减少了JPA作为数据访问方案地代码量

1.2 定义数据访问层

使用Spring Data JPA建立数据库访问层十分简单,只需定义一个继承JpaRepository的接口即可.

```
public interface PersonRepository extends JpaRepository<Person, Long> {}
```

1.3 配置使用Spring Data JPA

在Spring环境中,使用Spring Data JPA可以通过@EnableJpaRepository注解来开启Spring Data JPA的支持,@EnableJpaRepository接收的value参数用来扫描数据访问层所在包下的数据访问的接口定义

还需要配置DataSource,PlatformTransactionManager等相关必须的bean

具体实现参见ch8_2

2.Spring Data REST

1.1 什么是Spring Data REST

Spring Data JPA是基于Spring Data的repository之上,可以将respository自动输出为REST资源.目前Spring Data REST支持将Spring Data JPA,Spring Data MongoDB,Spring Data Neo4j....的repository自动转换成REST服务.

1.2 Spring MVC中配置使用Spring Data REST

可以通过继承此类(RepositoryRestMvcConfiguration)或者@Import(RepositoryRestMvcConfiguration.class)

1.3 定制

(1)定制根路径

```
spring.data.rest.base-path=/api
```

(2)定制节点路径

默认规则就是在实体类之后加"s"来形成路径,

3.声明式事务

Spring支持声明式事务,使用@Transactional注解在方法上表明该方法需要事务支持:

```
@Bean
public PlatformTransactionManager transactionManager(){
   JpaTransactionManager transcationManager = new JpaTransactionManager();
   transactionManager.setDataSource(dataSource())
   return transactionManager;
}
```

Spring提供了一个@EnableTransactionManagement注解在配置类上开启声明式事务的支持.

3.1 Spring Boot的事务支持

(1) 自动配置的事务管理器

JDBC作为数据访问技术的时候,使用DataSourceTransactionManager(datasource)

JPA作为数据访问技术的时候,使用JpaTransactionManager();

(2) 自动开启注解的事务支持

在Spring Boot 中无须显示开启使用@EnableTransactionManagement注解

4 数据缓存Cache

1.Spring缓存支持

1.1 Spring支持的CacheManager

@Bean注入一个CacheManager

1.2 声明式缓存

Spring提供了4个注解来声明缓存规则,在配置类上@EnableCaching注解即可

注解	解释
@Cacheable	在方法执行前Spring先查询缓存中是否有数据,如果有数据,则直接返回缓存数据;若没有数据,调用方法并将方法返回值放进缓存
@CachePut	无论怎样.,都会将方法的返回值放到缓存,@CachePut的属性和@Cacheable保持一致
@CacheEvict	将一条或多条数据从缓存中删除
@Caching	可以通过@Caching注解组合多个注解策略在一个方法上

参见ch8_5

5 非关系型数据库NoSql

5.1 mongoDB

参见ch8_5

5.1.1 Spring的支持

Spring对MongoDB的支持主要通过Spring Data MongoDB来实现的,提供如下功能:

(1) Object/Document映射注解支持

Spring Data MongoDB提供了如下注解:

注解	解释
@Document	映射领域对象与MongoDB的一个文档
@Id	映射当前属性是ID
@DbRef	当前属性将参考其他的文档
@Field	为文档的属性定义名称
@Version	将当前属性作为版本

(2) MongoTemplate

需要一个MongoClient和MongoDbFactory来配置数据库连接属性

```
@Bean
public MongoClient client(){
   MongoClient client = new MongoClient(new ServerAddress("127.0.0.1",27017));
   return client;
}

@Bean
public MongoDbFactory mongoDbFactory(){
   String database = new
MongoClientURI("mongodb:/localhost/test").getDatabase();
   return new SimpleMongoDbFactory(client(),database);
}

@Bean
public MongoTemplate mongoTemplate(MongoFactory mongoDbFactory){
   return new MongoTemplate(mongoDbFactory);
}
```

(3)Repository的支持

类似于Spring Data JPA,Spring Data MongoDB也提供了Repository的支持,使用方式和Spring Data JPA一致

通过@EnableMongoRepositories开启

5.1.2 Spring Boot的支持

自动配置了@EnableMongoRepositories

支持方法名查询

@Query("{'age':?0}") // 支持@Query查询,查询参数构造JSON字符串即可.

5.2 Redis

参见ch8_6_redis

5.2.1 Spring的支持

(1) 配置

根据Redis不同的Java客户端,Spring Data Redis提供了如下的ConnectionFactory:

JedisConnectionFactory:使用Jedis作为Redis客户端

JredisConnectionFactory:使用Jredis作为Redis客户端

LettuceConnectionFactory:使用Lettuce作为Redis客户端

SrpConnectionFactory:使用Spullara/redis-protocol作为Redisk客户端

配置方式如下:

```
@Bean
public RedisConnectionFactory redisConnectionFactory(){
   return new JedisConnectionFactory();
}
```

RedisTemplate配置方式如下:

```
@Bean
public RedisTemplate<Object,Object> redisTemplate(){
   RedisTemplate<Object,Object> template = new RedisTemplate();
   template.setConnectionFactory(redisConnectionFactory());
   return template;
}
```