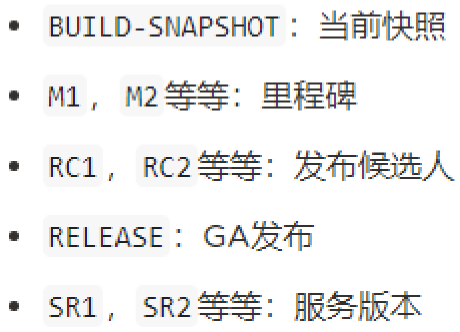
SpringBoot整合Spring Data JPA

一、简介

1.1 Spring Data Jpa是什么

Spring Data Jpa是Spring Data系列下的一个框架，操作数据库的，底层实现依赖于Hibernate Jpa

作用跟Mybatis、Hibernate一样都是操作数据库，通过注解实现数据库的操作



- `BUILD-SNAPSHOT`：当前快照
- `M1`，`M2`等等：里程碑
- `RC1`，`RC2`等等：发布候选人
- `RELEASE`：GA发布
- `SR1`，`SR2`等等：服务版本

1.2 优点

1.3 核心技术

1.3.1提供的四大接口

Repository 标记 是操作数据库的接口

CrudRepository 继承了Repository

提供常规的CRUD操作，增删改查

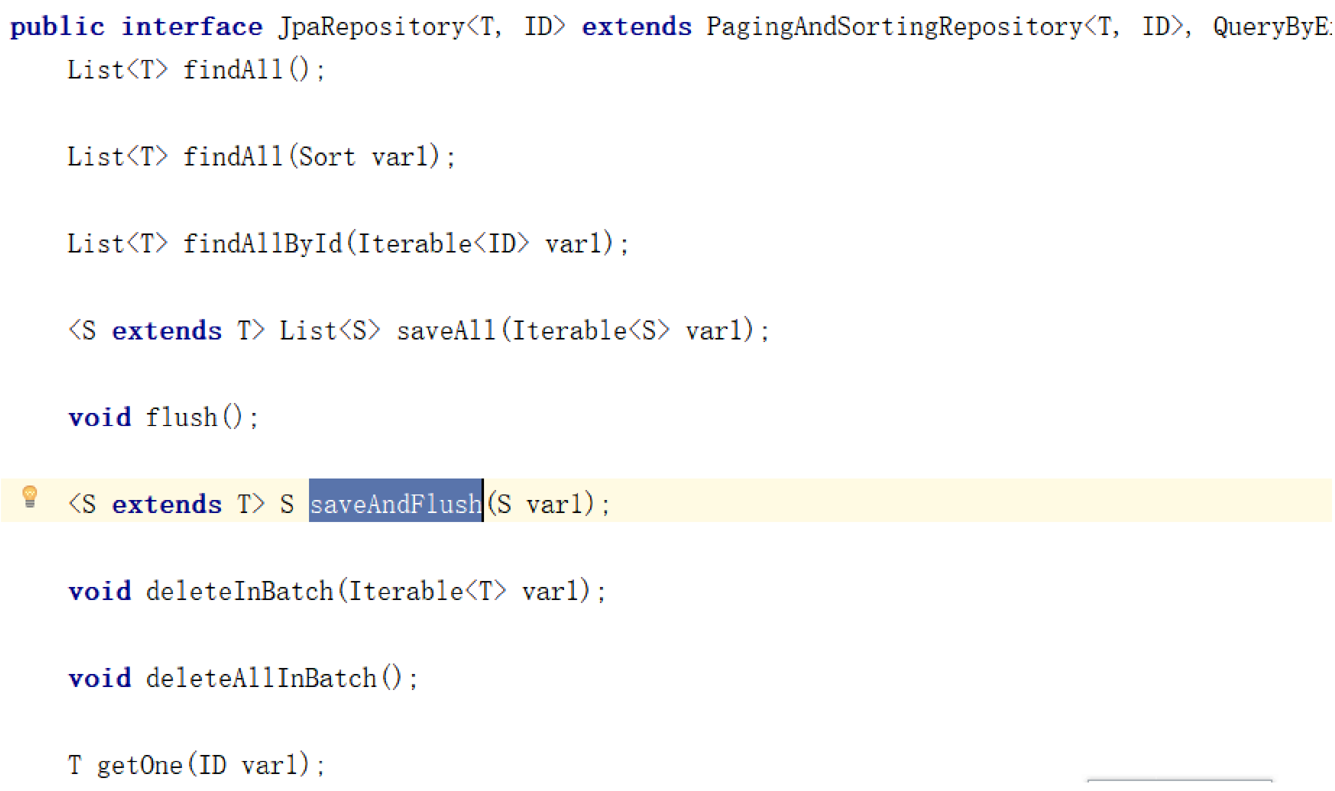
```
public interface CrudRepository<T, ID extends Serializable>
  extends Repository<T, ID> {

  <S extends T> S save(S entity);      ❶

  Optional<T> findById(ID primaryKey); ❷

  Iterable<T> findAll();               ❸

  long count();                        ❹

  void delete(T entity);               ❺

  boolean existsById(ID primaryKey);   ❻

  // … more functionality omitted.
}
```

PagingAndSortingRepository：继承了CrudRepository

扩展了分页和排序方法

```
public interface PagingAndSortingRepository<T, ID extends Serializable>
  extends CrudRepository<T, ID> {

  Iterable<T> findAll(Sort sort);

  Page<T> findAll(Pageable pageable);
}
```

JpaRepository：继承了PagingAndSortingRepository



```
public interface JpaRepository<T, ID> extends PagingAndSortingRepository<T, ID>, QueryByE
    List<T> findAll();

    List<T> findAll(Sort var1);

    List<T> findAllById(Iterable<ID> var1);

    <S extends T> List<S> saveAll(Iterable<S> var1);

    void flush();

    <S extends T> S saveAndFlush(S var1);

    void deleteInBatch(Iterable<T> var1);

    void deleteAllInBatch();

    T getOne(ID var1);
```

1.3.2 JPQL

HQL:Hibernate 面向对象查询语言

Sql: HQL:

表名 类名

字段 属性

数据 对象

Jpa 操作数据库 提供了一种面向对象的操作数据库的方式，语言

面向对象查询语言：JPQL 类似 HQL

Sql: JPQL:

表名 类名

字段 属性

数据 对象

1.3.3 方法名解析查询

定义方法名，Spring Data JPA可以根据方法名解析 自动生成SQL语句

二、Spring Boot 整合Spring Data Jpa

2.1初体验

1.依赖jar

``

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

2.配置

application.properties

设置：

数据库连接信息

Spring Data Jpa相关设置

``

```
#数据库连接设置
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/db_1901?
serverTimezone=UTC&characterEncoding=UTF8
spring.datasource.username=root
spring.datasource.password=root
#Spring Jpa 相关设置 设置数据库的方言
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL57Dialect
```

3、编写代码

实体层-entity

``

```
@Entity
@Table(name = "t_itproject")
@Data
public class ItProject {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)//主键的生成策略
    private Integer id;
    @Column(length = 30)
    private String name;
    private int persons;
    private int month;
    @Column(length = 50)
    private String info;
}
```

持久层-dao

``

```
public interface ItProjectDao extends JpaRepository<ItProject,Integer> {
}
```

业务逻辑层-service

``

```
public interface ItProjectService {
    R save(ItProject project);
    R queryAll();
    R del(int id);
}
```

业务逻辑层-实现层-service.impl

``

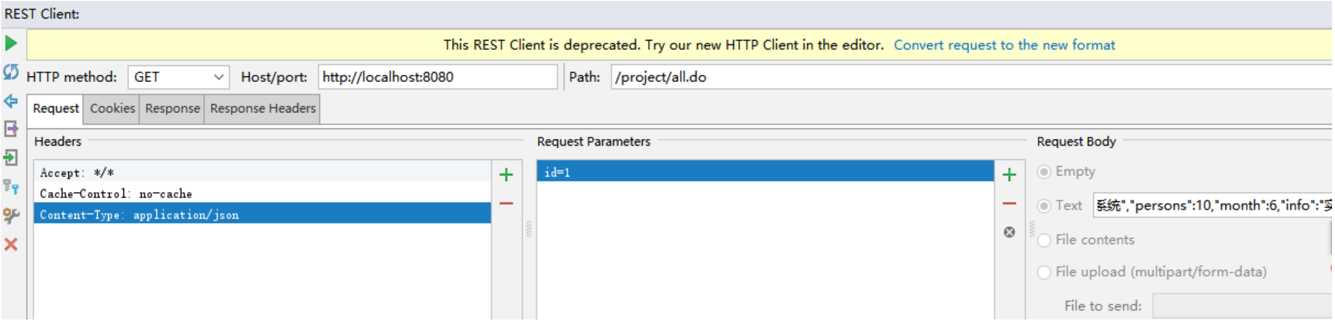
```
@Service
public class ItProjectServiceImpl implements ItProjectService {
    @Autowired
    private ItProjectDao projectDao;
    @Override
    public R save(ItProject project) {
        return R.exec(projectDao.save(project)!=null,null);

    }
    @Override
    public R queryAll() {
        return R.exec(true,projectDao.findAll());
    }

    @Override
    public R del(int id) {
        projectDao.deleteById(id);
        return R.exec(true,null);
    }
}
```

控制层-api

```
   @RestController
```

```java
public class ItProjectApi {
    @Autowired
    private ItProjectService projectService;
    @GetMapping("/project/all.do")
    public R all(){
        return projectService.queryAll();
    }
    @PostMapping("/project/save.do")
    public R save(@RequestBody ItProject project){
        return projectService.save(project);
    }
    @DeleteMapping("project/del.do")
    public R del(int id){
        return projectService.del(id);
    }
}
```

4、运行并测试



REST Client:

This REST Client is deprecated. Try our new HTTP Client in the editor. Convert request to the new format

HTTP method: GET Host/port: http://localhost:8080 Path: /project/all.do

Request | Cookies | Response | Response Headers

Headers

```
Accept: */*
Cache-Control: no-cache
Content-Type: application/json
```

Request Parameters

```
id=1
```

Request Body

○ Empty

● Text 系统","persons":10,"month":6,"info":"

○ File contents

○ File upload (multipart/form-data)

File to send:

2.2Spring Data Jpa操作数据库的方式

2.2.1 接口

提供了长度额四个接口，封装了常用方法

使用的时候，只需要创建接口 继承指定的接口

`public interface ItProjectDao extends JpaRepository<ItProject,Integer>`

2.2.2 JPQL

采用提供的JPQL(面向对象查询语言)语言操作数据库

Sql: JPQL:

表名 类名

字段 属性

数据 对象

提供注解：@Query

JPQL:只支持：Select Update Delete

默认的@Query 只能执行查询语句

如果想要执行修改和删除，那么必须在方法上加上注解：@Modifying

执行修改和删除的时候需要强制开启事务

SpringBoot项目开启事务：

1、开关类使用注解：@EnableTransactionManagement

2、业务逻辑实现层的方法上，使用注解：@Transactional

``

```
//基于JPQL语法 类就是表 对象就是数据 属性就是字段
@Modifying //标记 修改或删除语句
@Query(value = "update ItProject p set p.name=:name where p.id=:id")
void update(@Param("id") int id, @Param("name") String name);
@Query(value = "from ItProject ")
List<ItProject> all();
```

2.2.3 SQL

Spring Data JPA也支持SQL语句

使用的时候：使用注解@Query

需要设置注解的属性：nativeQuery=true

``

```
//基于SQL操作
@Query(value = "select * from t_itproject order by id desc",nativeQuery =true)
List<ItProject> sqlall();
```

2.2.4 命名查询

@NameQuery修饰的映射类

@NamedQueries 修饰映射类 可以有多个命名查询

命名查询的query属性对应的语句：JPQL语句

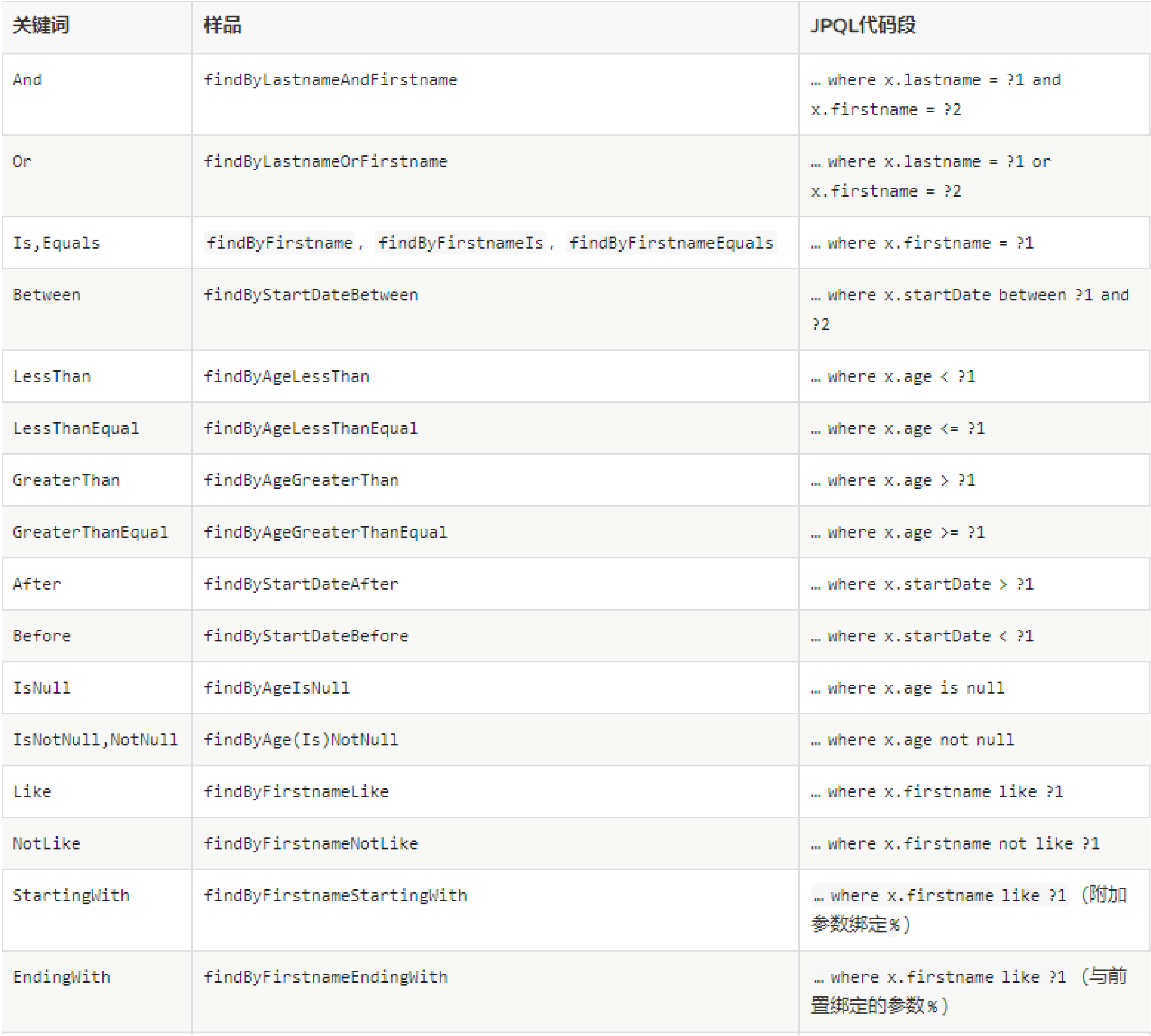
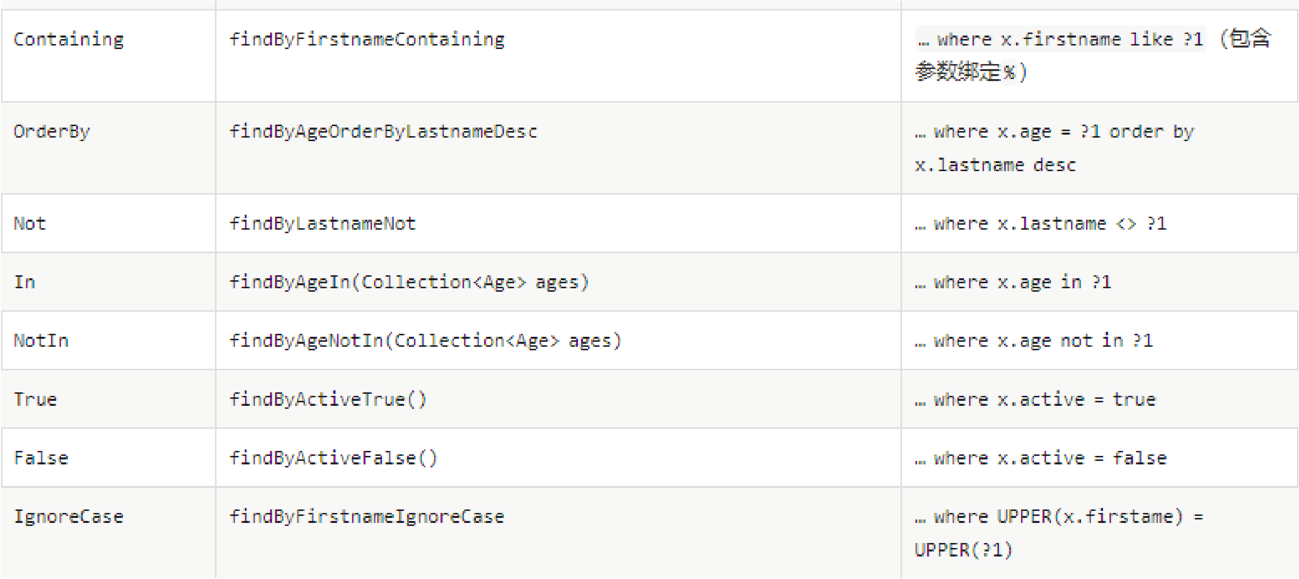
``

```
@NamedQueries({@NamedQuery(name = "",query = ""),
        @NamedQuery(name = "",query = "")})
@NamedQuery(name = "getAll",query = "from ItProject")
```

2.2.5 方法名解析查询

Spring Data Jpa 支持对方法名解析查询：按照一定的规则，定义方法，根据方法名的解析实现sql的自动生成。

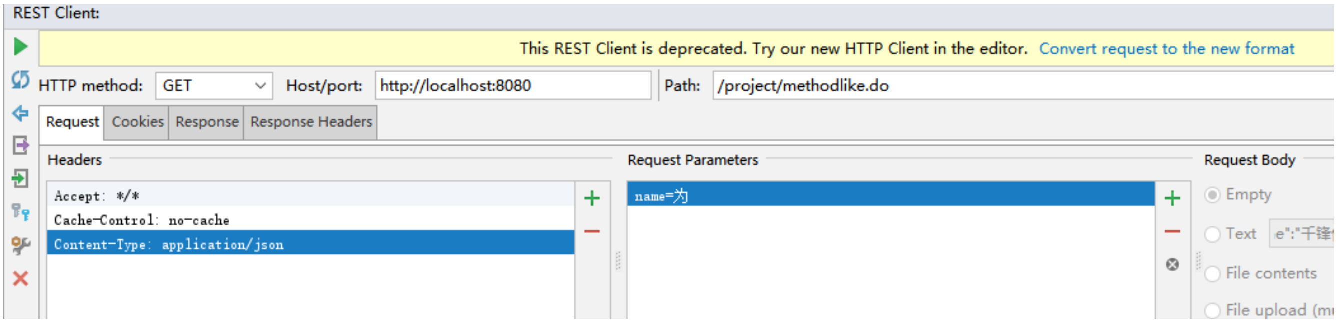
表3.方法名称中支持的关键字

关键词	样品	JPQL代码段
`And`	`findByLastnameAndFirstname`	`… where x.lastname = ?1 and x.firstname = ?2`
`Or`	`findByLastnameOrFirstname`	`… where x.lastname = ?1 or x.firstname = ?2`
`Is,Equals`	`findByFirstname`，`findByFirstnameIs`，`findByFirstnameEquals`	`… where x.firstname = ?1`
`Between`	`findByStartDateBetween`	`… where x.startDate between ?1 and ?2`
`LessThan`	`findByAgeLessThan`	`… where x.age < ?1`
`LessThanEqual`	`findByAgeLessThanEqual`	`… where x.age <= ?1`
`GreaterThan`	`findByAgeGreaterThan`	`… where x.age > ?1`
`GreaterThanEqual`	`findByAgeGreaterThanEqual`	`… where x.age >= ?1`
`After`	`findByStartDateAfter`	`… where x.startDate > ?1`
`Before`	`findByStartDateBefore`	`… where x.startDate < ?1`
`IsNull`	`findByAgeIsNull`	`… where x.age is null`
`IsNotNull,NotNull`	`findByAge(Is)NotNull`	`… where x.age not null`
`Like`	`findByFirstnameLike`	`… where x.firstname like ?1`
`NotLike`	`findByFirstnameNotLike`	`… where x.firstname not like ?1`
`StartingWith`	`findByFirstnameStartingWith`	`… where x.firstname like ?1`（附加参数绑定`%`）
`EndingWith`	`findByFirstnameEndingWith`	`… where x.firstname like ?1`（与前置绑定的参数`%`）
`Containing`	`findByFirstnameContaining`	`… where x.firstname like ?1`（包含参数绑定`%`）
`OrderBy`	`findByAgeOrderByLastnameDesc`	`… where x.age = ?1 order by x.lastname desc`
`Not`	`findByLastnameNot`	`… where x.lastname <> ?1`
`In`	`findByAgeIn(Collection<Age> ages)`	`… where x.age in ?1`
`NotIn`	`findByAgeNotIn(Collection<Age> ages)`	`… where x.age not in ?1`
`True`	`findByActiveTrue()`	`… where x.active = true`
`False`	`findByActiveFalse()`	`… where x.active = false`
`IgnoreCase`	`findByFirstnameIgnoreCase`	`… where UPPER(x.firstame) = UPPER(?1)`

``

```
//方法名解析查询
ItProject findById(int id);
List<ItProject> findByNameLike(String name);
```



REST Client:

This REST Client is deprecated. Try our new HTTP Client in the editor. Convert request to the new format

HTTP method: GET Host/port: http://localhost:8080 Path: /project/methodlike.do

Headers

```
Accept: */*
Cache-Control: no-cache
Content-Type: application/json
```

Request Parameters

```
name=为
```

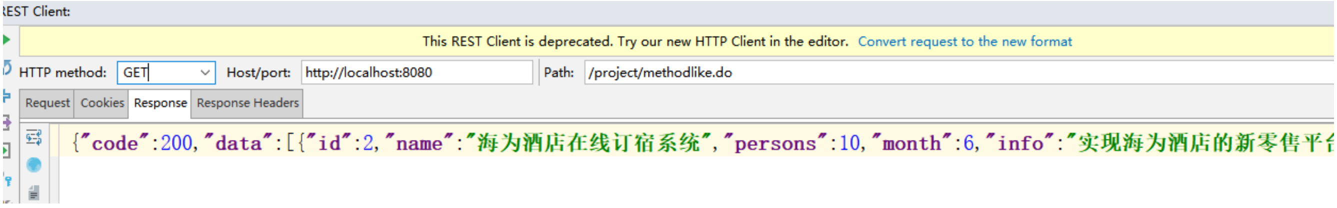
Request Body

● Empty

○ Text e":"千锋

○ File contents

○ File upload (m



REST Client:

This REST Client is deprecated. Try our new HTTP Client in the editor. Convert request to the new format

HTTP method: GET Host/port: http://localhost:8080 Path: /project/methodlike.do

```
{"code":200,"data":[{"id":2,"name":"海为酒店在线订宿系统","persons":10,"month":6,"info":"实现海为酒店的新零售平台
```

源码下载地址：https://github.com/xingpenghui/SpringBoot_1901