Atitit ql topic ---spel api

目录

[1.1. 功能分类4大类 1](#_Toc28614)

[1.2. 功能具体 1](#_Toc2990)

[2. 12：表达式模板 36 1](#_Toc21278)

[3. 5：调用方法 13 2](#_Toc29961)

[4. 4：SpEL中访问List、Map等元素集合 9 2](#_Toc2437)

[4.1. 查询语句 2](#_Toc29835)

[4.2. 安全导航查询单个属性节点 字段 2](#_Toc13431)

[4.3. From #root['data'] where id《10003 3](#_Toc17005)

## 功能分类4大类

5：调用方法 13

6：算术、比较、赋值、三元运算符 16

12：表达式模板 36

## 功能具体

Spring表达式语言：SpEL语法 2

1：直接量表达式 3

2：在表达式中创建数组 5

3：在表达式中创建List集合 7

4：SpEL中访问List、Map等元素集合 9

5：调用方法 13

6：算术、比较、赋值、三元运算符 16

7：类型运算符 19

8:调用构造器 21

9：安全导航操作 23

10:集合的选择 26

11：集合的投影 29

12：表达式模板 36

# 12：表达式模板 36

String sql = "insert into football\_match\_t( id,match\_event\_id, match\_status ,match\_time,tee\_time,"

// + "home\_id,away\_id,which\_round,neutral\_site,create\_time,delete\_flag,match\_detail,"

// + "animation,intelligence,squad,video)values("

// + "#{[id]},'#{[match\_event\_id]}','#{[match\_status]}','#{[match\_time]}','#{[tee\_time]}',"

// + "#{[home\_id]},#{[away\_id]},#{[which\_round]},#{[neutral\_site]},now(),0, '#{[match\_time]}',"

// + "0,0,0,0)";

// // sql = processVars(sql, json.getAsJsonObject("data") );

// sql = QlSpelUtil.parse(sql, param);

// logger.info(sql);

//

# 5：调用方法 13

T(class).m1(“xxxx”,”bbbb”)

New pkg.class().m1()

# 4：SpEL中访问List、Map等元素集合 9

## 查询语句

## 安全导航查询单个属性节点 字段

String expressionString = "#root['data'][0]['id']";

9：安全导航操作 23

10:集合的选择 26

11：集合的投影 29

## From #root['data'] where id《10003

expressionString = "#root['data'].?[['id']<10003]";

//"#myMap.?[key.length()<5&&value>90]"

# 调用bean

九、Bean引用：SpEL支持使用“@”符号来引用Bean，在引用Bean时需要使用BeanResolver接口实现来查找Bean，Spring提供BeanFactoryResolver实现；

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 | @Test  public void testBeanExpression() {   ClassPathXmlApplicationContext ctx = new ClassPathXmlApplicationContext();   ctx.refresh();   ExpressionParser parser = new SpelExpressionParser();   StandardEvaluationContext context = new StandardEvaluationContext();   context.setBeanResolver(new BeanFactoryResolver(ctx));   Properties result1 = parser.parseExpression("@systemProperties").getValue(context, Properties.class);   Assert.assertEquals(System.getProperties(), result1);  } |

# Other

3. 类表达式

SpEL使用T() 运算符调用类作用域的方法和常量。

@Value("#{T(java.lang.Math).PI}")

@Value("#{T(java.lang.Math).random()}")

@Value("#{T(java.lang.Math).random() \* 100.0 }")

1

2

3

4. 访问 properties

spel有两个可用的预定义变量 “systemProperties” 和 “systemEnvironment”。

systemProperties — java.util.Properties对象，从运行环境中检索属性。相当于java代码System.getProperty(arg0)

systemEnvironment — java.util.Properties对象，检索运行环境的具体属性。相当于java代码System.getenv(arg0)

@Value("#{ systemProperties['user.region'] }")

@Value("#{ systemEnvironment['profile'] }")

1

2

使用Spring的表达接口求值

下面的代码使用SpEL API来解析文本字符串表达式 Hello World.

ExpressionParser parser = new SpelExpressionParser();

Expression exp = parser.parseExpression("'Hello World'");

String message = (String) exp.getValue(); // "hello world"

exp = parser.parseExpression("'Hello World'.concat('!')");

message = (String) exp.getValue();// "Hello World!"

1

2

3

4

5

6

模板数据绑定

#用户评分

SCORE=auditInfo.auditInfo==null?"":auditInfo.auditInfo["score"]

#渠道号

CHANNEL\_NO=auditInfo.artificialAuditInfo.channelNo

#渠道名称

CHANNEL\_NAME=auditInfo.artificialAuditInfo.channelName

1

2

3

4

5

6

Map<String, Expression> expressions = new HashMap<>();

propertis.keySet().forEach(

k -> expressions.put(k, parser.parseExpression(propertis.getString(k))));

Map<String, Object> target = new HashMap<>();

expressions.keySet().forEach(key ->

target.put(key, expressions.get(key).getValue(origin))});

————————————————

版权声明：本文为CSDN博主「老螺丝」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/swordcenter/article/details/75239878

(···条消息)Spring表达式语言：SpEL语法\_Java\_VipMao的博客-CSDN博客