java 注解实战

动态SQL生成

Java我最强

王欣

欢迎关注

58赶集
后端高级工程师

GitHub

https://github.com/linkwechat

邮箱

linkwechat@foxmail.com

目录

1 基础入门

Java我最强

1. 什么是注解（Annotation）？

Annotation（注解）就是Java提供了一种元程序中的元素关联任何信息和着任何元数据（metadata）的途径和方法。Annotion(注解)是一个接口，程序可以通过反射来获取指定程序元素的Annotion对象，然后通过Annotion对象来获取注解里面的元数据。

Annotation(注解)是JDK5.0及以后版本引入的。它可以用于创建文档，跟踪代码中的依赖性，甚至执行基本编译时检查。从某些方面看，annotation就像修饰符一样被使用，并应用于包、类 型、构造方法、方法、成员变量、参数、本地变量的声明中。这些信息被存储在Annotation的“name=value”结构对中。

Annotation的成员在Annotation类型中以无参数的方法的形式被声明。其方法名和返回值定义了该成员的名字和类型。在此有一个特定的默认语法：允许声明任何Annotation成员的默认值：一个Annotation可以将name=value对作为没有定义默认值的Annotation成员的值，当然也可以使用name=value对来覆盖其它成员默认值。这一点有些近似类的继承特性，父类的构造函数可以作为子类的默认构造函数，但是也可以被子类覆盖。

1 基础入门

Java我最强

1. 什么是注解（Annotation）？

Annotation能被用来为某个程序元素（类、方法、成员变量等）关联任何的信息。需要注意的是，这里存在着一个**基本的规则：Annotation不能影响程序代码的执行，无论增加、删除 Annotation，代码都始终如一的执行。**另外，尽管一些annotation通过java的反射api方法在运行时被访问，而java语言解释器在工作时忽略了这些annotation。正是由于java虚拟机忽略了Annotation，导致了annotation类型在代码中是“不起作用”的； 只有通过某种配套的工具才会对annotation类型中的信息进行访问和处理。本文中将涵盖标准的Annotation和meta-annotation类型，陪伴这些annotation类型的工具是java编译器（当然要以某种特殊的方式处理它们）。

2. 元注解

元注解的作用就是负责注解其他注解。Java5.0定义了4个标准的meta-annotation类型，它们被用来提供对其它 annotation类型作说明。Java5.0定义的元注解：

1.**@Target**,
2.**@Retention**,
3.**@Documented**,
4.**@Inherited**

这些类型和它们所支持的类在java.lang.annotation包中可以找到。

3. @Target

@Target说明了Annotation所修饰的对象范围：Annotation可被用于 packages、types（类、接口、枚举、Annotation类型）、类型成员（方法、构造方法、成员变量、枚举值）、方法参数和本地变量（如循环变量、catch参数）。在Annotation类型的声明中使用了target可更加明晰其修饰的目标。

作用：用于描述注解的使用范围（即：被描述的注解可以用在什么地方）

取值(ElementType)有：

1.CONSTRUCTOR:用于描述构造器
2.FIELD:用于描述域
3.LOCAL_VARIABLE:用于描述局部变量
4.METHOD:用于描述方法
5.PACKAGE:用于描述包
6.PARAMETER:用于描述参数
7.TYPE:用于描述类、接口(包括注解类型) 或enum声明

4. @Retention

@Retention定义了该Annotation被保留的时间长短：某些Annotation仅出现在源代码中，而被编译器丢弃；而另一些却被编译在class文件中；编译在class文件中的Annotation可能会被虚拟机忽略，而另一些在class被装载时将被读取（请注意并不影响class的执行，因为Annotation与class在使用上是被分离的）。使用这个meta-Annotation可以对 Annotation的“生命周期”限制。

作用：表示需要在什么级别保存该注释信息，用于描述注解的生命周期（即：被描述的注解在什么范围内有效）

取值（RetentionPoicy）有：

1.SOURCE:在源文件中有效（即源文件保留）

2.CLASS:在class文件中有效（即class保留）

3.RUNTIME:在运行时有效（即运行时保留）

Retention meta-annotation类型有唯一的value作为成员，它的取值来自java.lang.annotation.RetentionPolicy的枚举类型值。

5. @Documented

@Documented用于描述其它类型的annotation应该被作为被标注的程序成员的公共API，因此可以被例如javadoc此类的工具文档化。Documented是一个标记注解，没有成员。

6. @Inherited

@Inherited 元注解是一个标记注解，@Inherited阐述了某个被标注的类型是被继承的。如果一个使用了@Inherited修饰的annotation类型被用于一个class，则这个annotation将被用于该class的子类。

注意：@Inherited annotation类型是被标注过的class的子类所继承。类并不从它所实现的接口继承annotation，方法并不从它所重载的方法继承annotation。

当@Inherited annotation类型标注的annotation的Retention是RetentionPolicy.RUNTIME，则反射API增强了这种继承性。如果我们使用java.lang.reflect去查询一个@Inherited annotation类型的annotation时，反射代码检查将展开工作：检查class和其父类，直到发现指定的annotation类型被发现，或者到达类继承结构的顶层。

2 项目实战

Java我最强

1. 什么场景下适合用注解?

近期的项目开发中遇到了一个比较棘手的问题，需要将数据库上线写入的数据保存起来以备出问题时可以及时回滚，同时希望保存的回滚数据是可视化的SQL，以便出问题时可以手动修复。

如果使用传统的方式开发，需要依据每个数据库实体的字段依次拼接SQL，如果只是一个实体人工可以处理过来，可如果有几十个这就不是人力可以简单办到的，这样机械性质的工作无法避免会出现**高错误率**，而且当数据库实体更新时，又会是进入新一轮的更新地狱。

但是冷静想想，虽然需要生成SQL的实体差别很大，但是每个实体转换SQL的方式却大同小异，也就是说转换的过程可以抽象成一个统一的流水线，每个实体可以加上一些说明来表明转换规则，这样一个统一的转换过程就可以完成了，而注解正是完成这个构想的关键性的说明。

所以综上所述，注解比较适用的范围是**相同性质对象进行统一过程处理**的场景，接下来我们通过实际的项目来学习注解的使用。

2 项目实战

Java我最强

2. 注解定义

```java
package com.linkwechat.action.annotation;

import java.lang.annotation.Documented;

/**
 * 数据库表名的注解
 *
 * @author linkwechat linkwechat@foxmail.com
 */
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Table {
    /**
     * 表明
     *
     * @return String
     */
    String name() default "className";
}
```

2 项目实战

Java我最强

2. 注解定义

```java
package com.linkwechat.action.annotation;

import java.lang.annotation.Documented;

/**
 * 数据库表列名字段的注解
 *
 * @author linkwechat linkwechat@foxmail.com
 */
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Column {
    /**
     * 列名
     *
     * @return String
     */
    String name() default "fieldName";
}
```

2. 注解定义

```java
package com.linkwechat.action.annotation;

import java.lang.annotation.Documented;

/**
 * 非数据库表列名字段的注解
 *
 * @author linkwechat linkwechat@foxmail.com
 */
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface NotDBColumn {
}
```

2. 注解定义

```java
package com.linkwechat.action.annotation;

import java.lang.annotation.Documented;

/**
 * 数据库表主键id字段的注解
 *
 * @author linkwechat linkwechat@foxmail.com
 */
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Id {
}
```

2 项目实战

Java我最强

3. 实体定义

```java
package com.linkwechat.action.entity;

import java.io.Serializable;

/**
 * 用户实体
 *
 * @author linkwechat linkwechat@foxmail.com
 * @version 1.0
 */
@Table(name = "t_user")
public class User implements Serializable {

    private static final long serialVersionUID = 1L;

    /**
     * 用户编号
     */
    @Id
    private Long id;

    /**
     * 用户名称
     */
    private String username;

    /**
     * 用户手机
     */
    @Column(name = "phone")
    private String mobile;

    /**
     * 用户邮件
     */
    private String email;

    /**
     * 时间戳
     */
    @NotDBColumn
    private Date timestamp;

    public Long getId() {
        return id;
```

3. 关键方法

```
private static final Log log = LogFactory.getLog(SQLGenerator.class);

private static final String charset = "UTF-8";

/**
 * 转义正则特殊字符
 *
 * @param keyword
 *            关键词
 * @return String
 */
public static String escapeExprSpecialWord(String keyword) {
    if (StringUtils.isNotBlank(keyword)) {
        String[] fbsArr = { "\\", "'", "\"" };
        for (String key : fbsArr) {
            if (keyword.contains(key)) {
                keyword = keyword.replace(key, "\\" + key);
            }
        }
    }
    return keyword;
}
```

3. 关键方法

```
/**
 * 获取对象中指定字段的值
 *
 * @param obj
 *            待获取对象
 * @param fieldName
 *            字段 名称
 * @return String
 */
public static String getObjFieldValue(Object obj, String fieldName) throws Exception {
    if (obj == null || fieldName == null || fieldName.equals("")) {
        return null;
    }

    // 将属性的首字符大写，构造get方法
    String method = "get" + fieldName.substring(0, 1).toUpperCase() + fieldName.substring(1);
    try {
        // 调用getter方法获取属性值
        Method m = obj.getClass().getMethod(method);
        Object value = m.invoke(obj);

        String valueStr = null;
        if (value == null) {
            return "null";
```

3. 关键方法

```
            if (value == null) {
                return "null";
            } else if (value instanceof String) {
                valueStr = "'" + escapeExprSpecialWord(String.valueOf(value)) + "'";
            } else if (value instanceof Short || value instanceof Integer || value instanceof Long
                    || value instanceof Float || value instanceof Double) {
                valueStr = String.valueOf(value);
            } else if (value instanceof Boolean) {
                if ((Boolean) value) {
                    valueStr = "1";
                } else {
                    valueStr = "0";
                }
            }
            if (value instanceof Date) {
                valueStr = "'" + DateUtils.getMySQLDate((Date) value) + "'";
            }

            return valueStr;
        } catch (Exception e) {
            log.error("Execute " + obj.getClass().getName() + "." + method + "() error!", e);
            throw e;
        }
    }
```

2 项目实战

Java我最强

3. 关键方法

```
/**
 * 获取对象中SQL主键的字段和取值
 *
 * @param obj
 *            待获取对象
 * @return Map&lt;String,String&gt;
 */
public static Map<String, String> getSQLPrimaryColumns(Object obj) throws Exception {
    if (obj == null) {
        return null;
    }

    Map<String, String> sqlPrimaryColumns = new LinkedHashMap<String, String>();
    Field[] fields = obj.getClass().getDeclaredFields();
    for (Field field : fields) {
        if (field.isAnnotationPresent(Id.class)) {
            if (field.isAnnotationPresent(NotDBColumn.class)) {
                continue;
            } else if (field.isAnnotationPresent(Column.class)) {
                String value = getObjFieldValue(obj, field.getName());
                if (value == null) {
                    continue;
                }
```

3. 关键方法

```
        for (Field field : fields) {
            if (field.isAnnotationPresent(Id.class)) {
                if (field.isAnnotationPresent(NotDBColumn.class)) {
                    continue;
                } else if (field.isAnnotationPresent(Column.class)) {
                    String value = getObjFieldValue(obj, field.getName());
                    if (value == null) {
                        continue;
                    }
                    Column column = field.getAnnotation(Column.class);
                    String key = column.name();
                    sqlPrimaryColumns.put(key, value);
                } else {
                    String value = getObjFieldValue(obj, field.getName());
                    if (value == null) {
                        continue;
                    }
                    String key = field.getName();
                    sqlPrimaryColumns.put(key, value);
                }
            }
        }
        return sqlPrimaryColumns;
    }
```

2 项目实战

Java我最强

3. 关键方法

```
/**
 * 依据数据库对象创建删除的SQL
 *
 * @param objList
 *            数据库对象列表
 * @return List&lt;String&gt;
 */
public static List<String> createDeleteSQL(List<?> objList) throws Exception {
    if (objList == null || objList.size() == 0) {
        return null;
    }

    List<String> sqlList = new ArrayList<String>();
    for (Object obj : objList) {
        if (!obj.getClass().isAnnotationPresent(Table.class)) {
            continue;
        }
        Table table = obj.getClass().getAnnotation(Table.class);

        Map<String, String> sqlPrimaryColumns = getSQLPrimaryColumns(obj);
        if (sqlPrimaryColumns == null || sqlPrimaryColumns.size() == 0) {
            continue;
        }
```

3. 关键方法

```
        Table table = obj.getClass().getAnnotation(Table.class);

        Map<String, String> sqlPrimaryColumns = getSQLPrimaryColumns(obj);
        if (sqlPrimaryColumns == null || sqlPrimaryColumns.size() == 0) {
            continue;
        }

        StringBuilder sb = new StringBuilder();
        sb.append("delete from `").append(table.name()).append("` where ");

        StringBuilder sbPrimaryKey = new StringBuilder();
        for (String column : sqlPrimaryColumns.keySet()) {
            sbPrimaryKey.append("`").append(column).append("` = ").append(sqlPrimaryColumns.get(column))
                    .append(" and");
        }

        if (sbPrimaryKey.length() > 4) {
            String keys = sbPrimaryKey.substring(0, sbPrimaryKey.length() - 4);
            sb.append(keys).append(";");
            sqlList.add(sb.toString());
        }
    }
    return sqlList;
}
```

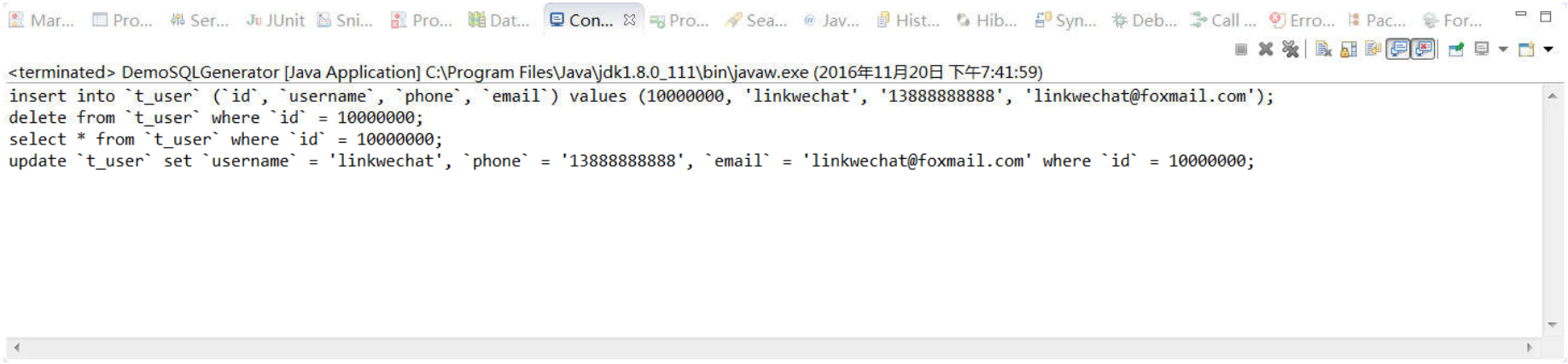
4. 使用示例

```
public class DemoSQLGenerator {

    public static void main(String[] args) {
        User user = new User();
        user.setId(10000000L);
        user.setUsername("linkwechat");
        user.setMobile("13888888888");
        user.setEmail("linkwechat@foxmail.com");
        user.setTimestamp(new Date());

        List<User> userList = new ArrayList<User>();
        userList.add(user);

        try {
            System.out.println(SQLGenerator.createInsertSQL(userList).get(0));
            System.out.println(SQLGenerator.createDeleteSQL(userList).get(0));
            System.out.println(SQLGenerator.createSelectSQL(userList).get(0));
            System.out.println(SQLGenerator.createUpdateSQL(userList).get(0));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

2 项目实战

Java我最强

4. 使用示例



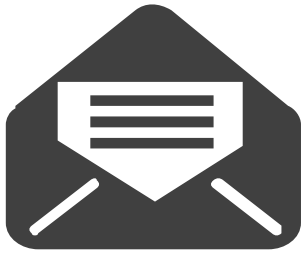
3 Q&A

Java我最强

There Is No End to Learning

GitHub

https://github.com/linkwechat



邮箱

linkwechat@foxmail.com

Java我最强

THANK YOU

感谢聆听

基础入门篇参考文献

http://www.cnblogs.com/peida/archive/2013/04/23/3036035.html
http://www.cnblogs.com/peida/archive/2013/04/24/3036689.html

项目实战篇示例代码

https://github.com/linkwechat/linkwechat-annotation