链滴

jpa 规范的模拟数据生成代码

作者：wuhongxu

原文链接：https://ld246.com/article/1490130731065

来源网站：链滴

jpa规范的模拟数据生成代码

在做项目的时候想到，我们写代码很多时候需要测试，如果每个实体类都要去数据库将数据建好，那多么的麻烦的一件事情，为什么不能通过一个工具将模拟数据生成好，我们直接可以拿来用呢？

基于以上想法，于是决定写一个能生成模拟数据的工具类。

有什么需求?

- 主键是自动生成的，那么就不要去生成了
- 字段规定了长度的，那么生成的数据就应该在此长度之内
- 最好能通过setter方法来设置值
- 数据尽量做到随机
- 有些字不想设置的，能够很简单的避开

具体实现代码

以下代码可以直接拿到spring环境中使用，注释已经写在代码中：

首先是 TestTool类：

```
package site.zido.cydServer.common.tools.testToolbox;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;
import site.zido.cydServer.common.tools.StringTool;

import javax.annotation.Resource;
import javax.persistence.Column;
import javax.persistence.GeneratedValue;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * 用于测试的工具类.
 *
 * 生成数据方法支持所有基本类型的包装类型以及Date类型
 * Date: 2017/3/22 0022
 * Time: 2:44
 *
 * @author <a href="http://zido.site">wuhongxu</a>.
 * @version 1.0.0
```

```
 */
@Component
public class TestTool {
    private static Logger logger = LoggerFactory.getLogger(TestTool.class);
    @Resource
    private StringTool stringTool;

    /**
     * 自动向实体类注入模拟值
     * @param entities 实体类
     * @param <T> 类型
     */
    public <T> void generateEntities(T... entities) {
        for (T entity : entities) {
            generate(entity);
        }
    }

    /**
     * 通过类型和长度生成模拟实体集合
     * @param _class 类型
     * @param length 长度
     * @param <T> 类型
     * @return 模拟实体集合
     */
    public <T> List<T> generateEntities(Class<T> _class,int length) {
        List<T> list = new ArrayList<>();
        for (int i = 0; i < length; i++) {
            list.add(generate(_class));
        }
        return list;
    }

    /**
     * 根据类型生成一个已注入模拟值的实体类
     * @param _class 类型集合
     * @param <T> 类型
     * @return 实体类
     */
    public <T> T generate(Class<T> _class) {
        T entity = null;
        try {
            entity = _class.newInstance();
        } catch (InstantiationException e) {
            logger.error("必须设置一个公有无参构造函数");
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
        final Field[] fields = _class.getDeclaredFields();
        for (Field field : fields) {
            injectValue(entity, field);
        }
        return entity;
```

```
}

/**
 * 自动注入模拟值
 * @param entity 实体类
 * @param <T> 类型
 */
public <T> void generate(T entity) {
    final Field[] fields = entity.getClass().getDeclaredFields();
    for (Field field : fields) {
        injectValue(entity, field);
    }
}

/**
 * 向实体注入值
 * @param entity 实体的实例
 * @param field 属性
 * @param <T> 类型
 */
private <T> void injectValue(T entity, Field field) {
    final Class<?> _class = entity.getClass();
    final Class<?> type = field.getType();
    if (field.getAnnotation(NoInjectValue.class) != null)
        return;
    final GeneratedValue generatedValue = field.getAnnotation(GeneratedValue.class);
    if (generatedValue != null)
        return;
    final Column annotation = field.getAnnotation(Column.class);

    if (type == String.class) {
        int length;
        if (annotation == null)
            length = 255;
        else {
            length = annotation.length();
        }
        final String s = randomString(length, true);

        try {
            final String name = captureName(field.getName());
            final Method method = _class.getMethod("set" + name, String.class);
            method.invoke(entity, s);
        } catch (NoSuchMethodException e) {
            logger.error(field.getName() + "值必须有set方法");
            e.printStackTrace();
        } catch (IllegalAccessException | InvocationTargetException e) {
            logger.error(field.getName() + "注入值出错");
            e.printStackTrace();
        }
        return;
    }
    if (type == Date.class) {
        try {
```

```
                final String name = captureName(field.getName());
                final Method method = _class.getMethod("set" + name, Date.class);
                method.invoke(entity, new Date());
            } catch (NoSuchMethodException e) {
                logger.error(field.getName() + "值必须有set方法");
                e.printStackTrace();
            } catch (IllegalAccessException | InvocationTargetException e) {
                logger.error(field.getName() + "注入值出错");
                e.printStackTrace();
            }
            return;
        }
        int length;
        if (annotation == null)
            length = 9;
        else {
            final int l = annotation.length();
            length = l > 9 ? 9 : l;
        }
        try {
            Constructor<?> constructor = field.getType().getConstructor(String.class);
            final String name = captureName(field.getName());
            final Method method = _class.getMethod("set" + name, String.class);
            method.invoke(entity, constructor.newInstance(randomInt(length, true) + ""));
        } catch (NoSuchMethodException | IllegalAccessException | InstantiationException | Invo
cationTargetException e) {
            logger.error("---------------属性名为" + field.getName() + "的类型不支持----------------
");
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * 生成随机int值
     *
     * @param length   int值的长度
     * @param lessThan 是否允许小于此长度
     * @return int值
     */
    public int randomInt(int length, boolean lessThan) throws Exception {
        if(length > 9){
            throw new Exception("不支持长度大于9的情况");
        }
        if(!lessThan){
            StringBuilder result = new StringBuilder("");
            for (int i = 0; i < length; i++) {
                if (i == 0) {
                    result.append(((int) (Math.random() * 9)) + 1);
                    continue;
                }
                result.append((int) (Math.random() * 10));
            }
```

```
            return Integer.parseInt(result.toString());
        }
        final int pow = (int)Math.pow(10, length);
        return (int) (Math.random()*pow);
    }
    /**
     * 生成随机String字符串
     *
     * @param length   String字符串的长度
     * @param lessThan 是否允许小于此长度
     * @return int值
     */
    public String randomString(int length,boolean lessThan){
        StringBuilder result = new StringBuilder("");
        for (int i = 0; i < length; i++) {
            if(lessThan && Math.random() > 0.5){
                continue;
            }
            char c = ' ';
            char or = (char) ((int) (Math.random() * 94) + c);
            result.append(or);
        }
        return result.toString();
    }

    /**
     * 字符串首字母变大写
     * @param str 字符串
     * @return
     */
    public String captureName(String str) {
        char[] cs=str.toCharArray();
        cs[0]-=32;
        return String.valueOf(cs);
    }
}
```

然后是不需要注入模拟值字段的注解 NoInjectValue：

```
package site.zido.cydServer.common.tools.testToolbox;

import java.lang.annotation.*;

/**
 * 在生成模拟实体的时候会读取此注解，如果有此注解的属性，将不会被注入模拟值.
 * Date: 2017/3/22 0022
 * Time: 3:19
 *
 * @author <a href="http://zido.site">wuhongxu</a>.
 * @version 1.0.0
 */
```

```
@Target(ElementType.FIELD) //此注解只能放在属性上
@Retention(RetentionPolicy.RUNTIME) // 注解会在class字节码文件中存在，在运行时可以通过反
获取到
@Documented//说明该注解将被包含在javadoc中
public @interface NoInjectValue {
}
```

怎么使用?

见我的测试类

```
package site.zido.cydServer.repository;

        import org.junit.Test;
        import org.junit.runner.RunWith;
        import org.springframework.boot.test.context.SpringBootTest;
        import org.springframework.test.context.junit4.SpringRunner;
        import org.springframework.transaction.annotation.Transactional;
        import site.zido.cydServer.common.tools.testToolbox.TestTool;
        import site.zido.cydServer.entities.Goods;

        import javax.annotation.Resource;
        import java.util.List;

/**
 * cyd.
 * Date: 2017/3/22 0022
 * Time: 2:21
 *
 * @author <a href="http://zido.site">wuhongxu</a>.
 * @version 1.0.0
 */
@SpringBootTest
@RunWith(SpringRunner.class)
@Transactional
public class GoodsRepositoryTest {
    @Resource
    private GoodsRepository repository;
    @Resource
    private TestTool tools;
    /**
     * 存储数据测试
     */
    @Test
    public void testSaveRepository(){
        final Goods goods = tools.generate(Goods.class);
        final Goods save = repository.save(goods);
        System.out.println(save.getId());
        assert save.getId()!=null;
    }


    @Test
```

```
    public void testGenerate(){
        //注意这些类的id并没有写入模拟数据哟
        final List<Goods> goodss = tools.generateEntities(Goods.class, 20);
        repository.save(goodss);
    }

}
```

实体类

```
package site.zido.cydServer.entities;

import site.zido.cydServer.common.tools.testToolbox.NoInjectValue;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import java.util.Date;

/**
 * cyd.
 * Date: 2017/3/18 0018
 * Time: 18:35
 *
 * @author <a href="http://zido.site">wuhongxu</a>.
 * @version 1.0.0
 */
@Entity
public class Goods {
    @Id
    @GeneratedValue
    private Long id;
    @Column(nullable = false)
    private String title;
    @Column(nullable = false)
    private Date createTime;
    @Column(nullable = false)
    private String content;
    @NoInjectValue
    private String anyOne;

    public Long getId() {
        return id;
    }

    public Goods setId(Long id) {
        this.id = id;
        return this;
    }

    public String getTitle() {
        return title;
    }
```

```
    public Goods setTitle(String title) {
        this.title = title;
        return this;
    }

    public Date getCreateTime() {
        return createTime;
    }

    public Goods setCreateTime(Date createTime) {
        this.createTime = createTime;
        return this;
    }

    public String getContent() {
        return content;
    }

    public Goods setContent(String content) {
        this.content = content;
        return this;
    }

    public String getAnyOne() {
        return anyOne;
    }

    public Goods setAnyOne(String anyOne) {
        this.anyOne = anyOne;
        return this;
    }
}
```

最后是我的招牌笑容smile

希望会有人觉得有用，嘿嘿