代码生成器说明文档

1.什么是代码生成器

代码生成器顾名思义就生成代码的一个软件。为了节省人力成本，在日常的企业开发中，代码生成器使用的较为普遍。其实在我们目前所学的技术中，就有代码生成器的影子。比如我们学习的 JSP 技术，浏览器并不能直接识别 JSP 文件。当用户通过浏览器访问某个 JSP 页面的时候，其中的交互分为分为几个步骤：

- 用户通过浏览器访问 JSP 页面
- Tomcat 找到对应 JSP 资源，将 JSP 资源转化(编译，翻译)成为 HTML 数据
- Tomcat 将 HTML 数据进行返回

在此，我们就可以这样理解

- 我们所编写 JSP 其实是编写了一个 HTML 的模板。
- JSP 页面的标签，相当于模板中的占位符。
- Tomcat 根据 JSP 模板，生成 HTML 页面。
- 数据库的数据不同，则生成的 HTML 页面也不同。

最后我们得出一个结论：

Tomcat 可以根据 JSP 模板和数据生成不同的 HTML 页面。所以 Tomcat 对 JSP 的处理过程我们就可以认为是一个代码生成器的机制。

2.代码生成器原理

在第 1 小节中，其实我们已经以 Tomcat 和 JSP 举例，为大家说明了代码生成器的原理，在此我们再来总结一下。代码生成器的原理就是：**根据模板+数据生成不同文件。**

比如以下的两段代码。

1.ItripHotelOrderMapper 文件

```java
package cn.itrip.mapper.itripHotelOrder;
import ...

public interface ItripHotelOrderMapper {

    public ItripHotelOrder getItripHotelOrderById(@Param(value = "id") Long id)throws Exception;

    public List<ItripHotelOrder>    getItripHotelOrderListByMap(Map<String,Object> param)throws Exception;

    public Integer getItripHotelOrderCountByMap(Map<String,Object> param)throws Exception;

    public Integer insertItripHotelOrder(ItripHotelOrder itripHotelOrder)throws Exception;

    public Integer updateItripHotelOrder(ItripHotelOrder itripHotelOrder)throws Exception;

    public Integer deleteItripHotelOrderById(@Param(value = "id") Long id)throws Exception;

}
```

2.ItripHotelOrderMapper 文件

```java
package cn.itrip.mapper.itripHotelRoom;
import ...

public interface ItripHotelRoomMapper {

    public ItripHotelRoom getItripHotelRoomById(@Param(value = "id") Long id)throws Exception;

    public List<ItripHotelRoom> getItripHotelRoomListByMap(Map<String,Object> param)throws Exception;

    public Integer getItripHotelRoomCountByMap(Map<String,Object> param)throws Exception;

    public Integer insertItripHotelRoom(ItripHotelRoom itripHotelRoom)throws Exception;

    public Integer updateItripHotelRoom(ItripHotelRoom itripHotelRoom)throws Exception;

    public Integer deleteItripHotelRoomById(@Param(value = "id") Long id)throws Exception;

}
```

在上述 ItripHotelOrderMapper 和 ItripHotelOrderMapper 文件中，我们注意到，两个文件的代码很相似。除了高亮部分不一致以外，其它的代码均一致。因此，我们可以设想，我们能不能编写模板生成这样的 Mapper 类。高亮部分使用占位符来进行代替，具体内容根据具体数据进行置换。

3.代码生成器三要素

以上我们了解了代码生成器的概念和原理，接下来我们开始着手编写属于我们自己的代码生成器。我们将代码生成器的分为三部分：

- 模板：生成文件的模板文件。
- 数据：生成文件所需要的关键数据。
- 合成机制：使用数据置换模板中的占位符，生成新的文件的机制。

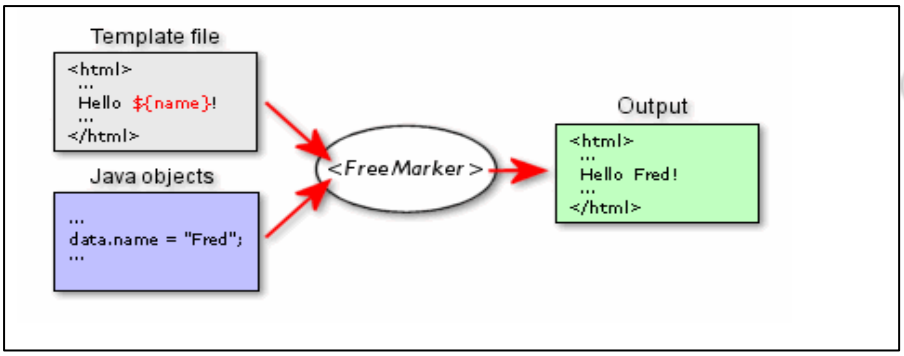
其中数据我们可以从数据库中进行获取，模板文件，以及根据模板文件生成代码的机制我们选用较为成熟的模板引擎: FreeMarker

3. FreeMarker

3.1 FreeMarker 介绍

FreeMarker 是一款模板引擎： 即一种基于模板和要改变的数据，并用来生成输出文本（HTML 网页、电子邮件、配置文件、源代码等）的通用工具。FreeMarker 是免费的，基于 Apache 许可证 2.0 版本发布。其模板编写为 FreeMarker Template Language（FTL），属于简单、专用的语言。

FreeMarker 原理介绍:



3.2 FreeMarker 数据合成

综上所述 FreeMarker 就是根据模板和数据生成我们想要的文件。具体的 FreeMarker 模板文件如下图所示。

test.ftl

```html
<html>
<head>
    <title>${name}</title>
</head>
<body>
    <p>${msg}</p>
</body>
</html>
```

HelloFreeMarker 根据 test.ftl 生成 HTML 数据

```java
public class HelloFreeMarker{
    private Configuration cfg;
    public void init() throws Exception{
        //初始化FreeMarker配置
        //创建一个Configuration实例
        cfg = new Configuration();
        //设置FreeMarker的模版文件位置
        cfg.setDirectoryForTemplateLoading(new File("templates"));
    }

    public void process()throws Exception{
        Map root = new HashMap();
        root.put("name", "FreeMarker!");
        root.put("msg" , "您已经完成了第一个FreeMarker的示例");
        Template t = cfg.getTemplate("test.ftl");
        t.process(root, new OutputStreamWriter(System.out));
    }

    public static void main(String[] args)throws Exception{
        HelloFreeMarker hf = new HelloFreeMarker();
        hf.init();
        hf.process();
    }
```

3.3 FreeMarker 语法

3.3.1 输出数据

FreeMarker 可以输出的数据包括基本数据类型的数据和对象数据类型的数据，这其中包括输出对象数据类型的属性的值。

- 输出基本技术类型或对象属性的语法:

- **${数据}或者${数据.属性}**

Eg1:

```
package cn.${package}.pojo;
import java.io.Serializable;
import java.util.Date;
/**
*   ${table.comment}
*/
public class ${table.className} implements Serializable {
        <#list table.cloumns as cloumn>
        //${cloumn.comment}
        private ${cloumn.javaType} ${cloumn.fieldName};
        </#list>
        //get set 方法
        <#list table.cloumns as cloumn>
            public void set${cloumn.upperCaseCloumnName} (${cloumn.javaType}  ${cloumn.fieldName}){
                this.${cloumn.fieldName}=${cloumn.fieldName};
            }
            public  ${cloumn.javaType} get${cloumn.upperCaseCloumnName}(){
                return this.${cloumn.fieldName};
            }
        </#list>
}
```

3.3.2 循环读取

```
<#list sequence as item>
    <#if item_has_next>.....
    <#else>.....
    </#if>
</#list>
```

```
<select id="get${table.className}CountByMap" resultType="Integer"  parameterType="java.util.Map">
    select count(*) from ${table.tableName}
    <trim prefix="where" prefixOverrides="and | or">
    <#list table.columns as column>
        <#if column_has_next>
            <if test="${column.fieldName} != null and ${column.fieldName}!=''">
                and ${column.columnName}=${r"#{"}${column.fieldName}}
            </if>
        <#else>
            <if test="${column.fieldName} != null and ${column.fieldName}!=''">
                and ${column.columnName}=${r"#{"}${column.fieldName}}
            </if>
        </#if>
    </#list>
    </trim>
</select>
```

3.3.3 逻辑判断

语法：

```
<#if condition>...
    <#elseif condition2>...
    <#elseif condition3>.....
    <#else>.....
</#if>
```

Eg：

```
<select id="get${table.className}CountByMap" resultType="Integer"  parameterType="java.util.Map">
    select count(*) from ${table.tableName}
    <trim prefix="where" prefixOverrides="and | or">
    <#list table.columns as column>
        <#if column_has_next>
            <if test="${column.fieldName} != null and ${column.fieldName}!=''">
                and ${column.columnName}=${r"#{"}${column.fieldName}}
            </if>
        <#else>
            <if test="${column.fieldName} != null and ${column.fieldName}!=''">
                and ${column.columnName}=${r"#{"}${column.fieldName}}
            </if>
        </#if>
    </#list>
    </trim>
</select>
```

4.数据之 JDBC

4.1 使用 JDBC 获取数据库所有表信息

```
//连接数据库
Class.forName(DBDRIVER);
conn = DriverManager.getConnection(DBURL, DBUSER, DBPASS);
DatabaseMetaData dbmd = conn.getMetaData();
ResultSet resultSet = dbmd.getTables(null, null, null, new String[]{"TABLE"});
while (resultSet.next()) {
    //获取表名并输出
    String tableName = resultSet.getString("TABLE_NAME");
    System.out.println(tableName);
}
```

4.2 使用 JDBC 获取数据库表注释

```
private static String getCommentByTableName(String tableName) throws Exception {
        Connection conn = DriverManager.getConnection(DBURL, DBUSER, DBPASS);
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SHOW CREATE TABLE " + tableName);
        String comment = null;
        if (rs != null && rs.next()) {
            comment = parse(rs.getString(2));
        }
        rs.close();
        stmt.close();
        conn.close();
        return comment;
 }
```

4.3 使用 JDBC 获取数据库表列信息

```
ResultSet rs = dbmd.getColumns(null, "%", tableName, "%");

while (rs.next()) {

        //获取列名

        System.out.println(rs.getString("COLUMN_NAME"));

        //获取列注释

        System.out.println(rs.getString("REMARKS"));

        //获取列类型

        System.out.println(rs.getString("TYPE_NAME"));

 }
```

5.Module 模板

```
package cn.${package}.pojo;
import java.io.Serializable;
import java.util.Date;
/***
*   ${table.comment}
*/
public class ${table.className} implements Serializable {
        <#list table.columns as column>
        //${column.comment}
        private ${column.javaType} ${column.fieldName};
        </#list>
        //get set 方法
        <#list table.columns as column>
            public void set${column.upperCasecloumnName} (${column.javaType}  ${column.fieldName}){
                this.${column.fieldName}=${column.fieldName};
            }
            public  ${column.javaType} get${column.upperCasecloumnName}(){
                return this.${column.fieldName};
            }
        </#list>
}
```

6.Mapper.xml 模板

```
import java.util.List;
import java.util.Map;

public interface ${table.className}Mapper {

    public ${table.className} get${table.className}ById(@Param(value = "id") Long id)throws Exception;

    public List<${table.className}> get${table.className}ListByMap(Map<String,Object> param)throws Exception;

    public Integer get${table.className}CountByMap(Map<String,Object> param)throws Exception;

    public Integer insert${table.className}(${table.className} ${lowerClassName})throws Exception;

    public Integer update${table.className}(${table.className} ${lowerClassName})throws Exception;

    public Integer delete${table.className}ById(@Param(value = "id") Long id)throws Exception;

}
```