一、 Java教程

1. HelloWorld

创建文件 HelloWorld.java(**文件名需与类名一致**), 代码如下:

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
#当然你得把javac和java加入PATH, "c:\Program Files\Java\jdk1.8.0_181\bin\javac"
$ javac HelloWorld.java
$ java HelloWorld
Hello World
```

入门内容详见： http://www.runoob.com/java/java-tutorial.html

2. JAVA案例：必应bing每日壁纸

必应搜索官网每天都会选取一幅高清美图作为背景，如何让自己的电脑每天同步**必应每日图片**作为桌面背景呢?

2.1. 案例思路

- 如何获取必应bing每日壁纸的地址
  - 解析bing首页的HTML
    - HTTP请求
    - 正则表达式
  - 查看有没有接口提供
    - HTTP请求
    - 分析接口协议（request，response）
    - xml，json解析器
  - 如何根据本机的分辨率匹配
- 如何将获取的图片设置到本机壁纸
  - window壁纸的设置原理
    - 注册表：计算机\HKEY_CURRENT_USER\Control Panel\Desktop
      - WallPaper
      - WallpaperStyle
    - 修改完后如何刷新桌面
    - 兼容性，window XP只支持BMP

        - 如何将JPG转换成BMP
    - 通过java进行设置
        - jna
            - 注册表
            - SystemParametersInfoA
            - DLL查看器
- 扩展功能
    - 定时切换
    - 开机启动
    - 保存历史记录
    - 支持Ubuntu等linux桌面

2.2. 技术调研

相关jar下载

- https://mvnrepository.com/artifact/net.java.dev.jna/jna-platform
- https://mvnrepository.com/artifact/net.java.dev.jna/jna

修改壁纸可通过改windows注册表实现

```
Advapi32Util.registrySetStringValue(WinReg.HKEY_CURRENT_USER,
                                    "Control Panel\\Desktop", "Wallpaper", fullFnm);
//WallpaperStyle = 10 (Fill), 6 (Fit), 2 (Stretch), 0 (Tile), 0 (Center)
//For windows XP, change to 0
Advapi32Util.registrySetStringValue(WinReg.HKEY_CURRENT_USER,
                                    "Control Panel\\Desktop", "WallpaperStyle", "10");
Advapi32Util.registrySetStringValue(WinReg.HKEY_CURRENT_USER,
                                    "Control Panel\\Desktop", "TileWallpaper", "0");
```

然后要调用User32.dll里的SystemParametersInfoA函数刷新桌面，并将更改通知给其他程序。

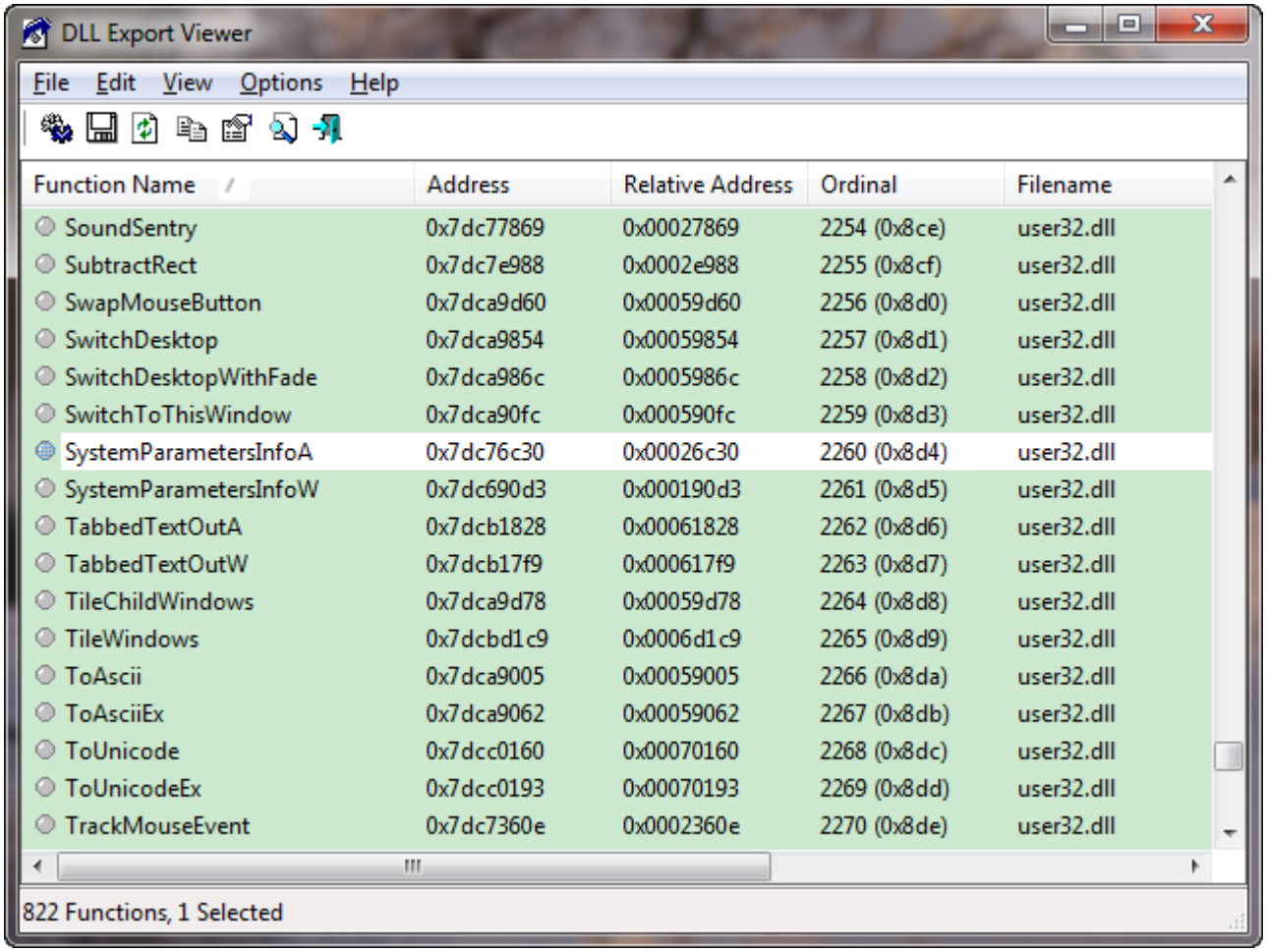
```
// refresh the desktop using User32.SystemParametersInfo(), so avoiding an OS reboot
int SPI_SETDESKWALLPAPER = 0x14;
int SPIF_UPDATEINIFILE = 0x01;
int SPIF_SENDWININICHANGE = 0x02;

boolean result = MyUser32.INSTANCE.SystemParametersInfoA(SPI_SETDESKWALLPAPER, 0,
                                                         fullFnm, SPIF_UPDATEINIFILE |
SPIF_SENDWININICHANGE );

```

如下图可看到User32.dll里有个SystemParametersInfoA函数 （http://www.nirsoft.net/utils/dll_export_viewer.html）

选择打开C:\WINDOWS\system32\user32.dll文件



2.3. 技术实现

2.3.1. Ctrl + C && Ctrl + V

在网上随便抄一段代码：我们不生产代码，我们只是代码的搬运工

```
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.util.HashMap;
import com.sun.jna.Native;
import com.sun.jna.platform.win32.WinDef.UINT_PTR;
import com.sun.jna.win32.*;

public class MyWallpaper {
```

```
    public static void main(String[] argc) throws ParserConfigurationException, SAXException, IOException {
        MyWallpaper wallpaper = new MyWallpaper();

        do {
            String path = wallpaper.getThePath();
            wallpaper.downLoadWallpaper(path);
            wallpaper.settingWallpaper();
        } while (wallpaper.isConnect() != true);
    }

    public interface SPI extends StdCallLibrary {

        long SPI_SETDESKWALLPAPER = 20;
        long SPIF_UPDATEINIFILE = 0x01;
        long SPIF_SENDWININICHANGE = 0x02;

        SPI INSTANCE = (SPI) Native.loadLibrary("user32", SPI.class, new HashMap<Object, Object>() {
            {
                put(OPTION_TYPE_MAPPER, W32APITypeMapper.UNICODE);
                put(OPTION_FUNCTION_MAPPER, W32APIFunctionMapper.UNICODE);
            }
        });

        boolean SystemParametersInfo(UINT_PTR uiAction, UINT_PTR uiParam, String pvParam, UINT_PTR fWinIni);
    }

    public boolean isConnect() throws IOException {
        Runtime runtime = Runtime.getRuntime();
        Process process = runtime.exec("ping www.baidu.com");
        InputStream is = process.getInputStream();
        InputStreamReader isr = new InputStreamReader(is);
        BufferedReader br = new BufferedReader(isr);
        if (br.readLine() == null) {
            // System.out.println("The network is wrong!");
            return false;
        } else {
            // System.out.println("The network is well");
            return true;
        }
    }

    public String getThePath() throws ParserConfigurationException, SAXException, IOException {
        // getting the path of the bing jpg picture via analysis xml
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        DocumentBuilder builder = factory.newDocumentBuilder();
        Document document = builder.parse("http://www.bing.com/HPImageArchive.aspx?format=xml&idx=0&n=8");
        document.normalize();
        String relativePath = document.getElementsByTagName("url").item(0).getTextContent();
```

```
68          String path = "http://www.bing.com/" + relativePath;
69          return path;
70      }
71
72      public void downLoadWallpaper(String path) throws IOException {
73          // download the jpg file
74          path = path.replace("1366x768", "1920x1080");
75          URL url = new URL(path);
76          DataInputStream dis = new DataInputStream(url.openStream());
77          FileOutputStream fos = new FileOutputStream(new File("C:\\TEMP\\wallpaper.jpg"));
78          byte[] buffer = new byte[1024];
79          int length;
80          while ((length = dis.read(buffer)) > 0) {
81              fos.write(buffer, 0, length);
82          }
83
84          dis.close();
85          fos.close();
86      }
87
88      public void settingWallpaper() {
89          String localpath = "C:\\TEMP\\wallpaper.jpg";
90
91          SPI.INSTANCE.SystemParametersInfo(new UINT_PTR(SPI.SPI_SETDESKWALLPAPER), new
    UINT_PTR(0), localpath,
92                  new UINT_PTR(SPI.SPIF_UPDATEINIFILE | SPI.SPIF_SENDWININICHANGE));
93      }
94  }
```

2.3.2. 编译运行

```
1  # 编译，注意引用第三方jar的用法
2  "c:\Program Files\Java\jdk1.8.0_181\bin\javac" -Djava.ext.dirs="./libs" ./MyWallpaper.java
3  # 运行
4  java -Djava.ext.dirs="./libs" MyWallpaper
```

2.3.3. 打包jar

首先创建 `MANIFEST.MF`

```
1  Manifest-Version: 1.0
2  Created-By: 1.8.0_181 (Oracle Corporation)
3  Class-Path: libs\jna-4.2.0.jar libs\jna-platform-5.2.0.jar libs\gson-2.8.5.jar
4  Main-Class: MyWallpaper
5
```

> 注意：
>
> Class-Path: 和Main-Class: 后边都有一个空格，必须加上，否则会打包失败，错误提示为：Invalid header field；
>
> 写完Main-Class后一定要回车（即最后一行是空白行），让光标到下一行，否则最后一行内容打包时被吃掉了

```
$ "c:\Program Files\Java\jdk1.8.0_181\bin\jar" cvfm wallpaper.jar MANIFEST.MF
MyWallpaper.class MyWallpaper$SPI.class MyWallpaper$SPI$1.class

已添加清单
正在添加: MyWallpaper.class(输入 = 3279) (输出 = 1771)(压缩了 45%)
正在添加: MyWallpaper$SPI.class(输入 = 990) (输出 = 560)(压缩了 43%)
正在添加: MyWallpaper$SPI$1.class(输入 = 736) (输出 = 445)(压缩了 39%)

# 将jar和libs目录打包发布，运行代码
java -jar c:\Tools\BingWallpaper\wallpaper.jar
```

2.3.4. 使用IDE进行开发

使用vscode，eclipse都行

修改自动生成的 `.classpath`

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
    <classpathentry kind="con"
path="org.eclipse.jdt.launching.JRE_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/JavaSE-1.8"/>
    <classpathentry kind="src" path="src"/>
    <classpathentry kind="output" path="bin"/>
    <classpathentry kind="lib" path="libs/jna-4.2.0.jar"/>
    <classpathentry kind="lib" path="libs/jna-platform-5.2.0.jar"/>
    <classpathentry kind="lib" path="libs/gson-2.8.5.jar"/>
</classpath>
```

改使用json作为接口的协议，代码详见 `examples\BingWallpaper\`

2.3.5. 引入mvn

2.3.5.1. 安装maven

首先下载maven，然后把路径加入path，在vscode中通过从maven原型生成。

- groupId：com.test.common
- artifactId：wallpaper

2.3.5.2. 引入依赖

通过以下路径查询对应的Maven的引入办法，修改 `pom.xml`

- https://mvnrepository.com/artifact/net.java.dev.jna/jna-platform
- https://mvnrepository.com/artifact/net.java.dev.jna/jna
- https://mvnrepository.com/artifact/com.google.code.gson/gson

修改pom.xml后，vscode等会自动下载对应的依赖去MAVEN_HOME

2.3.5.3. 了解MAVEN_HOME

```
# 查看当前的MAVEN_HOME
mvn -v
```

MAVEN_HOME的定义，通过环境变量等方式都可以

2.3.5.4. maven的设置

```
<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <pluginGroups>
  </pluginGroups>
  <proxies>
  </proxies>
  <servers>
  </servers>
  <mirrors>
    <mirror>
      <id>alimaven</id>
      <name>aliyun maven</name>
      <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
      <mirrorOf>central</mirrorOf>
    </mirror>
  </mirrors>
  <profiles>
  </profiles>
</settings>
```

> 目前对我们比较关键的是设置mirror，可以高速下载

2.3.5.5. 代码修改运行

重新拷贝代码过来，可以运行了，如果使用了最新版本，可能会导致部分方法参数要修改，例如 `Native.loadLibrary` ，这也是开源软件常见的事情。

2.3.5.6. mvn常用命令

创建Maven的普通Java项目：

```
mvn archetype:create
    -DgroupId=packageName
    -DartifactId=projectName
```

创建Maven的Web项目：

```
mvn archetype:create
    -DgroupId=packageName
    -DartifactId=webappName
    -DarchetypeArtifactId=maven-archetype-webapp
```

编译源代码：

```
mvn compile
```

打包：

```
mvn package
```

只打jar包:

```
mvn jar:jar
```

查看当前项目已被解析的依赖:

```
mvn dependency:list
mvn dependency:tree
```

2.3.6. 引入单元测试

运行测试命令

```
mvn test
```

修改测试代码:

```
package com.test.common;

import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.regex.PatternSyntaxException;

import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;

/**
 * Unit test for simple App.
 */
public class AppTest extends TestCase {
    /**
     * Create the test case
     *
     * @param testName name of the test case
     */
    public AppTest(String testName) {
        super(testName);
    }

    /**
     * @return the suite of tests being tested
     */
    public static Test suite() {
        return new TestSuite(AppTest.class);
    }

    /**
     * Rigourous Test :-)
     */
```

```
    public void testApp() {
        try {
            String str = App.sendGETRequest("https://cn.bing.com/HPImageArchive.aspx?format=js&idx=0&n=1");
            JsonParser parser = new JsonParser();
            JsonObject object = (JsonObject) parser.parse(str);
            JsonArray array = object.get("images").getAsJsonArray();
            assertTrue(array.size() >= 1);
            JsonObject subObject = array.get(0).getAsJsonObject();
            String relativePath = subObject.get("url").getAsString();
            System.out.println(relativePath);
            boolean foundMatch = false;
            try {
                Pattern regex = Pattern.compile("/az/hprichbg/rb/.+");
                Matcher regexMatcher = regex.matcher(relativePath);
                foundMatch = regexMatcher.matches();
            } catch (PatternSyntaxException ex) {
                // Syntax error in the regular expression
            }
            assertTrue(foundMatch);
        } catch (Exception e) {
            e.printStackTrace();
            assertTrue(false);
        }
    }
}
```

再次运行测试

```
mvn test surefire-report:report
```

2.3.7. 发布应用

具体参考： http://maven.apache.org/plugins/maven-assembly-plugin/usage.html#Resources

2.3.7.1. Building an Assembly

修改 pom.xml

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>3.1.1</version>
            <configuration>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id> <!-- this is used for inheritance merges -->
                    <phase>package</phase> <!-- bind to the packaging phase -->
                    <goals>
```

```
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

2.3.7.2. Creating an Executable JAR

修改 pom.xml

```
<build>
    <plugins>
        <plugin>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>3.1.1</version>
            <configuration>
                <archive>
                    <manifest>
                        <mainClass>com.test.common.App</mainClass>
                    </manifest>
                </archive>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id> <!-- this is used for inheritance merges -->
                    <phase>package</phase> <!-- bind to the packaging phase -->
                    <goals>
                        <goal>single</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

2.3.7.3. 生成包含依赖可执行的jar

```
mvn package
```

2.3.8. 练习

独立完成上面介绍的练习，根据自身条件完成扩展功能。

3. JAVA案例：壁纸库

3.1. 数据源分析

`idx`：**索引**

http://cn.bing.com/HPImageArchive.aspx?idx=0&n=1 更换 idx= 这个参数

http://cn.bing.com/HPImageArchive.aspx?idx=1&n=1 将要得到昨天的图片

`n`：**图片数量**

http://cn.bing.com/HPImageArchive.aspx?idx=0&n=7 显示近一周的图片

3.2. 数据存储与文件存储

- 文件系统
  - linux
  - windows
- sql数据库
  - mysql
  - PostgreSQL
  - oracle
  - sqlite
- 结构化文本文件
  - xml
  - json
- nosql
  - mongodb
  - redis
  - Elasticsearch
  - Solr
- 云
  - 阿里云oss
  - opensearch

3.3. 数据结构设计

```
CREATE TABLE `wallpaper` (
  `id` int(11) NOT NULL,
  `title` varchar(200) NOT NULL,
  `url` varchar(200) NOT NULL,
  `copyright` varchar(200) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `wallpaper`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `url` (`url`);

ALTER TABLE `wallpaper`
  MODIFY `id` int(11) NOT NULL AUTO_INCREMENT;
```

3.4. 持久层操作（CRUD）

> CRUD是指在做计算处理时的增加(Create)、读取查询(Read)、更新(Update)和删除(Delete)几个单词的首字母简写。CRUD主要被用在描述软件系统中数据库或者持久层的基本操作功能。

3.4.1. JDBC

在mvn中追加mysql-connector-java： **https://mvnrepository.com/artifact/com.mysql.jdbc/com.springsource.com.mysql.jdbc/**

代码详见examples

3.4.2. ORM

ORM是对象和关系型数据库映射，是把Java中的JavaBean对象和数据库表进行映射，使数据库表中的记录和JavaBean对象一一对应，从而大大简化原来直接使用JDBC时，手工拼写SQL带来的不便。

ORM通过配置文件，使数据库表和JavaBean类对应起来，提供简便的操作方法，增、删、改、查记录，不再拼写字符串生成SQL，编程效率大大提高，同时减少程序出错机率，增强数据库的移植性，方便测试。

但是原生的JDBC具有更强的灵活性，适合复杂多变的SQL应用。

ORM底层也是用JDBC，ORM的关键在于解决对象和关系数据库的映射。所以ORM具有OO的优势，ORM也继承了OO的缺点。

3.4.2.1. 使用MyBatis Generator自动创建代码

首先配置 `generatorConfig.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
  PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
  "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">
<generatorConfiguration>
    <!-- 数据库驱动-->
    <classPathEntry location="c:\Users\lgc653\.m2\repository\mysql\mysql-connector-java\8.0.15\mysql-connector-java-8.0.15.jar"/>
    <context id="DB2Tables" targetRuntime="MyBatis3">
        <commentGenerator>
            <property name="suppressDate" value="true"/>
            <!-- 是否去除自动生成的注释 true: 是 :  false:否 -->
            <property name="suppressAllComments" value="true"/>
        </commentGenerator>
        <!--数据库链接URL，用户名、密码 -->
        <jdbcConnection driverClass="com.mysql.cj.jdbc.Driver" connectionURL="jdbc:mysql://localhost:3306/train?useUnicode=true&amp;characterEncoding=utf8&amp;serverTimezone=GMT%2B8&amp;useSSL=false&amp;nullCatalogMeansCurrent=true" userId="root" password="">
        </jdbcConnection>
        <javaTypeResolver>
            <property name="forceBigDecimals" value="false"/>
        </javaTypeResolver>
        <!-- 生成模型的包名和位置-->
        <javaModelGenerator targetPackage="com.test.common.domain" targetProject="src/main/java">
            <property name="enableSubPackages" value="true"/>
```

```
            <property name="trimStrings" value="true"/>
        </javaModelGenerator>
        <!-- 生成映射文件的包名和位置-->
        <sqlMapGenerator targetPackage="com.test.common.mapping"
targetProject="src/main/java">
            <property name="enableSubPackages" value="true"/>
        </sqlMapGenerator>
        <!-- 生成DAO的包名和位置-->
        <javaClientGenerator type="XMLMAPPER" targetPackage="com.test.common.IDao"
targetProject="src/main/java">
            <property name="enableSubPackages" value="true"/>
        </javaClientGenerator>
        <!-- 要生成的表 tableName是数据库中的表名或视图名 domainObjectName是实体类名-->
        <table tableName="wallpaper" domainObjectName="Wallpaper"
enableCountByExample="false" enableUpdateByExample="false" enableDeleteByExample="false"
enableSelectByExample="false" selectByExampleQueryId="false"></table>
    </context>
</generatorConfiguration>
```

然后执行生成命令，执行成需确认以下jar

- mybatis-generator-core
- mysql-connector-java
- mybatis

```
java -jar c:\Users\lgc653\.m2\repository\org\mybatis\generator\mybatis-generator-
core\1.3.7\mybatis-generator-core-1.3.7.jar -configfile generatorConfig.xml -overwrite
```

生成后会有以下问题：

Cannot obtain primary key information from the database, generated objects may be incomplete

> 参考 https://blog.csdn.net/jpf254/article/details/79571396 解决

Result Maps collection already contains value for com.test.common.IDao.WallpaperMapper.BaseResultMap

> 将自动生成的 `WallpaperMapper.xml` 中的重复定义部分删除

3.4.2.2. 创建配置文件mybatis-config.xml

当然也可以是其它名字，放在 `.classpath` 中定义的target/classes根目录

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC"></transactionManager>
            <dataSource type="POOLED">
                <property name="driver" value="com.mysql.jdbc.Driver"/>
```

```
                <property name="url" value="jdbc:mysql://localhost:3306/train?useUnicode=true&amp;characterEncoding=utf8&amp;serverTimezone=GMT%2B8&amp;useSSL=false" />
                <property name="username" value="root"/>
                <property name="password" value=""/>
            </dataSource>
        </environment>
    </environments>
    <mappers>
        <mapper resource="com/test/common/mapping/WallpaperMapper.xml" />
    </mappers>
</configuration>
```

> 注意mappers的定义

3.4.2.3. 编写测试文件

配置文件通过 `Resources.getResourceAsStream("mybatis-config.xml")` 来读取。

```
public void testUpdateWallpaper() throws IOException {
    System.out.println("更新一个用户");
    // 读取mybatis-config.xml文件
    InputStream resourceAsStream = Resources.getResourceAsStream("mybatis-config.xml");
    // 初始化mybatis,创建SqlSessionFactory类的实例
    SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(resourceAsStream);
    // 创建session实例
    SqlSession session = sqlSessionFactory.openSession();
    try {
        com.test.common.domain.Wallpaper wallpaper = new com.test.common.domain.Wallpaper();
        wallpaper.setId(1);
        wallpaper.setTitle("pkd888888");
        wallpaper.setUrl("pkd888888");
        wallpaper.setCopyright("pkd888888");
        int count = session.update("com.test.common.IDao.WallpaperMapper.updateByPrimaryKey", wallpaper);
        assertTrue(count == 1);
        // session.delete("com.test.common.IDao.WallpaperMapper.deleteByPrimaryKey",
        // wallpaper);
        session.commit();
        System.out.println(count);
    } finally {
        session.close();
    }
}
```

通过mvn命令进行测试

```
mvn test
```

4. 结语

知行合一，翻译为“The unity of Inner knowledge and action”——内在的知识和行动的统一。

Jeff Dean