### **一、背景**

软件测试过程中，最重要、最核心就是测试用例的设计，也是测试童鞋、测试团队日常投入最多时间的工作内容之一。  
然而，传统的测试用例设计过程有很多痛点：

1、使用Excel表格进行测试用例设计，虽然成本低，但版本管理麻烦，维护更新耗时，用例评审繁琐，过程报表统计难...

2、使用TestLink、TestCenter、Redmine等传统测试管理工具，虽然测试用例的执行、管理、统计比较方便，但依然存在编写用例效率不高、思路不够发散、在产品快速迭代过程中比较耗时等问题...

3、公司自研测试管理工具，这是个不错的选择，但对于大部分小公司、小团队来说，一方面研发维护成本高，另一方面对技术要有一定要求...

4、...

基于这些情况，现在越来越多公司选择使用思维导图这种高效的生产力工具进行用例设计，特别是敏捷开发团队。  
事实上也证明，思维导图其发散性思维、图形化思维的特点，跟测试用例设计时所需的思维非常吻合，所以在实际工作中极大提升了我们测试用例设计的效率，也非常方便测试用例评审。  
但是与此同时，使用思维导图进行测试用例设计的过程中也带来不少问题：

1、测试用例难以量化管理、执行情况难以统计；

2、测试用例执行结果与BUG管理系统难以打通；

3、团队成员用思维导图设计用例的风格各异，沟通成本巨大；

4、...

综合以上情况，我们可以发现不同的测试用例设计方式，各有各个的优劣。  
那么问题来了，我们能不能将它们各自优点合在一起呢？这样不就可以提升我们的效率了！  
于是，这时候 XMind2TestCase 就应运而生了，该工具基于 Python 实现，通过制定测试用例通用模板，  
然后使用 [XMind](https://www.xmind.cn/) 这款广为流传且开源的思维导图工具进行用例设计。  
其中制定测试用例通用模板是一个非常核心的步骤（具体请看[使用指南](https://github.com/zhuifengshen/xmind2testcase/blob/master/webtool/static/guide/index.md)），有了通用的测试用例模板，我们就可以在 XMind 文件上解析并提取出测试用例所需的基本信息，  
然后合成常见测试用例管理系统所需的用例导入文件。这样就将 XMind 设计测试用例的便利与常见测试用例系统的高效管理结合起来了！  
当前 XMind2TestCase 已实现从 XMind 文件到 TestLink 和 Zentao(禅道) 两大常见用例管理系统的测试用例转换，同时也提供 XMind 文件解析后的两种数据接口  
（TestSuites、TestCases两种级别的JSON数据），方便快速与其他测试用例管理系统打通。

### **二、使用示例**

#### **1、Web工具示例**

#### **2、转换后用例预览**

#### **3、TestLink导入结果示例**

### **三、安装方式**

pip3 install xmind2testcase

### **四、版本升级**

pip3 install -U xmind2testcase

### **五、使用方式**

#### **1、命令行调用**

Usage:  
xmind2testcase [path\_to\_xmind\_file] [-csv] [-xml] [-json]  
Example:  
xmind2testcase /path/to/testcase.xmind        => output testcase.csv、testcase.xml、testcase.json  
xmind2testcase /path/to/testcase.xmind -csv   => output testcase.csv  
xmind2testcase /path/to/testcase.xmind -xml   => output testcase.xml  
xmind2testcase /path/to/testcase.xmind -json  => output testcase.json

#### **2、使用Web界面**

web\_tool\_cli  
Usage:  
xmind2testcase [webtool] [port\_num]  
Example:  
xmind2testcase webtool        => launch the web testcase convertion tool locally -> 127.0.0.1:5001  
xmind2testcase webtool 8000   => launch the web testcase convertion tool locally -> 127.0.0.1:8000

#### **3、API调用**

import json  
import xmind  
from xmind2testcase.zentao import xmind\_to\_zentao\_csv\_file  
from xmind2testcase.testlink import xmind\_to\_testlink\_xml\_file  
from xmind2testcase.utils import xmind\_testcase\_to\_json\_file  
from xmind2testcase.utils import xmind\_testsuite\_to\_json\_file  
from xmind2testcase.utils import get\_xmind\_testcase\_list  
from xmind2testcase.utils import get\_xmind\_testsuite\_list

def main():  
    xmind\_file = 'docs/xmind\_testcase\_template.xmind'  
    print('Start to convert XMind file: %s' % xmind\_file)  
    zentao\_csv\_file = xmind\_to\_zentao\_csv\_file(xmind\_file)  
    print('Convert XMind file to zentao csv file successfully: %s' % zentao\_csv\_file)  
    testlink\_xml\_file = xmind\_to\_testlink\_xml\_file(xmind\_file)  
    print('Convert XMind file to testlink xml file successfully: %s' % testlink\_xml\_file)  
    testsuite\_json\_file = xmind\_testsuite\_to\_json\_file(xmind\_file)  
    print('Convert XMind file to testsuite json file successfully: %s' % testsuite\_json\_file)  
    testcase\_json\_file = xmind\_testcase\_to\_json\_file(xmind\_file)  
    print('Convert XMind file to testcase json file successfully: %s' % testcase\_json\_file)  
    testsuites = get\_xmind\_testsuite\_list(xmind\_file)  
    print('Convert XMind to testsuits dict data:\n%s' % json.dumps(testsuites, indent=2, separators=(',', ': '), ensure\_ascii=False))  
    testcases = get\_xmind\_testcase\_list(xmind\_file)  
    print('Convert Xmind to testcases dict data:\n%s' % json.dumps(testcases, indent=4, separators=(',', ': ')))  
    workbook = xmind.load(xmind\_file)  
    print('Convert XMind to Json data:\n%s' % json.dumps(workbook.getData(), indent=2, separators=(',', ': '), ensure\_ascii=False))  
    print('Finished conversion, Congratulations!')

if \_\_name\_\_ == '\_\_main\_\_':  
    main()

#### **4、XMind用例文件转为JSON数据**

xmind\_testcase\_demo

##### **（1）转为测试用例JSON数据**

from xmind2testcase.utils import get\_xmind\_testcase\_list  
xmind\_file = 'docs/xmind\_testcase\_demo.xmind'  
testcases = get\_xmind\_testcase\_list(xmind\_file)  
print(testcases)

Output:  
[  
    {                                                # 测试用例  
        "name": "测试用例1",                           # 用例标题  
        "version": 1,                                 # 用例版本  
        "summary": "测试用例1",                        # 用例摘要  
        "preconditions": "前置条件",                   # 前置条件  
        "execution\_type": 1,                          # 用例执行类型（1：手动、2：自动）  
        "importance": 1,                              # 优先级（1：高、2：中、3：低）  
        "estimated\_exec\_duration": 3,                 # 预计执行时间（分钟）  
        "status": 7,                                  # 用例状态（1：草稿、2：待评审、3：评审中、4：重做、5、废弃、6：feature、7：终稿）  
        "steps": [                                    # 测试步骤列表  
            {  
                "step\_number": 1,                     # 编号  
                "actions": "测试步骤1",                 # 步骤内容  
                "expectedresults": "预期结果1",         # 预期结果  
                "execution\_type": 1                    # 执行类型（1：手动，2：自动）  
            },  
            {  
                "step\_number": 2,  
                "actions": "测试步骤2",  
                "expectedresults": "预期结果2",  
                "execution\_type": 1  
            }  
        ],  
        "product": "我是产品名",                          # 产品名称  
        "suite": "我是模块名(测试集1)"                     # 测试集（模块名）  
    },  
    {  
        "name": "测试用例2",  
        "version": 1,  
        "summary": "测试用例2",  
        "preconditions": "前置条件",  
        "execution\_type": 1,  
        "importance": 1,  
        "estimated\_exec\_duration": 3,  
        "status": 7,  
        "steps": [  
            {  
                "step\_number": 1,  
                "actions": "测试步骤1",  
                "expectedresults": "预期结果1",  
                "execution\_type": 1  
            },  
            {  
                "step\_number": 2,  
                "actions": "测试步骤2（预期结果2可以为空）",  
                "expectedresults": "",  
                "execution\_type": 1  
            },  
            {  
                "step\_number": 3,  
                "actions": "测试步骤3",  
                "expectedresults": "预期结果3",  
                "execution\_type": 1  
            },  
            {  
                "step\_number": 4,  
                "actions": "测试步骤4",  
                "expectedresults": "预期结果4",  
                "execution\_type": 1  
            }  
        ],  
        "product": "我是产品名",  
        "suite": "我是模块名(测试集1)"  
    },  
    {  
        "name": "测试用例3（测试步骤和预期结果可以都为空）",  
        "version": 1,  
        "summary": "测试用例3（测试步骤和预期结果可以都为空）",  
        "preconditions": "无",  
        "execution\_type": 1,  
        "importance": 2,  
        "estimated\_exec\_duration": 3,  
        "status": 7,  
        "steps": [ ],  
        "product": "我是产品名",  
        "suite": "我是模块名(测试集1)"  
    },  
    {  
        "name": "测试步骤2（优先级默认为中）",  
        "version": 1,  
        "summary": "测试步骤2（优先级默认为中）",  
        "preconditions": "无",  
        "execution\_type": 1,  
        "importance": 3,  
        "estimated\_exec\_duration": 3,  
        "status": 7,  
        "steps": [  
            {  
                "step\_number": 1,  
                "actions": "测试步骤1",  
                "expectedresults": "预期结果1",  
                "execution\_type": 1  
            },  
            {  
                "step\_number": 2,  
                "actions": "测试步骤3",  
                "expectedresults": "",  
                "execution\_type": 1  
            }  
        ],  
        "product": "我是产品名",  
        "suite": "我是模块名(测试集2)"  
    },  
    {  
        "name": "测试用例3（前置条件默认为空） 无设置优先级，这里加入用例标题",  
        "version": 1,  
        "summary": "测试用例3（前置条件默认为空） 无设置优先级，这里加入用例标题",  
        "preconditions": "无",  
        "execution\_type": 1,  
        "importance": 2,  
        "estimated\_exec\_duration": 3,  
        "status": 7,  
        "steps": [ ],  
        "product": "我是产品名",  
        "suite": "我是模块名(测试集2)"  
    }  
]

##### **（2）转为测试集JSON数据**

from xmind2testcase.utils import get\_xmind\_testsuite\_list  
xmind\_file = 'docs/xmind\_testcase\_demo.xmind'  
testsuites = get\_xmind\_testsuite\_list(xmind\_file)  
print(testsuites)

Output:  
[  
  {                                                 # XMind画布（Sheet)列表  
    "name": "我是产品名",                             # 产品名称  
    "details": null,                                 # 产品摘要  
    "testcase\_list": [],                             # 用例列表  
    "sub\_suites": [                                  # 用例集列表  
      {  
        "name": "我是模块名(测试集1)",                  # 用例集1名称（模块名）  
        "details": null,                             # 用例集摘要  
        "testcase\_list": [                           # 用例列表  
          {                                          # 具体用例  
            "name": "测试用例1",  
            "version": 1,  
            "summary": "测试用例1",  
            "preconditions": "前置条件",  
            "execution\_type": 1,  
            "importance": 1,  
            "estimated\_exec\_duration": 3,  
            "status": 7,  
            "steps": [  
              {  
                "step\_number": 1,  
                "actions": "测试步骤1",  
                "expectedresults": "预期结果1",  
                "execution\_type": 1  
              },  
              {  
                "step\_number": 2,  
                "actions": "测试步骤2",  
                "expectedresults": "预期结果2",  
                "execution\_type": 1  
              }  
            ]  
          },  
          {  
            "name": "测试用例2",  
            "version": 1,  
            "summary": "测试用例2",  
            "preconditions": "前置条件",  
            "execution\_type": 1,  
            "importance": 1,  
            "estimated\_exec\_duration": 3,  
            "status": 7,  
            "steps": [  
              {  
                "step\_number": 1,  
                "actions": "测试步骤1",  
                "expectedresults": "预期结果1",  
                "execution\_type": 1  
              },  
              {  
                "step\_number": 2,  
                "actions": "测试步骤2（预期结果2可以为空）",  
                "expectedresults": "",  
                "execution\_type": 1  
              },  
              {  
                "step\_number": 3,  
                "actions": "测试步骤3",  
                "expectedresults": "预期结果3",  
                "execution\_type": 1  
              },  
              {  
                "step\_number": 4,  
                "actions": "测试步骤4",  
                "expectedresults": "预期结果4",  
                "execution\_type": 1  
              }  
            ]  
          },  
          {  
            "name": "测试用例3（测试步骤和预期结果可以都为空）",  
            "version": 1,  
            "summary": "测试用例3（测试步骤和预期结果可以都为空）",  
            "preconditions": "无",  
            "execution\_type": 1,  
            "importance": 2,  
            "estimated\_exec\_duration": 3,  
            "status": 7,  
            "steps": []  
          }  
        ],  
        "sub\_suites": []                            # 用例集中可以包含子用例集（目前只要产品类别下有用例集）  
      },  
      {  
        "name": "我是模块名(测试集2)",                  # 用例集2名称（模块名）  
        "details": "测试集摘要（详情）",  
        "testcase\_list": [  
          {  
            "name": "测试步骤2（优先级默认为中）",  
            "version": 1,  
            "summary": "测试步骤2（优先级默认为中）",  
            "preconditions": "无",  
            "execution\_type": 1,  
            "importance": 3,  
            "estimated\_exec\_duration": 3,  
            "status": 7,  
            "steps": [  
              {  
                "step\_number": 1,  
                "actions": "测试步骤1",  
                "expectedresults": "预期结果1",  
                "execution\_type": 1  
              },  
              {  
                "step\_number": 2,  
                "actions": "测试步骤3",  
                "expectedresults": "",  
                "execution\_type": 1  
              }  
            ]  
          },  
          {  
            "name": "测试用例3（前置条件默认为空） 无设置优先级，这里加入用例标题",  
            "version": 1,  
            "summary": "测试用例3（前置条件默认为空） 无设置优先级，这里加入用例标题",  
            "preconditions": "无",  
            "execution\_type": 1,  
            "importance": 2,  
            "estimated\_exec\_duration": 3,  
            "status": 7,  
            "steps": []  
          }  
        ],  
        "sub\_suites": []  
      }  
    ]  
  }  
]

##### **（3）XMind文件转换为JSON数据**

以上（1）TestCase数据、（2）TestSuite数据的获取，其实是基于[XMind](https://github.com/zhuifengshen/xmind)这个工具，对XMind文件进行解析和数据提取，然后转换而来。  
这个工具是在设计XMind2TestCase时，针对XMind单独抽取出来的库，提供了XMind思维导图创建、解析、更新的一系列方法。使用它可以直接将XMind文件转换为JSON数据：  
import xmind  
xmind\_file = 'docs/xmind\_testcase\_demo.xmind'  
workbook = xmind.load(xmind\_file)  
data = workbook.getData()  
print(data)

Output:  
[  
  {                                                    # XMind画布(sheet)列表  
    "id": "7hmnj6ahp0lonp4k2hodfok24f",                # 画布ID  
    "title": "画布 1",                                  # 画布名称  
    "topic": {                                         # 中心主题  
      "id": "7c8av5gt8qfbac641lth4g1p67",              # 主题ID  
      "link": null,                                    # 主题上的超链接信息  
      "title": "我是产品名",                             # 主题名称  
      "note": null,                                    # 主题上的备注信息  
      "label": null,                                   # 主题上标签信息  
      "comment": null,                                 # 主题上的批注（评论）信息  
      "markers": [],                                   # 主题上的图标信息  
      "topics": [                                      # 子主题列表  
        {  
          "id": "2rj4ek3nn4sk0lc4pje3gvgv9k",  
          "link": null,  
          "title": "我是模块名(测试集1)",                  # 子主题1  
          "note": null,  
          "label": null,  
          "comment": null,  
          "markers": [],  
          "topics": [                                    # 子主题下的子主题列表  
            {  
              "id": "3hjj43s7rv66uncr1srl3qsboi",  
              "link": null,  
              "title": "测试用例1",  
              "note": "前置条件\n",  
              "label": "手动（执行方式默认为手动）",  
              "comment": null,  
              "markers": [  
                "priority-1"  
              ],  
              "topics": [  
                {  
                  "id": "3djn37j1fdc6081de319slf035",  
                  "link": null,  
                  "title": "测试步骤1",  
                  "note": null,  
                  "label": null,  
                  "comment": null,  
                  "markers": [],  
                  "topics": [  
                    {  
                      "id": "7v0f1152popou38ndaaamt49l5",  
                      "link": null,  
                      "title": "预期结果1",  
                      "note": null,  
                      "label": null,  
                      "comment": null,  
                      "markers": []  
                    }  
                  ]  
                },  
                {  
                  "id": "2srtqqjp818clkk1drm233lank",  
                  "link": null,  
                  "title": "测试步骤2",  
                  "note": null,  
                  "label": null,  
                  "comment": null,  
                  "markers": [],  
                  "topics": [  
                    {  
                      "id": "4jlbo280urmid3qkd01j7h8jnq",  
                      "link": null,  
                      "title": "预期结果2",  
                      "note": null,  
                      "label": null,  
                      "comment": null,  
                      "markers": []  
                    }  
                  ]  
                }  
              ]  
            },  
            ...  
          ]  
        },  
        ...  
      ]  
    }  
  }  
]

具体参考：[xmind\_testcase\_demo.json](https://github.com/zhuifengshen/xmind2testcase/blob/master/docs/xmind_testcase_demo.json)

### **四、自动化发布：一键打 Tag 并上传至 PYPI**

每次在 \_\_ about \_\_.py 更新版本号后，运行以下命令，实现自动化更新打包上传至 [PYPI](https://pypi.org/) ，同时根据其版本号自动打 Tag 并推送到仓库：  
python3 setup.py pypi