# Testflow平台的开发说明

Testflow的开发从功能上分为两部分：Testflow框架和Testflow软件。

Testflow的开发基于依赖倒置的范式：首先定义Testflow框架内部和对外的接口，然后框架和软件可以同时基于这些定义的接口各自进行上层的开发：框架开发时需要实现接口定义的功能，软件开发时，需要依赖接口定义的组件实现功能。最终两个部分通过TestflowRunner作为实例化的接口，实现功能的对接。

# Testflow框架的开发设计

Testflow框架需要完成的组件主要包括：Testflow框架组件、Testflow设计时组件、Testflow运行时组件、TestflowRunner、Testflow用户组件接口及内置组件。

开发的顺序为：Testflow框架组件、Testflow用户组件接口->Testflow设计时组件、Testflow运行时组件、Testflow内置组件->TestflowRunner相关。Testflow软件可独立开发。

## Testflow框架组件

Testflow框架组件提供最底层的功能实现，主要包括：

* 日志模块，模块控制接口为Testflow.Modules.ILogService，该模块可能被所有框架接口和用户组件依赖。
* 组件接口管理模块，模块控制接口为Testflow.Modules.IComInterfaceManager
* 序列管理模块，模块控制接口为Testflow.Modules.ISequenceManager
* 参数检查模块，模块控制接口为Testflow.Modules.IParameterChecker
* 引擎控制模块，模块控制接口为Testflow.Modules.IEngineController
* 数据维护模块，模块控制接口为Testflow.Modules.IDataMaintainer，该接口为引擎模块和结果管理模块依赖(因为涉及到具体细节，暂时还没定义完成)
* 结果管理模块，模块接口为Testflow.Modules.IResultManager，该接口涉及到细节，暂时还未定义完成。

在框架组件中除了需要实现模块控制接口，额外需要实现的接口包括：

|  |  |  |
| --- | --- | --- |
| 接口 | 关联模块 | 说明 |
| Testflow.Data.Sequence命名空间里所有接口 | 序列管理模块 | 所有和序列数据相关的接口定义 |
| Testflow.Data.IAssemblyInfo  Testflow.Data.IAssemblyInfoCollection  Testflow.Data.ITypeData | 序列管理模块 | 组件和类型的相关信息 |
| Testflow.Data.Description所有接口 | 组件接口管理模块 | 所有和组件描述信息相关的接口定义 |

Testflow框架组件原则上没有开发顺序的依赖，但是因为数据维护模块和引擎控制模块、数据维护模块和结果管理模块之间关联较大且其中两个的具体接口互相依赖，所以这三个需要按照引擎控制模块->数据维护模块->结果管理模块的顺序开发。

各个模块需要完成的功能及实现说明如下所述：

### 组件接口模块

组件接口加载模块提供加载组件接口描述的功能，维护当前所有被加载的描述接口和索引号的映射。使用Assembly加载程序集时最好在另外的AppDomain执行(如果加载到当前的AppDomain会无法即时卸载，会严重增加整个应用的资源占用)。

需要实现的方法及说明如下：

* GetComInterfaceById：根据ID号获取对应的组件接口描述信息，如果没有返回null
* GetComponentInterface：根据路径或者程序集信息获取组件接口描述信息，如果没有返回null
* GetComponentInterfaces：根据路径集合和程序集信息集合获取一组组件接口描述信息，返回的List长度和路径集合长度相同，如果某个未加载成功，对应位置返回null
* GetPropertyType：根据ITypeData和属性名获取对应属性的类型信息。
* GetTypeProperties：获取某个类型的所有属性名

接口描述信息需要包括的内容如下，这些内容和接口定义一一对应：

* 该组件接口在全局的唯一ID
* 组件的签名，该字符串格式为“程序集名称-版本号”
* 程序集信息
  + 程序集名称
  + 程序集描述信息
  + 程序集所在路径，包括文件名
  + 程序集加载是否成功
  + 程序集版本号
* 类接口列表
  + 类名
  + 类描述信息
  + 类所在组件接口的全局索引号
  + 类在当前组件中唯一的类ID
  + 类的类型信息
    - 类的类型信息
    - 所在程序集名称
    - 命名空间
    - 类名
  + 方法接口列表
    - 方法名
    - 方法描述信息
    - 方法类型(构造方法/实例方法/静态方法)
    - 方法所在组件的接口信息的全局索引
    - 类在组件的ID号
    - 是否泛型方法
    - 方法返回值信息
    - 入参信息列表
    - 方法签名，格式为“类名.方法名(入参类型列表)”

### 序列管理模块

序列管理模块实现对序列的定义、获取序列各个类型数据实例和序列化/反序列化操作。其功能参照ISequenceManager接口的定义。

SequenceManager的部分方法涉及到可变长参数(接口定义中以param命名的参数)，这些参数及其索引号的定义如下。

|  |  |  |
| --- | --- | --- |
| 方法名 | 参数索引 | 功能说明 |
| LoadTestProject  LoadSequenceGroup | 0 | source参数为File: 读取的文件路径 |
| 1 | 哈希值不一致时是否强制读取，默认为“False”，强制读取时为”True”。 |

### 引擎控制模块

### 参数检查模块

参数检查模块根据组件接口模块的接口描述信息，检查当前的所有序列参数的配置情况，需要执行的操作包括：

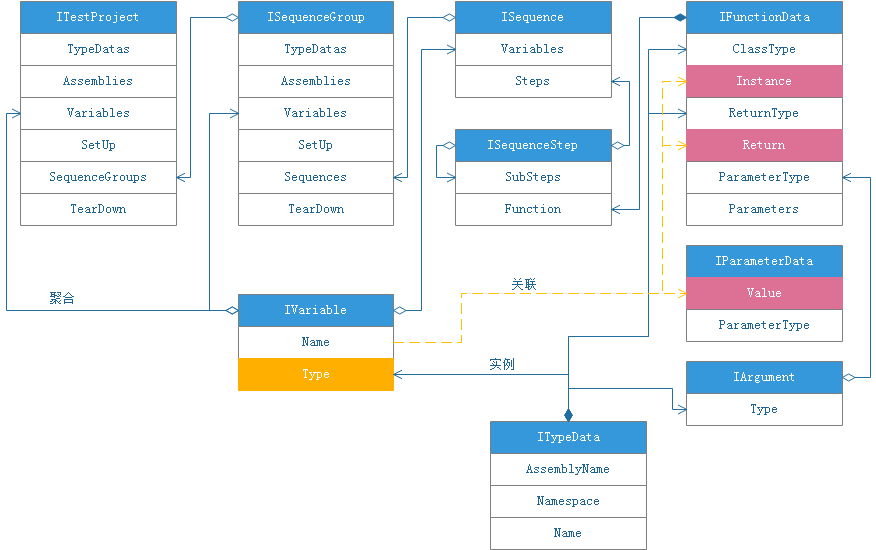
* 检查每个测试序列组的Argument入参有没有被成功配置。
* 检查每个测试序列组的参数配置和返回值配置类型是否匹配
* 检查变量在使用前是否被初始化且类型在整个序列中是否匹配。

该模块可以检查TestProject和SequenceGroup的配置，返回值为IWarningInfo的集合，每个IWarningInfo的实例对应一处检查错误。一个WarningInfo包含的信息有：

* 错误出现的SequenceGroup实例
* 错误出现的Sequence实例
* 错误出现的SequenceStep实例
* 错误信息
* 错误码

同时该模块应该对常见的错误类型/错误信息/错误码进行定义，且生成公共的文档。这些错误定义始终应当是向下兼容(更新错误定义后原来的定义不能被修改)。

参数检查涉及到的数据结构和关系图如下所示：



类型和属性间关系的说明如下：聚合关系表示线型箭头指向的属性是由菱形箭头指向类型的多个实例组成；实例关系表示线型箭头指向的属性是菱形箭头指向的接口类型的实例；关联关系表明被线型指向的属性实际上可能对应箭头起始方向类型的属性。

参数检查模块需要执行的操作以颜色块标记。黄色属性表示在参数检查过程中可能需要初始化的属性；红色属性表示需要校验的属性。

IVarialble的Name为变量名称(字符串)，Type为ITypeData类型的变量。在未初始化时所有变量的Type都为null。在参数检查的过程中，如果Type为null，则该模块在遍历该变量的定义范围时应当检查所有IFunctionData中的Instance和Return属性、IParameterData中的Value属性(只有在ParameterType为Variable时Value才代表变量的名称)，一旦发现该变量名被使用，则初始化对应Variable的Type属性：如果是在Instance中使用，则赋值为FunctionData的ClassType；如果在Return中使用，则赋值为FunctionData的ReturnType；如果在IParameterData的Value中被使用，则获取该IParameter对象在集合中的索引号，在对应索引的FunctionData.ParameterType中的Type对象赋值给Variable的Type对象。

执行检查时，需要依次遍历所有TestProject->SequenceGroup->Sequence->SequenceStep下的FunctionData。首先检查所有的Instance、Return(如果不存在则无需检查)、Parameters是否被配置。然后依次检查Instance对应的变量类型是否和定义的变量类型是否相同、Return对应的变量类型是否和定义的变量类型相同、所有ParameterData(如果ParameterType为Variable)对应的变量类型是否和对应索引号的ParameterType中的Type相同。

ITypeData方法提供Equals类型实现，可以直接调用if(typeData1.Equals(typeData2))执行判断，如果检查失败则创建WarnInfo实例，添加到WarnInfo集合中。待检查结束后将该集合返回。

### 数据维护模块

### 结果管理模块

## Testflow设计时组件

设计时服务提供序列编辑服务，这些功能承载于底层组件之上，实现测试序列的直接设计与增删操作。设计时组件需要实现Testflow.DesignTime命名空间下的所有接口，主要包括：

* IDesignTimeService：设计时服务，完成设计时TestProject级别的控制和功能管理
* IDesignTimeSession：单个测试序列组的设计时服务，完成设计时SequenceGroup级别的控制和功能管理
* IDesignTimeContext：单个DesignTimeSession的上下文信息。

## Testflow运行时组件

运行时服务提供序列运行服务，这些功能承载于底层组件之上，实现测试序列组的运行控制、运行时调试、状态维护、结果处理。运行时服务需要实现Testflow.Runtime命名空间下的所有接口，主要的三个类包括：

* IRunTimeService：运行时服务，完成 时TestProject级别的控制、状态维护和结果处理。
* IRunTimeSession：单个测试序列组的运行时服务，完成运行时SequenceGroup级别的控制、状态维护。
* IRunTimeContext：单个RunTimeSession的上下文信息。

## TestflowRunner

Testflow平台运行器，维护一个Testflow运行实例的所有功能的实例，包括所有组件和服务的创建/获取/销毁操作。该类为抽象单例类，在单个AppDomain中通过GetInstance方法可以获取唯一的实例，该方法的入参是TestflowRunnerOptions参数类，用以配置平台参数。该类中包括的主要属性有：

|  |  |
| --- | --- |
| 属性名 | 说明 |
| RuntimeService | 运行时服务 |
| DesignTimeService | 设计时服务 |
| ComInterfaceLoader | 组件接口加载器 |
| DataMaintainer | 数据维护组件，实现数据的持久化等功能 |
| EngineController | 运行引擎控制器 |
| ParameterChecker | 参数校验组件控制 |
| ResultManager | 结果管理组件 |
| SequenceManager | 序列管理组件 |
| LogService | 日志服务 |
| I18n | 国际化组件 |

该类需要依赖所有已实现的组件，所以GetInstance方法需要待所有组件和服务开发完成后实现。

## Testflow用户组件接口

Testflow用户组件接口定义运行引擎执行流程和Testflow软件面板可能需要的信息，需要待框架功能模块中持续整理，待框架功能模块开发完成后再添加。

## Testflow内置组件

Testflow内置组件提供多个接口，便于Testflow框架实现对组件的控制，目前需要实现的模块包括

# Testflow框架的功能规划

留待下个版本实现的功能：

* Testflow流程支持条件分支

# Testflow软件的设计规划

Testflow软件需要完成的组件主要包括：

* 序列管理面板：测试工程和序列组的管理、创建、打开界面组件
* 设计时面板：组件库加载相关界面组件、序列编辑相关UI界面、参数配置界面、变量声明界面、Output界面(用于显示参数检查和用户操作等相关信息)、
* 运行时面板：操作员界面组件、调试界面组件、实时状态管理界面组件
* 结果管理面板：数据库中的结果管理界面组件、报表生成和管理界面组件。