# Testflow框架接口说明

Testflow框架的接口是一组用于框架/软件开发和用户使用的接口规范，实现对Testflow整套业务流程的控制和管理。Testflow框架接口功能上主要有两类：Testflow框架功能的控制接口；用户自定义组件接口。

Testflow框架的功能控制接口提供了从创建框架平台运行实例到完成测试被释放的整个流程的控制接口，主要包括的接口功能有：框架的配置管理、测试序列数据的管理、框架模块的管理、软件设计时的功能管理、软件运行时的功能管理、结果统计与维护的底层支持。该类接口由一组互相关联的Interface/Enumeration/AbstractClass组成。Testflow框架的功能控制接口维护在解决方案的Dev/Common工程中。

用户自定义组件接口是用户在创建自定义组件时使用，便于在Testflow框架中实现组件的类型管理和同步控制等。该组接口由一组互相关联的Interface和Attribute组成。该接口的实现时非强制性的。用户自定义组件接口维护在解决方案的Dev/UserLibInterface工程中。

# 主要框架接口和流程说明

框架的主要功能使用功能控制接口完成。本章节重点说明核心的框架结构和在不同的应用场景下使用框架的流程。

## 框架功能控制接口的说明

Testflow平台中数据和功能互相独立。功能组件基于功能分层的结构开发：最底层的功能由各自独立的模块提供；基于这些底层模块，上层的设计时和运行时管理组件提供应用层的控制接口；在设计时和运行时接口以上，Testflow软件可以提供可视化的测试序列的设计与运行管理。

Testflow底层平台入口为TestflowRunner抽象单例类，该类通过TestflowRunnerOptions选项配置。通过该类可以访问和控制所有和平台相关的组件，包括模块控制接口和设计时/运行时控制接口。

模块控制接口提供所有模块相关的底层功能。这些接口主要为框架开发时定义，软件面板对平台的操作主要以设计时控制和运行时控制的接口完成。

设计时接口主要提供设计序列相关的功能，包括创建序列、编辑序列、编辑组件等功能。运行时接口主要提供测试序列运行的相关功能，包括运行、状态监视、Step调试、报表生成等工作。

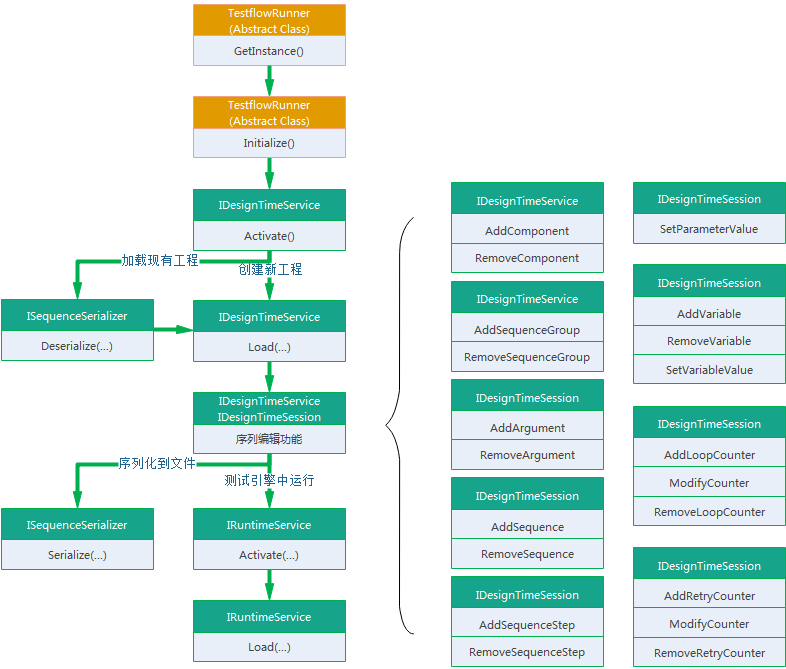
数据接口维护的信息包括：序列相关信息及对应参数配置信息、中间过程的状态信息、调试信息、执行结束的统计信息。其中接口描述信息提供设计时的序列编辑支持。序列流程信息保存用户编辑的所有序列信息，序列参数配置信息保存对应序列的参数配置，这些配置数据和序列信息相对独立。其他的状态和统计信息都由运行时维护，不能独立使用。

这些框架功能模块及各个模块中主要接口的组合关系图如下文件所示。



## Testflow框架的设计时流程

使用框架执行序列设计的接口流程如下图所示：



设计时的工作相关的类主要有三个，分别是：

* IDesigntimeService：设计时服务，承载所有设计时业务和TestProject级的设计时控制。该接口的SequenceSessions(IDesigntimeSession类型)维护了所有序列组的设计时会话。该接口是设计时的入口。
* IDesigntimeSession：设计时会话，承载单个SequenceGroup的设计时控制，所有的相关数据保存在Context属性(IDesigntimeContext类型)中。
* IDesigntimeContext：设计时上下文，承载单个SequenceGroup的设计上下文信息。所有SequenceGroup相关的数据和状态都可以从该对象中获取。

使用IDesigntimeService前需要初始化，该操作在Runner.Initialize中会连带完成。在切到设计时时需要激活设计时服务，这时可以通过调用空参的Load方法创建新的TestProject，或者通过ISequenceSerializer加载已有的TestProject或SequenceGroup(会创建只包含当前SequenceGroup的TestProject)。

打开TestProject后进入设计过程。可执行的操作包括：

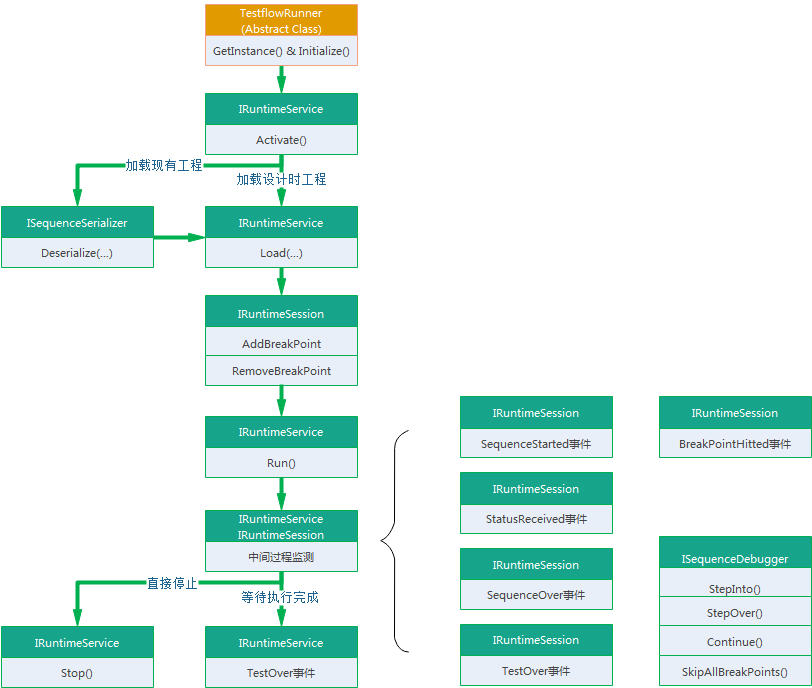
* 调用IDesigntimeService的AddComponent和RemoveComponent方法加载或卸载组件库到当前工程。这些组件在整个TestProject内都可以访问。
* 调用IDesigntimeService的AddSequenceGroup和RemoveSequenceGroup方法添加或删除已有的/新建的序列组文件。
* 调用IDesigntimeSession的AddArgument/RemoveArgument，对整个序列组添加或删除入参声明。
* 调用IDesigntimeSession的AddSequence/RemoveSequence，在测试序列组添加或删除测试序列。
* 调用IDesigntimeSession的AddSequenceStep和RemoveSequenceStep方法对测试序列/测试序列Step添加测试序列步骤。
* 调用IDesigntimeSession的SetParameterValue方法，配置单个Step的参数值。
* 调用IDesigntimeSession的AddVariable/RemoveVariable/SetVariableValue方法，分别对序列组/序列的Variable执行增加/删除/配置初始值的操作。
* 调用IDesigntimeSession的AddLoopCounter/ModifyCounter/RemoveLoopCounter方法，对某个Step执行增加/配置/删除循环计数器的操作。
* 调用IDesigntimeSession的AddRetryCounter/ModifyCounter/RemoveRetryCounter方法，对某个Step执行增加/配置/删除重试计数器的操作。

设计完成后可以有两种操作：

* 调用ISequenceSerializer方法，将设计完成的测试序列保存到文件中。
* 调用IRuntimeService的相关方法，将配置完成的序列组加载到运行引擎中执行。

## Testflow框架的运行时流程

使用框架执行测试序列运行的接口流程如下图所示：



运行时的工作相关的类主要有三个，分别是：

* IRuntimeService：设计时服务，承载所有运行时业务和TestProject级的运行时控制。该接口的Sessions(IRuntimeSession类型)维护了所有序列组的运行时会话。该接口是运行时的入口。
* IRuntimeSession：运行时会话，承载单个SequenceGroup的运行时控制，所有的相关数据保存在Context属性(IRuntimeContext类型)中。
* IRuntimeContext：运行时上下文，承载单个SequenceGroup的运行上下文信息。所有SequenceGroup相关的数据和状态都可以从该对象中获取。

使用IRuntimeService前需要初始化，该操作在Runner.Initialize中会连带完成。在切到运行时的时候需要激活运行时服务。然后调用Load方法加载待执行的序列，可以直接加载设计时正在操作的序列，或者使用ISequenceSerializer模块加载已有的测试工程/序列组到运行时。

用户可以配置RuntimeService的Configuration.Type属性，修改运行引擎为调试模式还是运行模式，其中**运行模式执行效率更高，调试模式支持Step级别的调试，并且调试模式下，测试序列组的所有测试序列只能串行执行**。如果选择运行模式，则可以调用各个序列组的IRuntimeSession的AddBreakPoint/RemoveBreakPoint方法，在任意Step上添加断点。

在完成以上操作后，执行IRuntimeService.Run方法开始执行测试。在运行过程中，所有的运行时状态都由事件获取，调试会提供专门的调试接口。在运行过程中的主要操作包括：

* IRuntimeSession的SequenceStarted事件：某个序列开始运行时触发该事件。
* IRuntimeSession的StatusReceived事件：获取运行引擎传递的运行时状态报告时触发该事件。
* IRuntimeSession的SequenceOver事件：某个序列执行结束后触发该事件，事件参数的报告中会包含这个序列执行的结果信息、关键信息和统计信息。
* IRuntimeSession的TestOver事件：整个测试序列组执行结束后触发该事件，事件参数的报告会包含整个序列组所有序列的执行统计信息和各个序列的详尽报告。
* IRuntimeSession的BreakPointHitted事件：触发到单个断点时触发该事件，事件参数中会包含当前断点下，所有变量的值信息。
* ISequenceDebugger的StepInto/StepOver/Continue/SkipAllBreakPoints方法，分别用来控制断点命中后的执行行为，包括进入子Step/进入下个同级Step的调试/继续执行直到下个断点/跳过当前序列所有断点执行。

在整个序列工程执行完成后会触发IRuntimeSevice的TestOver事件，该事件中包含了当前 测试工程下所有测试结果的统计信息。在执行过程中随时可以用IRuntimeService.Stop方法中断测试程序的执行。

# Testflow框架的功能控制接口

框架功能控制接口主要有7类，每个类别的功能说明如下表所示：

|  |  |
| --- | --- |
| 接口类别 | 功能说明 |
| 启动模块 | 创建框架平台实例、管理所有其他模块和功能。 |
| 模块控制 | 提供框架各个模块实例的直接控制 |
| 设计时控制 | 提供Testflow在序列设计时的相关服务和功能 |
| 运行时控制 | 提供Testflow在运行时的相关服务和功能 |
| 接口描述信息 | 加载和维护各个可用组件的接口描述信息 |
| 序列流程信息 | 维护序列流程的相关配置信息 |
| 序列参数配置信息 | 维护对应序列流程的参数配置数据 |
| 公共接口 | 提供框架中通用的接口 |

每个模块下各自的接口定义如下图所示。

## 启动模块接口定义

启动模块用于实现对框架的创建于初始化，主要包括两个接口类的定义，每个类的定义如下所示：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| TestflowRunner | AbstractClass | 创建和初始化平台运行实例，维护各个模块和功能的接口。单例类 |
| TestflowRunnerOptions | Class | 框架平台运行实例的参数配置 |

### TestflowRunner

FlowRunner的接口定义如下：

|  |  |  |
| --- | --- | --- |
| 静态方法 | | |
| 方法签名 | | 说明 |
| FlowRunner GetInstance(FlowRunnerOptions options) | | 获取或创建一个框架平台的运行实例。 |
| 属性 | | |
| 属性名 | 类型 | 说明 |
| RuntimeService | IRuntimeService | 运行时服务控制类 |
| DesignTimeService | IDesignTimeService | 设计时服务控制类 |
| ComInterfaceLoader | IComInterfaceLoader | 组件接口加载模块控制类 |
| ConfigurationManager | IConfigurationManager | 配置管理模块控制类 |
| DataPersistance | IDataPersistance | 数据持久化模块控制类 |
| EngineController | IEngineController | 引擎控制模块控制类 |
| LogService | ILogService | 日志服务模块控制类 |
| ParameterChecker | IParameterChecker | 参数检查模块控制类 |
| ResultManager | IResultManager | 结果管理模块控制类 |
| SequenceSerializer | ISequenceSerializer | 序列化模块控制类 |
| I18n | I18NInterface | 国际化组件 |
| Option | FlowRunnerOptions | 运行器选项 |
| 方法 | | |
| 方法签名 | | 说明 |
| void Initialize() | | 初始化框架平台 |
| void Dispose() | | 销毁当前Runner |

持续更新。。。

### TestflowRunnerOptions

|  |  |  |
| --- | --- | --- |
| 属性 | | |
| 属性名 | 类型 | 说明 |
| WorkDirectory | string | 工作目录 |
| Mode | RunMode | 运行模式。完整模式(启用所有模块)/最小运行模式(不运行设计时相关组件) |
| 方法 | | |
| 方法签名 | | 说明 |
| bool Equals(FlowRunnerOptions options) | | 两个Opitions是否相同 |

持续更新。。。

## 模块控制接口定义

模块接口定义模块提供了控制框架所有模块功能的接口，所有的接口定义如下所示：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| IComInterfaceLoader | Interface | 组件接口加载模块 |
| IConfigurationManager | Interface | 配置管理模块 |
| IDataPersistance | Interface | 数据持久化模块 |
| IEngineController | Interface | 引擎控制模块 |
| ILogService | Interface | 日志服务模块 |
| IParameterChecker | Interface | 参数检查模块 |
| IResultManager | Interface | 结果管理模块 |
| ISequenceSerializer | Interface | 序列持久化模块 |

## 序列流程信息接口定义

序列流程信息接口定义了保存序列流程定义的数据接口，主要的接口定义如下所示：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| ITestProject | Interface | 测试工程文件接口 |
| ISequenceGroupCollection | Interface | 测试序列组集合 |
| ISequenceGroup | Interface | 测试序列组 |
| ISequenceGroupInfo | Interface | 保存测试序列组相关信息的接口 |
| ISequence | Interface | 保存单个测试序列信息的数据结构 |
| ISequenceStep | Interface | 保存测试序列中单个步骤的数据结构 |
| IFunctionData | Interface | 保存一个步骤中调用的方法信息 |
| ILoopCounter | Interface | 序列步骤的循环计数器 |
| IRetryCounter | Interface | 序列步骤的重试计数器 |
| IVariable | Interface | 变量数据 |
| IArgument | Interface | 参数类，描述FunctionData的接口 |
| IAssemblyInfo | Interface | 保存单个Assembly的描述信息 |

## 序列参数配置信息接口

序列参数配置接口定义了对应测试序列的数值配置信息，主要的接口定义如下所示：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| ISequenceGroupParameter | Interface | 测试序列组参数配置 |
| ISequenceParameterInfo | Interface | 序列参数配置相关信息 |
| ISequenceParameter | Interface | 单个Sequence的参数配置 |
| ISequenceStepParameter | Interface | 序列单个步骤的参数配置信息 |

## 接口描述信息接口

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| IComInterfaceDescription | Interface | 组件接口描述类 |
| IClassInterfaceDescription | Interface | 类类型描述信息 |
| IFuncInterfaceDescription | Interface | 方法描述接口 |
| IPropertyDescription | Interface | 属性描述接口 |
| IArgumentDescription | Interface | 参数描述接口 |

## 设计时控制接口

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| IDesignTimeService | Interface | 设计时服务 |
| IDesignTimeSession | Interface | 设计时会话类 |
| IDesigntimeContext | Interface | 单个测试序列组的设计时上下文 |

## 运行时控制接口

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| IRuntimeService | Interface | 运行时服务 |
| IRuntimeSession | Interface | 运行时单个测试用例组运行的会话 |
| IRuntimeContext | Interface | 保存单个运行会话的上下文信息 |
| IRuntimeStatusInfo | Interface | 保存一个测试序列组运行时单个监视点的即时状态信息 |
| IPerformanceResult | Interface | 性能结果 |
| ISequenceFailedInfo | Interface | 失败信息类 |
| ISequenceTestResult | Interface | 保存一个测试序列组执行后的综合状态统计信息 |
| ISequenceStatusInfo | Interface | 保存一个测试序列组运行时单个监视点的综合状态信息 |
| ISequenceDebugger | Interface | 单个测试序列的调试会话 |
| IDebugInformation | Interface | 单个Step的调试信息 |
| IBreakPointsInfo | Interface | 单个序列的断点信息 |

## 公共接口

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| ITypeData | Interface | Testflow中的类型信息 |
| TestflowException | Class | Testflow的异常基类 |
| TestflowAssertException | Class | Testflow的断言异常 |
| TestflowDataException | Class | Testflow的数据异常 |
| TestflowInternalException | Class | Testflow内部异常 |
| TestflowRuntimeException | Class | Testflow运行时异常 |

持续更新。。。

# 用户组件接口

用户组件开发接口提供开发Testflow用户组件时的接口声明，这些接口是非强制使用的，使用这些接口能便于框架在设计时或运行时管理用户组件。

## 设计时的用户组件接口定义

设计时的用户组件主要用以控制用户组件的类/属性/方法在设计时的相关行为。目前定义的用户接口组件包括：

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| TestflowElementAttribute | Attribute | 在Testflow中被使用的元素注解 |
| TestflowTypeAttribute | Attribute | Testflow中会被使用的数据类型 |
| HideAttribute | Attribute | 是否隐藏某个类/属性/方法 |

持续更新。。。

## 运行时的用户组件接口定义

设计时的用户组件主要用以控制用户组件的类/属性/方法在运行时的相关行为。目前定义的用户接口组件包括

|  |  |  |
| --- | --- | --- |
| 接口名称 | 类型 | 功能说明 |
| SynchronizedFunctionAttribute | Attribute | 资源的同步检测方法 |
| ISynchronizedResource | Interface | 同步资源的接口 |

持续更新。。。