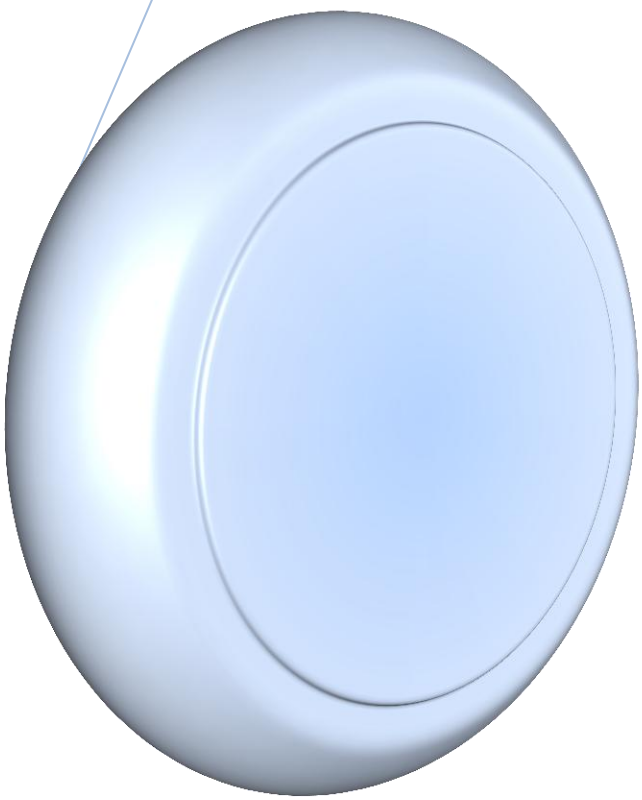

Chorus 本地模式版

使用手册

本文面向架构、程序开发人员简介了 Chorus 及本地框架版的理念及其功能。用以引导用户更好的了解并导入 Chorus 于产品研发过程中。

Allen
4/9/2011

目录

导入 1

简介 2

发布订阅模型 2

双向同步化 2

Chorus 做什么 2

Chorus 不做什么 2

Chorus 的版本 3

基线版本 3

执行版本 3

Chorus 的优势 4

功能概述 6

要约 6

泛型操作 6

发布 6

注册 6

注销 7

执行器 7

订阅 8

注册 8

注销 8

执行序列 9

框架其它操作 9

行为描述类 10

发布行为 10

默认返回值 10

本地限定 10
单线程限定 10
参数转换器 10
返回值转换器 11
跟踪 11
订阅行为 11
返回值检查器 11
本地限定 12
并发运行 12
强制执行 12
执行顺序 12
参数转换器 12
返回值转换器 13
异常容忍 13
跟踪 13
并发运行 14
订阅的并发运行 14
功能 14
执行顺序调整 14
与执行顺序行为共同作用 15
与强制执行行为共同作用 15
与参数转换器、返回值转换器、返回值检查器、异常容忍行为共同作用 15
发布控制 15
实例信息提供 16
发布获取 16
订阅获取 16

一般信息 16
跟踪信息 16
信息封装和内容 16
跟踪功能 18
关于示例程序 19
Hello World 19
Demo1：Hello World! 19
Demo2：多订阅调用 19
Demo3：带返回值的调用 19
行为描述使用 19
Demo4：返回值检查以及默认返回值 19
Demo5：执行顺序控制 20
Demo6：订阅的参数和返回值转换器 20
Demo7：发布的参数和返回值转换器 20
Demo8：异常容忍和重定向 20
Demo9：异常容忍的影响范围以及强制执行 20
并发执行 20
Demo10：简单多线程模式 20
Demo11：可取消的多线程模式以及单线程限定 20
Demo12：后置式多线程模式与执行顺序控制 20
实例信息提供 21
Demo13：实例信息提供 21
跟踪功能 21
Demo14：发布的跟踪 21
Demo15：订阅的跟踪 21
实用实例 21

Demo16：实际使用案例 21

致谢 22

导入

小李：“小张在吗？帮我把这个文件分发一下。”

小王：“小张出去了。我来帮你发吧。”

小李：“好的。”

这种对话，也许您也经常听到。从技术角度来看，为什么小王可以代替小张为小李发文件呢？

- 小李关注的是作业的完成，而不是执行作业的人；
- 小张和小王都可以理解小李说的话；而且，
- 小张和小王都可以，虽然可能方法不同，完成小李要求的本项工作。

在软件工程中，尤其是面向对象的程序开发中，又何尝不是惊人的类似？

- 一项功能的调用者，通常亦只是关注执行的结果，而并不想，虽然由于技术所限而有时不得不，关注由哪个模块去执行；
- 如果这个功能可以由多个模块替换，那么他们必须拥有同样的相关接口；而且，
- 不论这些功能模块的处理流程如何，他们的目标都是完成这项工作。

在面向对象的软件设计之初，通常架构人员和主要设计者都会关注于将上述故事可以在产品中呈现。但随着设计的逐层细化，功能要求的逐渐丰富，这些美好的愿景却也慢慢成为了一种追求，而不是要求。随着程序进入“狂热开发阶段”，这些仅存的追求也逐渐被抛之脑后，最终得到的结果，可能与最初设计的精妙之处大相径庭。

您需要一个工具，一个方法，一种保障。您需要 Chorus。

简介

Chorus 是一个基于 Microsoft .net Framework 4.0 的双向同步化的基于主题的发布订阅模型。

发布订阅模型

发布/订阅（publish/subscribe 或 pub/sub）是一种消息范式，消息的发送者（发布者）不是计划发送其消息给特定的接收者（订阅者）。而是发布的消息分为不同的类别，而不需要知道什么样的订阅者订阅。订阅者对一个或多个类别表达兴趣，于是只接收感兴趣的消息，而不需要知道什么样的发布者发布的消息。

在基于主题的系统中，消息被发布到主题或命名通道上。订阅者将收到其订阅的主题上的所有消息，并且所有订阅同一主题的订阅者将接收到同样的消息。发布者负责定义消息的类别，订阅者才能订阅。

——维基百科

这种模型提供了松散耦合的方式，各个部件的相互依存仅松散的体现在逻辑中，而非物理要求。

双向同步化

Chorus 有别于常见的发布订阅模型。

通常的发布订阅模型，是基于应用程序之间，为其提供接口功能。因此，他们多采用异步通讯机制，并且只能单向传输数据（双向传输采用两条互逆的单向管道模拟进行）。这种机制最大限度的减小了程序间的互相影响，但却不适合用于程序内的通讯。

程序内要求的是高效的双向访问。我们既不能期望方法调用会延迟数秒甚至数分钟，也不愿将所有的方法均定义为无返回值。因此，Chorus 采用了双向同步化的执行机制。即信息会在执行时传递，且处理结果也将按照原路即时返回。

CHORUS 做什么

Chorus 是一个框架，它实现了多个部件之间的通讯功能。在它的协助下，软件系统中的每个部件可以自由通讯，而不论这些部件运行在同一个程序，同一台计算机，还是在世界的任何角落。

Chorus 是一种模型，它展示了软件系统内部的物理结构。在它的支持下，软件系统中的每个部件可以随心拼装，而不论这些部件构造于您的产品，您的企业，还是获取自其他供应商。

Chorus 是一份标准，它定义了每个程序部件的接口规格。在它的规范下，软件系统中的每个部件可以严格定义，而不论这些部件开发于同一个员工，同一个小组，还是来自外包公司。

CHORUS 不做什么

Chorus 不会影响您的系统的逻辑架构。Chorus 仅代表一种物理结构，相关部件所属的逻辑层次并不会因此而发生任何改变。为了创造出精美的系统，您仍然需要进行良好的逻辑架构设计。

Chorus 不会接管您的系统的对象管理。Chorus 框架仅维护部件之间消息的传递，而不会接管对象的创建、销毁操作，也不会修改您的部件的任何属性。您仍然需要对每个部件的生存周期进行维护。

Chorus 不会掌控您的系统的执行流程。Chorus 框架仅关注消息传递的过程，而不会试图控制您的整体系统。您甚至可以在系统运行的过程中随时终止框架的运行，除受其管理的通讯不再有效之外，您的系统不会有任何影响。

CHORUS 的版本

Chorus 分为基线版本以及各个执行版本。

基线版本

基线版本为所有执行版本的设计基线。基线版本中不包含框架的执行代码，而只定义了框架的接口规范。基于基线版本接口开发的各部件，可以在各执行版本的框架中直接使用而无需修改。除对应框架的特定功能无法实现外，无任何其它功能差异。

基线版本包含：

- SecretNest.Chorus.FrameworkBaseline.dll：框架基线，定义了框架的接口点；
- SecretNest.Chorus.InteroperationBaseline.dll：接口基线，定义了访问框架所需的所有实体类、枚举、行为描述类及其它协助功能等；以及，
- 对应的帮助文件。

由于各个执行版本中引用了基线版本，且对应的帮助文件中也包含了基线的部分。因此您通常不需要单独获取这个版本。

帮助文件采用微软 MSHelpViewer 格式，您需要使用相应工具并安装帮助文件包即可浏览。Visual Studio 2010 光盘中包含此工具，且 Visual Studio 2010 的相关帮助亦使用此格式。

执行版本

您现在获取的是本地模式版。

本地模式版

本地模式版是 Chorus 在单一程序环境中的实现。它提供了在同一个程序内部的发布订阅模型功能，不包括跨程序、网络信息传递功能。

本地模式版包含：

- SecretNest.Chorus.FrameworkBaseline.dll：框架基线，定义了框架的接口点；
- SecretNest.Chorus.InteroperationBaseline.dll：接口基线，定义了访问框架所需的所有实体类、枚举、行为描述类及其它协助功能等；

- SecretNest.Chorus.Framework.LocalMode.dll：本地模式版框架，实现了本地模式版功能；
- SecretNest.Chorus.Interoperation.LocalMode.dll：本地模式版接口，定义了访问本地框架所特有的实现；以及，
- 对应的帮助文件：包含了本地模式版与基线版本的描述。

本地模式版基于基线版开发，实现了绝大多数功能，但不包括与网络功能相关的部分。具体功能可参考后文对照表。

帮助文件采用微软 MSHelpViewer 格式，您需要使用相应工具并安装帮助文件包即可浏览。Visual Studio 2010 光盘中包含此工具，且 Visual Studio 2010 的相关帮助亦使用此格式。

其它版本

Chorus 目前仅发布本地模式版。但在路标规划中，还包含两个后续版本：

- 网络模式版：提供标准的分布式程序模型，基于网络传输消息，可以支持本地多程序以及网络多程序的互操作；以及，
- 负载均衡模式版：在网络模式版基础上，实现多处理终端的负载均衡、故障转移、动态升级等高级特性。

CHORUS 的优势

使用 Chorus 本地模式版，您的系统可以做到：

- 封闭执行代码。每个部件无需了解其它部件的功能以及开发状态，即可轻松进行开发、调试、封装和发布。
- 货架化拼装。您可以在您的资源库中按照项目实际需要进行选择，拼装成符合用户需求的项目产品。拼装缺口而产生的新开发的组件则可以加入资源库中，以丰富随后项目的选择。拼装出的产品完美符合用户的需求，且可以又不包含任何用户所不需要的功能，以加快执行效率。
- 压缩执行时间。通过并行化执行代码（可选功能），您的程序可以提高执行效率，充分为多核处理器提供优化支持。
- 独立的鉴权与审计支持。您的鉴权、审计以及其它类似功能的模块，可以在不改动任何业务代码的情况下实现其功能，且可以根据您的需要随时加入和移除。

使用 Chorus 网络模式版，您的系统还可以做到：

- 封装网络功能。每个部件无需关注网络传输，系统亦无需考量网络调用等技术细节，即可获得完整的网络分布式调用功能。
- 打破程序壁垒。使用网络模式的多个程序之间，可以轻松对接，不同程序互访，如同访问自身部件一般的轻松和方便。

使用 Chorus 负载均衡模式版，您的系统更可以做到：

- 负载均衡和故障转移。对于实现相同功能的终端，Chorus 可自动调度其负载，并在其中某些终端故障时，自动转移其负载至其它终端。

- 支持软件动态升级。对于相同功能的终端，您可以随时升级您的业务应用部件，得益于故障转移与版本控制功能，无需停止服务即可实现系统的动态升级。
- 支持硬件动态扩充。当您的硬件系统已经无法满足业务压力时，简单的增加新的硬件，即可及时拓展您的系统能力。

您的系统可以轻松拥有这些功能，只因您选择了 Chorus。

功能概述

Chorus 本地框架在实例化后即可提供服务。程序分别注册发布和订阅后，即可实现互联操作。

本章节不涉及行为描述、实例信息提供的介绍。相关信息将在后文中以独立章节进行描述。

要约

人与人要交流，必须掌握同样的语言。程序部件也同样如此。Chorus 提供了沟通的管道，却未限制传导的信息。现在，我们先来完成对信息的定义。

对于一个 Chorus 应用，传导的信息主要有以下三个要素：

- 名称：消息的名称，Chorus 以此区分不同的管道，使得不相关的事件互不影响；
- 参数实体：Chorus 可以带有一个任意类型的参数，我们可以选择使用系统定义的类型或自定义的类、枚举、结构体等；以及
- 返回值实体：Chorus 允许返回一个任意类型的值，我们可以选择使用系统定义的类型或自定义的类、枚举、结构体等，也可以不使用任何返回值。

特别的，对于参数和返回值，我们建议其类型应为可序列化的。使用调试功能的实体深拷贝（在实例信息提供章节详述）或以后的带有网络功能的框架版本时，可序列化是一个必须的选择。

泛型操作

Chorus 框架的发布、订阅相关的方法，均提供泛型版本与非泛型版本可供选择。泛型版本仅提供对应的参数、返回值约定以及执行期间的强制类型转换，以便用户编写程序时获得更加明晰的指引。泛型功能中并不涉及多类型之间的转换，也不会因为其参数、返回值的类型不同而区分消息管道（消息管道仅以消息名称作为区分）。

发布

在定义要约之后，我们必须向需要发布信息的程序部件提供一个发布信息的方式。这就是发布的注册。相应的，它的反向操作称为发布的取消。

本地模式版框架，实现了基线版本的相关功能要求，提供了标准的发布注册与注销。

注册

方法名	RegisterPublisher
参数	String messageName，消息名称；以及 PublisherBehaviorCollection behaviors，行为描述，可选项。
返回值	PublisherTicket，注册存根。

注册一个带有返回值的发布。其中 messageName 是对应消息的名称，behaviors 为行为描述集合，可选项，后文将有独立章节描述。返回值为 PublisherTicket 的实例，其中包含了本次注册的描述以及一个重要只读属性 Executor。Executor 为执行器，在本章的后文中有单独段落介绍。

注册不需要返回值的发布，请使用此方法：

方法名	RegisterVoidPublisher
参数	String messageName，消息名称；以及 PublisherBehaviorCollection behaviors，行为描述，可选项。
返回值	PublisherTicket，注册存根。

此方法与 RegisterPublisher 功能相似，参数、返回值相同。唯一区别在于其注册的发布，在执行过程中不会获得任何返回值。

注销

当注册过的发布不再需要时，使用此方法注销：

方法名	UnregisterPublisher
参数	PublisherTicket publisherTicket，注册时得到的注册存根；或 Guid publisherId，发布标识，可于注册存根中获取（属性 Id）。
返回值	无

参数两者取其一即可，功能相同。如果指定的发布在框架中无法找到，则会抛出异常。

发布注销后，其相关的所有执行器将不再会触发对应订阅的执行。

发布的注册与注销过程均不涉及非托管资源。因此，如果您希望退出系统或停止框架运作时，可以无需事先注销所有已注册的发布。

执行器

注册存根中包含 Executor 只读属性，可以获得对应发布的执行器。执行器是用户代码执行发布的代理，用户代码通过调用执行器的相关方法执行发布动作。

执行器包含两个方法：

方法名	Execute
参数	parameter，类型为 object 或注册时指定的参数类型，本次执行的参数。
返回值	类型为 object 或注册时指定的返回值类型，本次执行的返回值；或 无（对于无返回值的注册）。

以及

方法名	ExecuteWithProvider
参数	parameter，类型为 object 或注册时指定的参数类型，本次执行的参数。
返回值	ReturnWithInstanceServiceProvider 对象，实例信息提供器。

实例信息提供器中包含了执行的返回值（同 Execute 的返回值），以及更多其它信息，在实例信息提供章节详述。

两个方法除提供的返回值不同外，对订阅的用户代码执行功能并无差异。

无返回值的执行器，与同消息名的带有返回值的执行器，对订阅的用户代码执行功能并无差异，差异仅表现在执行后不返回任何返回值。

当执行器所属的发布被注销或框架被关闭后，执行器的方法将不会执行用户的订阅代码。对于有返回值的执行器，返回其默认返回值（将在行为描述章节中详述）；对于无返回值的执行器，则直接返回空。

订阅

在定义要约之后，我们必须向需要订阅信息的程序部件提供一个订阅信息的方式。这就是订阅的注册。相应的，它的反向操作称为订阅的注销。

本地模式版框架，实现了基线版本的相关功能要求，提供了标准的订阅注册与取消。

注册

方法名	RegisterSubscriber
参数	注册订阅的所需参数集。
返回值	SubscriberTicket，注册存根。

参数集包含：

- handler，订阅执行的实际方法：
  - 通常情况下，应为 Subscriber 委托的一个方法实例对象；或
  - 对于需要使用实例信息提供功能的，则为 SubscriberWithProvider 委托的一个方法实例对象。
- 消息名称配对机制：
  - String messageName：提供消息的准确名称，将订阅注册到对应的消息管道中；
  - String messageName 以及 PrefabricatedMatchingMethod messageNameMatching：以？及*匹配（参考 DOS/Windows 文件名的匹配方式）或正则匹配的方式，将订阅注册到所有符合名称规则的消息管道中；
  - MatchingHelper messageNameMatchingHelper：将订阅注册到所有符合用户自定义的匹配方式的消息管道中，用户需自行实现 MatchingHelper 的子类，并完成其中的匹配方法；或
  - 不提供此类参数：将订阅注册到所有的消息管道中。
- SubscriberBehaviorCollection behaviors：行为描述，可选提供。

方法的返回值为 SubscribberTicket 的实例，其中包含了本次注册的描述。

注册不提供返回值的订阅，请使用方法 RegisterVoidSubscriber。此方法的参数与返回值与 RegisterSubscriber 相似，差别仅限于 handler 定义为不带返回值的委托。

注销

当注册过的订阅不再需要时，使用此方法注销：

方法名	UnregisterSubscriber
参数	SubscriberTicket subscriberTicket，注册时得到的注册存根；或 Guid subscriberId，订阅标识，可于注册存根中获取（属性 Id）。
返回值	无

参数两者取其一即可，功能相同。如果指定的订阅在框架中无法找到，则会抛出异常。

订阅的注册与注销过程均不涉及非托管资源。因此，如果您希望退出系统或停止框架运作时，可以无需事先注销所有已注册的订阅。

执行序列

当一个发布的执行器方法被调用时，Chorus 框架则会逐个执行其对应的每个订阅，直到所有订阅执行完毕或某个订阅返回了可以接受的返回值为止，并将执行结果（如果有且需要）返回给执行器方法的调用者。

订阅返回的返回值可接受的条件，必须同时满足：

- 订阅有返回值（使用 RegisterSubscriber 而非 RegisterVoidSubscriber 注册）；且
- 订阅的返回值满足返回值检查。

默认的返回值检查为非空检查，即返回值不为 null 则可接收。由此可见，所有的值类型返回值均被视为可接受的返回值。如您需要改变这种情况，则需要将返回值由值类型改变为引用类型（例如使用 Nullable 封装），或在注册其订阅时提供返回值检查行为描述（后文详述）。

框架其它操作

Chorus 框架还提供有：

- Version 只读属性：返回框架的版本；以及
- Shutdown 方法：关闭框架功能。

框架关闭后，相关资源会被释放，框架功能将不可用。执行器将无作用化（参考上文执行器章节）。发布与订阅的注册、取消操作均会抛出异常。

行为描述类

Chorus 允许在注册发布、订阅时，附加以多个行为描述类的实例，用以改变默认的执行流程或功能。

本章节不涉及多线程执行与实例信息提供的介绍。相关信息将在后文中以独立章节进行描述。

发布行为

Chorus 的发布行为派生于 PublisherBehavior 类。在使用时，应将所有选择的发布行为实例化，并以其构造 PublisherBehaviorCollection 实例，再于注册时作为参数传入即可生效。

Chorus 的发布行为有：

默认返回值

类：PublisherDefaultReturnBehavior。

构造时需要传入：object defaultReturn。

功能：设置发布的默认返回值。当执行过程中无任何可接受的返回值被返回时，返回此行为描述设定的值至发布执行器方法的调用者。

缺省情况：未使用此行为描述，当执行过程中无任何可接受的返回值被返回时，对于值类型的返回其缺省值，对于引用类型返回 null。

本地限定

类：PublisherLocalOnlyBehavior。

功能：限定本发布仅执行在本地的相关订阅，而不会发布至网络其它终端中。本行为在 Chorus 的本地模式版中会被忽略。

缺省情况：不限定执行订阅的终端。

单线程限定

类：PublisherSingleThreadBehavior。

功能将在后文“并发运行”章节中描述。

缺省情况：遵循订阅的行为描述执行。

参数转换器

类：PublisherParameterTypeConverterBehavior。注意，本类为抽象类，用户必须实现其子类，并实现类的 Convert 方法。

功能：将执行器方法收到的参数，先通过此类的 Convert 方法转换后，再进行标准的执行流程。

缺省情况：将执行器方法收到的参数，直接送至标准执行流程使用。

返回值转换器

类：PublisherReturnTypeConverterBehavior。注意，本类为抽象类，用户必须实现其子类，并实现类的 Convert 方法。

功能：将标准流程的返回结果，先通过此类的 Convert 方法转换后，再返回至执行器方法的调用者。

缺省情况：将标准流程的返回结果，直接返回至执行器方法的调用者。

特别说明：

- 如果使用返回值转换器，不论执行器方法是否需要返回值，以及标准流程是否返回了可接受的返回值，均会执行此 Convert 方法；
- 如果同时使用了默认返回值与返回值转换器，当默认返回值生效时，其结果不会再次交由返回值转换器处理。

跟踪

类：PublisherExecutingTrackingBehavior。

功能：启用发布的跟踪功能。将在后文“实例信息提供”章节中描述。

订阅行为

Chorus 的订阅行为派生于 SubscriberBehavior 类。在使用时，应将所有选择的订阅行为实例化，并以其构造 SubscriberBehaviorCollection 实例，再于注册时作为参数传入即可生效。

Chorus 的订阅行为有：

返回值检查器

类：SubscriberDealingResultCheckHelperBehavior。注意，本类为抽象类，用户必须实现其子类，并实现类的 CheckFinished 方法。

功能：将对应订阅执行结果交由此类的 CheckFinished 方法判断是否为可接受的返回值。

缺省情况：执行默认的返回值接受判断，参考上文“执行序列”章节。

本地限定

类：SubscriberLocalOnlyBehavior。

功能：限定本订阅仅被本地的相关发布执行，而不会被网络其它终端的发布所影响。本行为在 Chorus 的本地模式版中会被忽略。

缺省情况：不限定发布来源的终端。

并发运行

类：SubscriberParallelRunningBehavior。

构造时传入（选择其一）：

- 并发运行的模式枚举：
  - ForceInPublisherThread：强制于发布执行器的方法线程中执行；
  - Default：单线程模式，默认值；
  - SimpleMultithread：简单多线程，并发执行；或
  - PostMultithread：后置式多线程。或
- CancellableMultithread 对应的取消方法委托实例。

功能将在后文“并发运行”章节中描述。

强制执行

类：SubscriberForceExecutingBehavior。

功能：不论是否已经获得了可接受的返回值，均需要执行此订阅。

缺省情况：当获得了可接受的返回值后，将不会执行未执行的订阅。

特别说明：当执行此订阅前已经获得了可接受的返回值时，此订阅的返回值（如果有）将不被采纳。

执行顺序

类：SequenceBehavior。

构造时传入：执行顺序号（整型数据）。

功能：控制执行的顺序。Chorus 以从小到大的顺序执行订阅。同顺序号的订阅执行顺序不定。

缺省情况：执行顺序号为 0。

参数转换器

类：SubscriberParameterTypeConverterBehavior。注意，本类为抽象类，用户必须实现其子类，并实现类的 Convert 方法。

功能：处理一次发布执行器方法时，将其传入的参数先通过此类 Convert 方法转换后，再传入对应的订阅。

缺省情况：处理一次发布执行器方法时，将其传入的参数直接传入对应的订阅。

返回值转换器

类：SubscriberReturnTypeConverterBehavior。注意，本类为抽象类，用户必须实现其子类，并实现类的 Convert 方法。

功能：执行完订阅后，将其返回值先通过此类的 Convert 方法转换后，再进行返回值可接受检查。

缺省情况：执行完订阅后，直接将其返回值进行返回值可接受检查。

特别说明：

- 如果使用返回值转换器，不论订阅是否有返回值，均会执行此 Convert 方法，因此参数可能为 null；
- 返回值可接受检查执行顺序位于返回值转换器后。

异常容忍

类：SubscriberExceptionToleranceBehavior。

构造时传入（可选）：异常处理的重定向方法委托的实例。

功能：忽略或重定向对应订阅执行过程中的异常。

当构造时传入了委托 ExceptionNotify（带有一个类型为 Exception 的参数）的实例，且订阅执行过程中发生异常时，此委托的实例将被执行，且参数为被捕获的异常。如未传入参数，则异常将被忽略。异常处理或忽略后，本订阅将被打断执行。后续订阅执行不受影响。

订阅的参数转换、执行、返回值转换、结果检查器均处于异常容忍的有效范围内。

缺省情况：当订阅执行过程中出现异常时，需要用户处理异常，且后续订阅亦不会被执行。

跟踪

类：SubscriberExecutingTrackingBehavior。

功能：启用订阅的跟踪功能。将在后文“实例信息提供”章节中描述。

并发运行

缺省情况下，所有订阅将在单线程下被逐一运行。但用户可以通过行为描述改变这种机制。

注意：dotNet 框架在处理多线程代码时，应注意相互的锁定以及对 UI 访问的线程限制。

订阅的并发运行

订阅行为中包含并发运行类：SubscriberParallelRunningBehavior。

构造时传入（选择其一）：

- 并发运行的模式枚举：
  - ForceInPublisherThread：强制于发布执行器的方法线程中执行；
  - Default：单线程模式，默认值；
  - SimpleMultithread：简单多线程，并发执行；或
  - PostMultithread：后置式多线程。或
- CancellableMultithread 对应的取消方法委托实例。

功能

- ForceInPublisherThread：当 Chorus 工作在网络模式且由本地某个发布执行器被执行时，强制要求本订阅于发布执行器方法所在线程内执行，在本地框架版中等同于 Default；
- Default：单线程模式，默认值；
- SimpleMultithread：简单多线程，Chorus 将对于每个订阅启动独立线程执行其用户代码；
- CancellableMultithread：可取消的多线程，Chorus 将对于每个订阅启动独立线程执行其用户代码，且当不再需要其继续执行时（已经由其他订阅返回了可接受的返回值），调用指定的委托实例；
- PostMultithread：后置式多线程，当发布执行器被返回后，再执行此订阅，订阅的返回值将被忽略。

由于可取消的多线程需要指定一个对应的取消方法委托实例，如果在构造时传入并发运行的模式枚举 CancellableMultithread 将会引发一个异常。应使用第二种构造函数，直接传入对应的取消方法委托实例来构造本订阅行为。

可取消的多线程对应的取消方法委托为 SubscriberRunningCancellationRequest，包含一个参数 Guid messageId，传入其消息识别码。有关消息识别码，请参考后文“实例信息提供”章节。

取消方法委托实例将在某个线程中被执行，此线程为获得了可接受的返回值的订阅所在的线程。

缺省功能：等同于 Default。单线程执行。

执行顺序调整

Chorus 框架会先对每个需要多线程处理的订阅开启独立线程并开始执行其用户代码，再逐个在当前线程执行单线程模式的订阅。Chorus 会在单线程模式的订阅执行之前，开始所有多线程订阅的线程，但不保证其代码均在单线程订阅执行之前开始执行。

在完成发布执行器的方法后，再对每个后置式多线程订阅开启独立线程并执行其用户代码。由于返回的过程需要小段执行时间，Chorus 框架不能保证后置式多线程的开始时间与发布执行器的返回时间的先后顺序。

与执行顺序行为共同作用

Chorus 会先将所有的订阅按照执行顺序号分组，并依次执行。

在同分组内，将参考上节“执行顺序调整”的方式进行执行。当且仅当本分组内所有非后置式多线程的订阅执行完成（包括单线程、多线程），才会执行下一顺序分组。

所有的后置式多线程订阅，将忽略其执行顺序，而在发布执行器方法返回后再统一执行。

与强制执行行为共同作用

当一个订阅同时使用可取消的多线程方式的并发运行与强制执行两种行为时，Chorus 将自动将并发运行切换为简单多线程方式。

与参数转换器、返回值转换器、返回值检查器、异常容忍行为共同作用

当一个订阅使用简单多线程、可取消的多线程或后置式多线程方式的并发运行时，其参数转换器、返回值转换器、返回值检查器、异常容忍均会工作于与其订阅相同的独立线程中。

发布控制

发布行为中包含单线程限定类：PublisherSingleThreadBehavior。

功能：强制所有的订阅在单线程模式下运行，而不论其行为描述是否指定为多线程功能（对简单多线程以及可取消的多线程有效，对后置式多线程无效）。通常，使用此行为可以确保 UI 更新的线程正常化。

缺省情况：遵循订阅的行为描述执行。

实例信息提供

注意：“实例信息提供”为英文原文 InstanceServiceProider 的中文翻译，名词词性，在本章节中未有动词含义。

Chorus 框架被执行器的方法触发后，会在内部进行一系列的动作。通常，用户并不关心这些流程信息，但当用户需要了解这些细节时，可以通过实例信息提供机制来有效获取。

对于每次触发，Chorus 框架都会产生一个消息识别码作为本次处理的唯一值。在实例信息提供机制中，用户可以获得此识别码。另外，可取消的多线程对应的取消方法委托实例被执行时，此识别码也将被作为参数而传入，以方便其代码确定需要被取消的订阅所在的线程等信息。

同时，实例信息提供机制，也会提供消息的名称。对于同时绑定于多个不同名称的订阅来说，通过此信息可以更好的了解正在处理的消息。

发布获取

对于发布，可以于其对应的执行器中获取实例信息提供。使用执行器中的 ExecuteWithProvider，返回值其中包含了本次执行的返回值以及对应的实例信息提供。

订阅获取

对于订阅，首先，应选择 SubscriberWithProvider 委托型式的方法而非 Subscriber 委托型式，即，在参数中含有实例信息提供。在框架执行至此时，对应的实例信息提供将被以参数的形式传入此订阅的方法。

一般信息

实例信息提供以 InstanceServiceProvider 类的实例形式提供。此类定义于基线版本中，因此可以在各种执行版本中无缝兼容。

实例信息提供包含：

- MessageId：Guid 类型，本次执行的消息识别码；
- MessageName：String 类型，消息的名称。

以及跟踪信息，在下章节中详述。

跟踪信息

注意：跟踪信息功能默认关闭。使用此功能会显著的降低执行效能。

信息封装和内容

实例信息提供中包含两个属性：

- TrackingProvider：提供特定执行版本的跟踪信息；以及
- BaselineTrackingProvider：提供基线版本的跟踪信息。

两者提供的信息来源相同。区别仅在于 TrackingProvider 的类型为特定执行版本的实现，通常包含更多信息；而 BaselineTrackingProvider 使用基线版本定义的类，拥有更好的兼容性。对比参考下表：

	基线版本	本地模式版	功能
PublisherBehaviors	包含	包含	发布的所有行为描述
EntryTime	包含	包含	本次发布执行的开始时间
ExitTime	包含	包含	本次发布执行的结束时间
Items	包含	包含	本次执行的所有订阅
ParameterType	未包含	包含	发布的参数类型
ReturnType	未包含	包含	发布的返回值类型
Parameter	未包含	包含	本次发布的参数
Return	未包含	包含	本次发布的返回值

EntryTime 的记录点位于参数转换器执行前；ExitTime 的记录点位于返回值转换器、默认返回器执行后。

ParameterType 与 ReturnType 仅与注册时提供的泛型有关，与实际参数、返回值的类型无关。

Parameter 值，来自入口点实际数据；Return 值，来自返回点的实际数据。两者均处于参数、返回值转换器的外层，即是未转换的参数以及转换后的返回值。

其中 Items 中包含的元素：

	基线版本	本地模式版	功能
SubscriberId	包含	包含	订阅标识
SubscriberBehaviors	包含	包含	订阅的所有行为描述
EntryTime	包含	包含	本次订阅执行的开始时间
ExitTime	包含	包含	本次订阅执行的结束时间
RunningStatus	包含	包含	订阅的运行状态
ParameterType	未包含	包含	订阅的参数类型
ReturnType	未包含	包含	订阅的返回值类型
Parameter	未包含	包含	本次订阅的参数
Return	未包含	包含	本次订阅的返回值
Exception	未包含	包含	本次订阅执行过程中的异常，无则为空

EntryTime 的记录点位于错误容忍起始点、参数转换器执行前；ExitTime 的记录点位于返回值转换器、结果检查器、错误容忍结束点执行后。

ParameterType 与 ReturnType 仅与注册时提供的泛型有关，与实际参数、返回值的类型无关。

Parameter 值，来自订阅参数的框架内数据；Return 值，来自返回点的框架内数据。两者均处于参数、返回值转换器的外层，即是未转换的参数以及转换后的返回值。

当对应的订阅开启了异常容忍后，Exception 将会被记录在此。如果订阅为可取消的多线程方式，在执行取消过程时发生的异常不会被记录在此。

跟踪功能

由于跟踪信息的记录和保存均需要较高的执行时间和空间，使用此功能会显著的降低执行效能，跟踪功能默认是关闭的，即上节中所述两个属性，通常为空值。

跟踪功能以行为描述的方式开启，发布行为与订阅行为均包含了跟踪功能相关的选择。

本地模式版提供了跟踪行为描述类的子类实现，附加以更多的功能。其类名与基线版的类名相同，保存于 SecretNest.Chorus.Interoperation.LocalMode 命名空间中。新类在构造时，需要给定 4 个参数，用以详细控制跟踪功能的开启程度：

- includePublisherBehaviors：布尔值，确定是否记录发布的行为描述，如未开启，则对应属性为空，下同；
- includeSubscribers：布尔值，确定是否记录所执行的订阅；
- includeSubscriberBehaviors：布尔值，确定是否记录所执行的所有订阅的行为描述，需要同时开启 includeSubscribers 方能生效；以及
- includeEntity：包含参数、返回值的方式。

其中，includeEntity 为枚举型，可能的值有：

- NotIncluded：不记录参数、返回值；
- Shadow：以浅拷贝的方式记录参数、返回值；或
- DeepCopy：以深拷贝的方式记录参数、返回值。

当使用 DeepCopy 模式时：

- 如参数、返回值为值类型：同 Shadow 模式；
- 如参数、返回值实现了 ICloneable：使用其 Clone 方法复制；或
- 其它情况：使用二进制深拷贝方式（需要序列化支持）。

includeEntity 影响跟踪信息的 Parameter 与 Return 属性。当 includeSubscribers 为真时，所有订阅的参数、返回值亦被影响；否则仅对发布有效。

如果选择使用基线版本的行为描述，将被解释为：

- 不包含发布的行为描述；
- 不记录所执行的订阅；且
- 不记录参数、返回值。

再次提醒您：使用此功能会显著的降低执行效能，请在当且仅当功能需要时，以最小需求的方式开启跟踪功能。

关于示例程序

跟随本地模式版，同时提供了由 C# 4.0 开发的 16 个示例程序的源代码，帮助您更好的理解和使用 Chorus。

您可以使用 Visual Studio 2010 的任何版本（包括 Express 版）打开和执行示例程序。Express 版本可自微软网站免费下载（http://www.microsoft.com/express/Downloads/#2010-Visual-CS）。

HELLO WORLD

Chorus 的简单应用示例。

DEMO1：HELLO WORLD!

一个发布、一个订阅的简单运行。注册一个发布 publisherTicket、一个订阅 subscriberTicket，并执行这个发布。

特别的，由于在调用“Hello Mars!”时并未注册任何订阅，因此您不会看到这行消息被输出。而执行参数为“Hello World!”时，由于 HelloWorld 方法已被注册为订阅，所以您可以看到屏幕输出“Hello World!”文本。

代码的最后注销了所注册的发布、订阅，并关闭框架。

DEMO2：多订阅调用

本示例演示了多个订阅如何同时被关联到一个消息管道中，并使用了三种不同名称匹配方式。

在代码的最后，也演示了当订阅被注销后，不会再被执行。

DEMO3：带返回值的调用

本示例演示了带有返回值的订阅、发布的执行。

本示例模拟了一种兑奖的数字匹配方式。当数字末尾为两个 0 时，给予奖金 100；当数字末尾为 50 时，给予奖金 50；当数字末尾为 5 时，给予奖金 5。并进行了 4 个数字的测试。

行为描述使用

本节的演示程序开始通过引入行为描述，引导用户进入高级功能使用。

DEMO4：返回值检查以及默认返回值

本示例基于 Demo3 修改，增加了对于奖励检查函数的自定义返回值检查。并在三个订阅均未返回可接受的返回值时，返回默认值（-100）。

DEMO5：执行顺序控制

本示例基于 Demo4 修改，增加了对订阅执行顺序的控制。由于每个订阅在匹配失败后会输出一个点号，您可以在最终执行时通过点号的个数来确定实际的执行顺序。

DEMO6：订阅的参数和返回值转换器

本示例演示了订阅的参数、返回值转换器。本示例中的发布使用的参数与返回值为 int，而订阅接受的参数和返回值为 bool，通过用户的代码进行转换。

DEMO7：发布的参数和返回值转换器

本示例演示了发布的参数、返回值转换器。本示例中的订阅使用的参数与返回值为 int，而发布采用了类似 XML 标记的方式提供参数和返回值，通过用户的代码进行转换。

DEMO8：异常容忍和重定向

本示例通过一个除 0 异常，介绍了异常容忍和异常处理的重定向机制。

DEMO9：异常容忍的影响范围以及强制执行

本示例演示了异常容忍的影响范围。当一个订阅出现异常且异常容忍开启时，其后续的订阅会继续正常执行。

另外，本例也对 MyProc 方法是用了强制执行确保其每次均被执行。

并发执行

本节的演示程序开始介绍并发执行机制。

DEMO10：简单多线程模式

本示例演示了三个订阅的简单多线程模式的并发执行。为了使演示明显，每个订阅均采用了线程 Sleep 方法延长执行时间，并定期输出。

DEMO11：可取消的多线程模式以及单线程限定

本示例演示了可取消的多线程模式的并发执行。用户可以尝试将发布行为已注释的包含有 PublisherSingleThreadBehavior 的代码行反注释，以查看单线程限定的效果。

DEMO12：后置式多线程模式与执行顺序控制

本示例演示了后置式多线程模式（会在执行器返回后继续执行），以及执行顺序在多线程下的效果（Test7 方法）。

实例信息提供

DEMO13：实例信息提供

本示例通过对 Demo11 的完善，演示了示例信息提供的信息。

跟踪功能

本节的演示程序，提供对跟踪功能的介绍。

DEMO14：发布的跟踪

本示例演示了发布获取的跟踪信息。请在“Console.ReadKey();”行增加断点，检查 provider 中的相关值。并可以调整 PublisherExecutingTrackingBehavior 构造时的 4 项参数，以对比 provider 中信息的不同。

DEMO15：订阅的跟踪

本示例演示了订阅获取的跟踪信息。请在 Test6 与 Test7 方法增加断点，检查 provider 中的相关值。并可以调整 SubscriberExecutingTrackingBehavior 构造时的 4 项参数，以对比 provider 中信息的不同。

实用实例

DEMO16：实际使用案例

本示例演示了一个综合应用：

- Commander：用以发布红、黄、绿消息，并记录返回值；
- Green：接获并处理绿色消息；
- Yellow：接获并处理黄色消息；
- Red：接获并处理红色消息；
- Response：提供消息返回值；以及
- Log：提供全程信息记录。

启动程序后，点击工具栏相应按钮即可启动相应应用。

致谢

特别感谢好友樱桃，给予编码规范、流程优化上的意见。

同时，感谢您作为我们的用户，您的需求是我们前进的动力。

祝 Chorus 协助您共创辉煌。