# 响应式宣言

*Published on September 16 2014. (v2.0)*

来自不同领域的组织正在不约而同地发现一些看起来如出一辙的软件构建模式。它们的系统更加稳健，更加有可回复性，更加灵活，并且以更好的定位来满足现代的需求。

这些变化之所以会发生，是因为近几年的应用需求出现了戏剧性的变化。仅仅在几年之前，大型应用意味着数十台服务器，数秒的响应时间，数小时的离线维护时间以及若干 GB 的数据。而在今天，应用被部署在一切场合，从移动设备到基于云的集群，这些集群运行在数以千计的多核心处理器的之上。用户期望毫秒级的响应时间以及 100% 的正常运行时间。数据则以 PB 为单位来衡量。昨天的软件架构已经完全无法地满足今天的需求。

我们相信，一种条理分明的系统架构方法是必要的，而且我们相信关于这种方法的所有必要方面已经逐一地被人们认识到：我们需要的系统是响应式的，具有可回复性的，可伸缩的，以及以消息驱动的。我们将这样的系统之为响应式系统。

以响应式系统方式构建的系统更加灵活，松耦合和可扩展。这使得它们更容易被开发，而且经得起变化的考验。它们对于系统失败表现出显著的包容性，并且当失败真的发生时，它们能用优雅的方式去应对，而不是放任灾难的发生。响应式系统是高度灵敏的，能够给用户以有效的交互式的反馈。

## 响应式系统是：

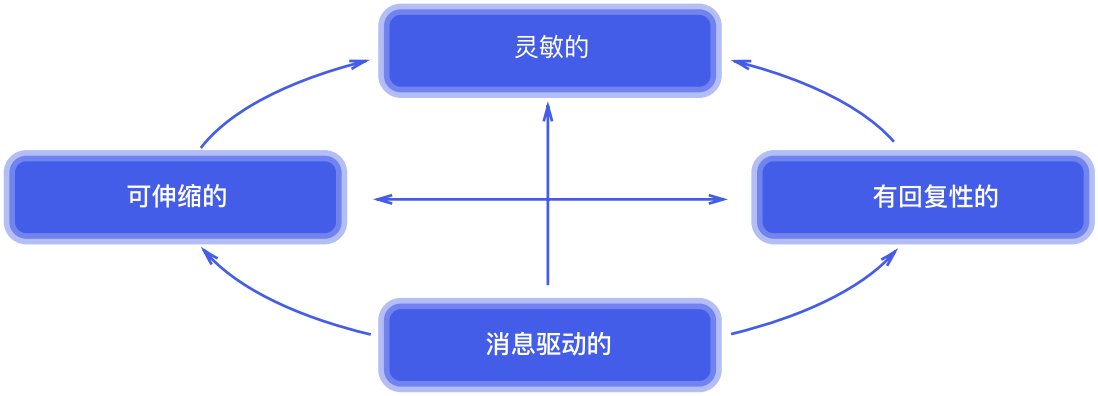
**灵敏的：** 只要有可能，系统就会及时响应。灵敏性是可用性和效用的基石，但还不止于此，灵敏性还意味着问题能够被更快地侦测到并得到有效地处理。灵敏的系统着眼于提供迅速和一致的响应时间，建立可靠的服务上限，因而它们可以交付一致的服务质量。这种一致的行为反过来又能简化出错处理，建立最终用户对系统的信心，并且促使他们与系统做进一步交互。

**有回复性的：** 系统在面临故障时也能保持灵敏度。可回复性不仅适用于高可用的关键任务系统——任何系统都可以具有这种属性，一个不具可回复性的系统一旦出现故障，就会变得不灵敏。可回复性可以通过复制，围控，隔离和委派等方式实现。在可回复性的系统中，故障被包含在每个组件中，各组件之间相互隔离，从而允许系统的某些部分出故障并且在不连累整个系统的前提下进行恢复。每个组件的恢复过程都可委派给另一个（外部的）组件来完成，并且在必要的位置可借助复制机制来确保系统的高可用性。这样一来，组件的客户就不必为处理组件自身的故障而承担压力。

**可伸缩的：** 系统在变化的工作负载下保持灵敏。响应式系统能够对输入速率的变化做出反应，通过增加或减少资源分配的方式服务这些输入。这意味着系统在设计上不存在争用点或中心的瓶颈，进而让共享或复制的模块有能力应对，并可以把输入分发给

这些模块。通过提供相关的实时性能测量数据，响应式系统既支持预测式量级扩展算法，也支持响应式的量级扩展算法。响应式系统可以在商业的硬件和软件平台上以经济实惠的方式实现可伸缩性。

**消息驱动的：**响应式系统依赖异步的消息传递 建立组件之间的界限，这一界限确保了松耦合，隔离，位置透明性等特性的实现，还提供了以消息的形式把故障委派出去的手段。利用显式的消息传递，可以通过建立和监视消息队列的方式实现负载管理、可伸缩性以及流程控制，还可以在必要时应用背压机制。位置透明的消息传递作为一种沟通手段，使得故障管理可以在相同的构造和语义下工作，无论实际的工作环境是跨集群的环境还是一个单独的主机。非阻塞沟通则允许消息接收者仅在激活的状态下消费资源，导向更少量的系统开销。




大系统由较小的系统构成，因此势必依赖于这些构成要素的响应式属性。这意味着响应式系统需要应用一定的设计原则，使这些属性在所有不同等级的构成要素上都适用，进而使它们可组合。世界上最大的系统所依赖的体系结构，都是基于这些属性构建的，它们每天都在服务数十亿人的需求。不要总是重新发现这些设计原则了，在开发之初就有意识地运用它们吧！现在正是时候！