Webhook 和 API：有什么区别?

作为人类，我们希望技术能帮助我们更快捷、更便捷地与更多人交流。但要实现这一目标，我们首先需要找到一种方法让技术能够彼此对话。

这就是 API 和 Webhook 的用武之地。

[API](http://cxyroad.com/ ”https://apifox.com/apiskills/what-is-api-endpoint/”) 和 Webhook 都能够促进两个应用之间的数据同步和传递。然而，它们的实现方式不同，因此用途略有差异。

为了解除此间的混淆，让我们看看 Webhook 和 API 的区别，以及各自最适用的场景。

Webhook 与 API：简明解释

简单来说，当你请求它时，API 会执行相应操作，而 Webhook 会在满足特定条件或发生特定事件时自动执行操作。让我们深入探讨一下。

API 可以用于从服务器与 example.com 通讯，通过这种通讯，API 可以列出、创建、编辑或删除项。不过，API 需要得到明确的指示。

而另一方面，Webhook 是 example.com 向服务器发出的自动调用。这些调用会在 example.com 上的特定事件发生时触发。例如，如果一个新用户在 example.com 上注册，自动调用可以配置为请求服务器发送一封欢迎邮件。

什么是 Webhook?

有时，Webhook 被称为反向 API，但这不完全正确。它们不是反向运行的，而是不需要你的请求，一旦有新数据，数据就会被发送。

要设置 Webhook，您只需向提供服务的网站注册一个 URL。该 URL 将接受数

据，并可以激活一个工作流程，使数据变得有用。在大多数情况下，你甚至可以指定服务提供商将在何种情况下向你传送数据。

Webhook 和 API 在请求方式上有所不同。例如，无论是否有数据更新响应，API 都会发出数据请求。而 Webhook 只会在你连接的外部系统有数据更新时通过 **[HTTP](http://cxyroad.com/”https://apifox.com/apiskills/what-are-the-http-status-codes/”)** POST 接收调用。

什么时候使用 Webhook

Webhook 通常用于执行较小的请求和任务，但在某些情况下，Webhook 比完整的 API 更合适。

一个常见场景是当你的应用或平台需要实时更新，但你不想浪费资源。这种情况下，Webhook 会非常有用。

另一个情况是在 API 本身性能不佳，或者根本没有 API 时。你可以创建一个替代解决方案，以获取你的应用运行所需的数据。

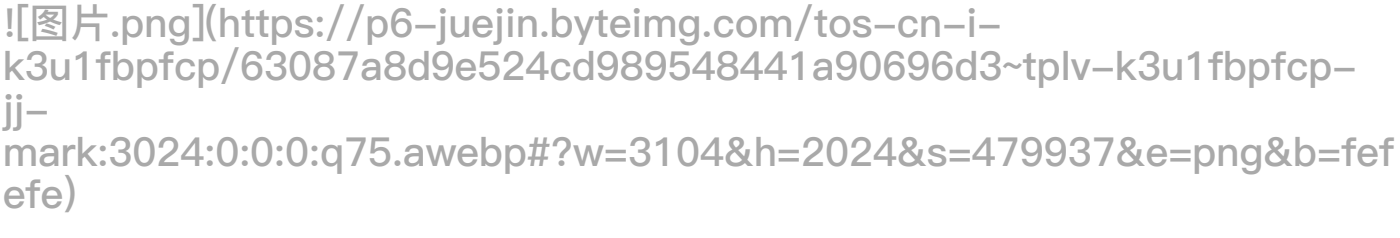
然而，需要注意的是，由于 Webhook 不会定期请求数据，只在有新数据时才会这样做，所以如果系统因某种原因离线，你可能永远不能了解到新的更新。而且，你对数据流的全面控制较少，因为你必须接受给定更新时提供的全部数据量。

实际中的 Webhook 示例

许多应用和工具依赖 Webhook，但主要用于较小的数据请求，而不是用于构建其服务的核心。尽管如此，仍有不少 Webhook 被有效使用的例子。

1. ButterCMS 的 Webhook 在任何人发布新博客文章或更新 CMS 内容时触发。
2. [Zapier 基本上就是一个巨大 Webhook](http://cxyroad.com/”https://zapier.com/zapbook/webhook/”)。你将某些应用连接起来，每当一个应用中发生事件时，它会触发另一个应用中的动作。
3. [Stripe 的 Webhook](http://cxyroad.com/”https://stripe.com/docs/webhooks”)会在订阅付款未能通过时自动向客户

发送电子邮件。
4. 我们可以在 **[Apifox](http://cxyroad.com/”http://apifox.com/?utm_source=opr&utm_medium=liam”)** 来设置 Webhook，支持发送通知事件到 HTTP Server，通过指定 URL 地址接收 POST 请求，可以将事件消息发送到 HTTP Server。



什么是 API?
========

API 代表应用程序编程接口。API 是一种通过通用通信方法让应用程序和平台相互连接的方式。要使 API 工作，需要有一个数据请求，然后是对该请求的响应。数据通常以 **[JSON](http://cxyroad.com/”https://apifox.com/apiskills/what-is-json/”)** 格式传递。

API 往往是许多现有软件和工具赖以存在的框架。例如，创建 Twitter 趋势报告的应用可以依赖 API 不断获取最新数据。大多数大型应用都集成了多个 API，以扩展其服务，如下所示。

什么时候使用 API
==========

当你的数据不断变化时，API 非常有效。如果你需要的数据相对静态，就没有使用 API 的意义。例如，如果你是一个需要定期更新其运输和追踪数据的电商店铺，那么你会不断发出请求。

每次轮询 API 时，你都会获得新数据。如果你的数据未持续更新，则不能保证另一端会有数据准备好。当这种情况发生时，你只是浪费资源。不过，如果你坚持使用 API，可以设置调用限制，限制在特定时间段内的调用次数。有些应用从一开始就限制你调用的次数，以减少其资源使用。

实际中的 API 示例
==========

正如前面提到的，API 无处不在。根据 [ProgrammableWeb 的最新结果](http://cxyroad.com/ "https://www.programmableweb.com/about")，目前已有超过 17,000 个现有 API。以下是一些依赖 API 的工具：

1. 作为 [API 优先的 CMS](http://cxyroad.com/ "https://buttercms.com/")，ButterCMS 拥有自己的 REST API，具有可预测的资源导向 URL，并使用 HTTP 响应代码来指示 API 错误——这些功能在[这个 React Universal Blog 构建](http://cxyroad.com/ "https://www.programmableweb.com/news/build-react-universal-blog-nextjs-and-buttercms/how-to/2017/05/02")中得到了展示。
2. Uber 也依赖于 Google Maps API、Twilio API、Braintree API 和 SendGrid API 来支持其应用。
3. [Slack 有一个 API](http://cxyroad.com/ "https://api.slack.com/") 使你能够将他们的消息功能集成到第三方应用中。

Webhook 和 API 的不同圈子
==========

这不是一个谁更好的问题，因为没有一种方法在所有情况下都优于另一种。问题在于你的应用目的和所需的数据类型。

举个例子，你可以把 API 想象成你发送给朋友的短信，以获取他们举办的活动信息。你提问，他们回应。

而对于 Webhook，你告诉朋友一旦他们在组织新的活动时就给你发短信，让你知道。你提出初始请求，而他们在有新信息时不断发送更新。

最终，大多数应用都会同时使用 API 和 Webhook 来创建一个能够在合适的时候传递合适类型数据的系统。