安全性简介

有许多方法可以处理安全性、身份认证和授权等问题。

而且这通常是一个复杂而「困难」的话题。

在许多框架和系统中，仅处理安全性和身份认证就会花费大量的精力和代码（在许多情况下，可能占编写的所有代码的 50% 或更多）。

FastAPI 提供了多种工具，可帮助你以标准的方式轻松、快速地处理**安全性**，而无需研究和学习所有的安全规范。

但首先，让我们来看一些小的概念。

没有时间?

如果你不关心这些术语，而只需要*立即*通过基于用户名和密码的身份认证来增加安全性，请跳转到下一章。

OAuth2

OAuth2是一个规范，它定义了几种处理身份认证和授权的方法。

它是一个相当广泛的规范，涵盖了一些复杂的使用场景。

它包括了使用「第三方」进行身份认证的方法。

这就是所有带有「使用 Facebook，Google，Twitter，GitHub 登录」的系统背后所使用的机制。

OAuth 1

有一个 OAuth 1，它与 OAuth2 完全不同，并且更为复杂，因为它直接包含了有关如何加密通信的规范。

如今它已经不是很流行，没有被广泛使用了。

OAuth2 没有指定如何加密通信，它期望你为应用程序使用 HTTPS 进行通信。

!!! tip 在有关**部署**的章节中，你将了解如何使用 Traefik 和 Let's Encrypt 免费设置 HTTPS。

OpenID Connect

OpenID Connect 是另一个基于 **OAuth2** 的规范。

它只是扩展了 OAuth2，并明确了一些在 OAuth2 中相对模糊的内容，以尝试使其更具互操作性。

例如，Google 登录使用 OpenID Connect（底层使用OAuth2）。

但是 Facebook 登录不支持 OpenID Connect。它具有自己的 OAuth2 风格。

OpenID（非「OpenID Connect」）

还有一个「OpenID」规范。它试图解决与 **OpenID Connect** 相同的问题，但它不是基于 OAuth2。

因此，它是一个完整的附加系统。

如今它已经不是很流行，没有被广泛使用了。

OpenAPI

OpenAPI（以前称为 Swagger）是用于构建 API 的开放规范（现已成为 Linux Foundation 的一部分）。

FastAPI 基于 **OpenAPI**。

这就是使多个自动交互式文档界面，代码生成等成为可能的原因。

OpenAPI 有一种定义多个安全「方案」的方法。

通过使用它们，你可以利用所有这些基于标准的工具，包括这些交互式文档系统。

OpenAPI 定义了以下安全方案：

- `apiKey`：一个特定于应用程序的密钥，可以来自：
  - 查询参数。
  - 请求头。
  - cookie。
- `http`：标准的 HTTP 身份认证系统，包括：
  - `bearer`：一个值为 `Bearer` 加令牌字符串的 `Authorization` 请求头。这是从 OAuth2 继承的。
  - HTTP Basic 认证方式。
  - HTTP Digest，等等。
- `oauth2`：所有的 OAuth2 处理安全性的方式（称为「流程」）。 *以下几种流程适合构建 OAuth 2.0 身份认证的提供者（例如 Google，Facebook，Twitter，GitHub 等）： * `implicit` * `clientCredentials` * `authorizationCode`
  - 但是有一个特定的「流程」可以完美地用于直接在同一应用程序中处理身份认证：
    - `password`：接下来的几章将介绍它的示例。
- `openIdConnect`：提供了一种定义如何自动发现 OAuth2 身份认证数据的方法。
  - 此自动发现机制是 OpenID Connect 规范中定义的内容。

!!! tip 集成其他身份认证/授权提供者（例如Google，Facebook，Twitter，GitHub等）也是可能的，而且较为容易。

最复杂的问题是创建一个像这样的身份认证/授权提供程序，但是 `**FastAPI**` 为你提供了轻松完成任务的工具，同时为你解决了重活。

FastAPI 实用工具

FastAPI 在 `fastapi.security` 模块中为每个安全方案提供了几种工具，这些工具简化了这些安全机制的使用方法。

在下一章中，你将看到如何使用 **FastAPI** 所提供的这些工具为你的 API 增加安全性。

而且你还将看到它如何自动地被集成到交互式文档系统中。