

AI 时代的可互操作本地优先软件

Noel De Martin • 2026-10-02 • 中文译文

根据 2026 年 10 月 2 日的演讲与问答整理，保留发言顺序、例子、限定条件和交流承接，校订口头重复、断句与明显识别错误。产品状态和后续计划以分享时点为准。时间对应交付视频。



Interoperable Local-First in the age of AI

noeldemartin.com/local-first-2026

可互操作的本地优先软件与AI • 画面 00:00:00

从做应用开始

Hello, my name is Noel



- Developer & Entrepreneur.
- Working with the Solid Protocol.
- Working in the Open.

noeldemartin.com | noeldemartin.social | @NoelDeMartin

noeldemartin.com/local-first-2026

Noel与开放开发 • 画面 00:00:05

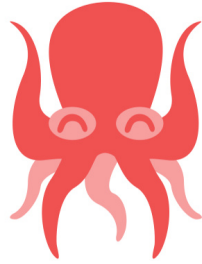
00:00:00 • Noel De Martin

大家好，我叫 Noel，是一名开发者和创业者。我研究 Solid 协议已经有一段时间了，后面会展开讲。我一直公开自己的开发过程，今天要讲的内容，以及我围绕 Solid 做的大部分工作，都是开源的。大家可以去我的 GitHub 看它们背后是怎么实现的。我也在写开发日志，记录自己正在做什么。所以，今天讲到的大多数内容，如果感兴趣，都可以继续深入了解。

My Apps



Focus



Media Kraken



Umai

Find them at noeldemartin.com/projects

noeldemartin.com/local-first-2026

Focus、MediaKraken与Umai • 画面 00:00:37

00:00:35 • Noel De Martin

我最初是从做应用开始的，也是因为做应用，才一步步接触到 Solid、本地优先这些东西。我做了几款应用，大家有兴趣可以用一用。屏幕上这三款是目前最主要、也比较稳定的应用。我的网站上还有其他项目，可以继续探索。不过，我主要做的事情还是开发应用。之所以开始自己做，是因为我对原来使用的软件不太满意。

三个理想：通用性、自主权与互操作性

My 3 Software Ideals

- Universality
 - The Web (any device)
 - Local-first (any location)

noeldemartin.com/local-first-2026

软件理想：通用性 · 画面 00:01:50

00:01:09 • Noel De Martin

进一步想过之后，我形成了三个想法：我希望自己的软件应该怎样工作。第一个是通用性，也就是 universality。我希望在任何设备上都能使用自己的软件，所以我的应用都是 Web 应用。Web 本身就是通用的，这是它的基本原则之一；网站已经可以在各种设备上使用，因此这一点就有了保障。

00:01:41 • Noel De Martin

我也希望在任何地方都能使用自己的应用。以前坐火车、坐地铁，没有网络的时候，应用就不能用了，这让我很烦。明明要做的事情根本不需要联网，应用却照样停在那里。思考这些问题时，我接触到了本地优先运动，很喜欢这种理念。所以，我的应用也都是本地优先的，这与我对应用的理想很契合。

My 3 Software Ideals

- Universality
- Sovereignty
 - Longevity
 - Privacy & Data Ownership
 - Untangle Network Effects

noeldemartin.com/local-first-2026

软件理想：自主权 · 画面 00:03:17

00:02:13 • Noel De Martin

第二个理想是自主权，也就是 sovereignty。它也是最初促使我接触 Solid 协议的原因，后面会再讲。自主权具体指什么？比如，软件要能长久使用。我希望自己的应用和软件能一直用下去。我开始自己做应用，一个主要原因就是受够了第三方应用突然不能用。有些应用我甚至付了费，但因为公司被收购，或者发生了别的事情，它们就开始变得越来越不好用。

00:02:50 • Noel De Martin

即使我喜欢这个应用，也在付费使用，开发方还是会改掉一些东西、删掉一些东西，最后我就没法继续用了。因此，我对软件还有一个理想：只要你愿意，就应该能一直用下去。另外还有隐私和数据所有权。我希望能决定自己的数据会被怎样处理，真正拥有自己的数据。最后，我也希望能解除网络效应带来的束缚。

00:03:21 • Noel De Martin

这也是我的一个愿望，不过我在这方面做得不多，因为我开发的并不是社交应用。说到社交软件，大多数时候，我还是得使用其他人都在用的平台。我有 Twitter，也就是 X 的账号，也有 Bluesky 和 Mastodon 的账号。但我没法决定只用自己做的社交网络，因为别人不会跟着来用。理想情况下，软件不应该被网络效应绑住。

让不同开发者与 AI 做出的应用自然互通

My 3 Software Ideals

- Universality
- Sovereignty
- Interoperability
 - Malleable Software
 - Interoperable Serendipity

noeldemartin.com/local-first-2026

软件理想：互操作性 · 画面 00:04:19

00:03:51 • Noel De Martin

人们应该能选择自己想用的软件，同时又能与其他人互通。这就引出了第三个理想：互操作性。即使你没听过 interoperability，可能也听过 malleable software，也就是可塑软件。我自己提出了一个说法，用来描述我希望达到的互操作状态：interoperable serendipity，可以理解为“不经意间实现的互操作”。我的定义是：不同开发者做出来的应用，不需要事先明确协调，就能协同工作。

Interoperable Serendipity happens when apps built by different developers work together without explicit coordination”

由不同开发者构建的应用无需预先协调即可协同工作 · 画面 00:04:42

00:04:30 • Noel De Martin

这是什么意思？比如，我正在用一个任务管理器，另一位开发者或者另一家公司又做了一个任务管理器，解决的是同一种需求。两者都是任务管理器，从功能上说，做的是同一件事。如果它们开箱即用地互通，就很好。有些应用确实能互操作，但往往是因为双方开发者用了同一套技术栈，或者同一种协议。

Interoperable Serendipity

“Interoperable Serendipity happens when apps built by different ~~developers~~ **AIs** work together without explicit coordination”

noeldemartin.com/local-first-2026

AI与意料之外的互操作 · 画面 00:06:11

00:05:01 • Noel De Martin

而且，大多数时候，他们是专门为此做了协调。我最终想实现的，是一种开箱即用就能协同工作的软件，大家不必特意考虑互操作性。这是我最初对这个词的描述。但到了今天，我觉得还需要加上一点：我们也希望用 AI 做出的应用能这样互通。现在，很多人会用 Lovable、v0，或者其他能够生成应用的 AI，来制作自己的应用。

00:05:40 • Noel De Martin

如果这些应用一做出来就能互操作，那就好了。我用 vibe coding 做一个任务管理器，你也用 vibe coding 做一个，两者最好能够互通。再进一步，即使我用同一个 AI 编程工具做出两个任务管理器，大多数时候，它们照样不能互操作。对我来说，这是一个值得解决的问题。我希望软件能以这种方式工作：一开始就能互通。



noeldemartin.com/solid-world-2025

互操作性进一步阅读 · 画面 00:06:23

00:06:15 • Noel De Martin

这个想法一直在引导我的许多工作。如果想了解它在技术上如何实现、又会带来哪些影响，可以看一下屏幕上的这场分享，我之前在里面讲得更详细。今天先给大家一个基本解释，概括一下它是怎么工作的。接下来要讲 Solid 协议，这是我选择用来解决互操作问题的技术。

Solid: 把应用与数据分开

The Solid Protocol



noeldemartin.com/local-first-2026

Solid协议 · 画面 00:06:40

What is Solid?

- A protocol for decentralized storage.
 - “Decouple apps from data”

noeldemartin.com/local-first-2026

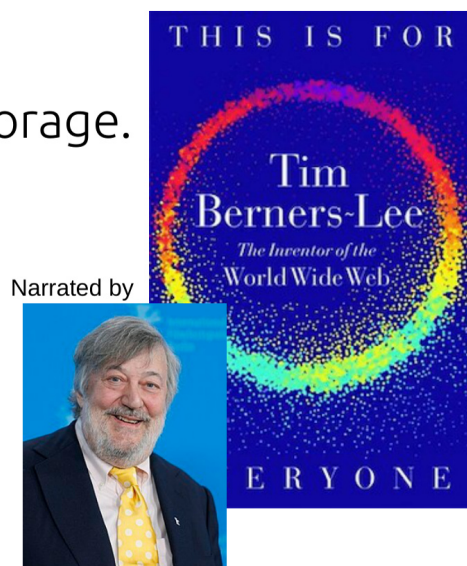
让应用与数据解耦 · 画面 00:07:00

00:06:47 • Noel De Martin

如果你还不了解 Solid：它是一个去中心化存储协议，核心想法是把应用和数据解耦，这也是谈论互操作性的另一种方式。Solid 由万维网的发明者 Tim Berners-Lee 提出，是 Web 的延伸。它并不是凭空冒出来、与 Web 完全无关的新协议，而是延续了他对 Web 应当如何工作的想法，把最初的 Web 理念进一步扩展到存储和数据所有权。

What is Solid?

- A protocol for decentralized storage.
- Created by Tim Berners-Lee.
...as an evolution of the Web.



noeldemartin.com/local-first-2026

Solid与Web的发展 • 画面 00:07:58

00:07:27 • Noel De Martin

如果对 Web 的历史和 Solid 感兴趣，我推荐屏幕上 Tim Berners-Lee 写的这本书。我记得好像是前两年出的，现在不太确定，总之是最近出版的。他在书里讲了自己如何构想出 Web，以及一直在思考什么、希望实现什么目标。如果不喜欢看书，或者更喜欢听播客，也可以听有声书，由 Stephen Fry 朗读。我觉得文字版和有声版都很好，对这个话题感兴趣的话，可以看看。

What is Solid?

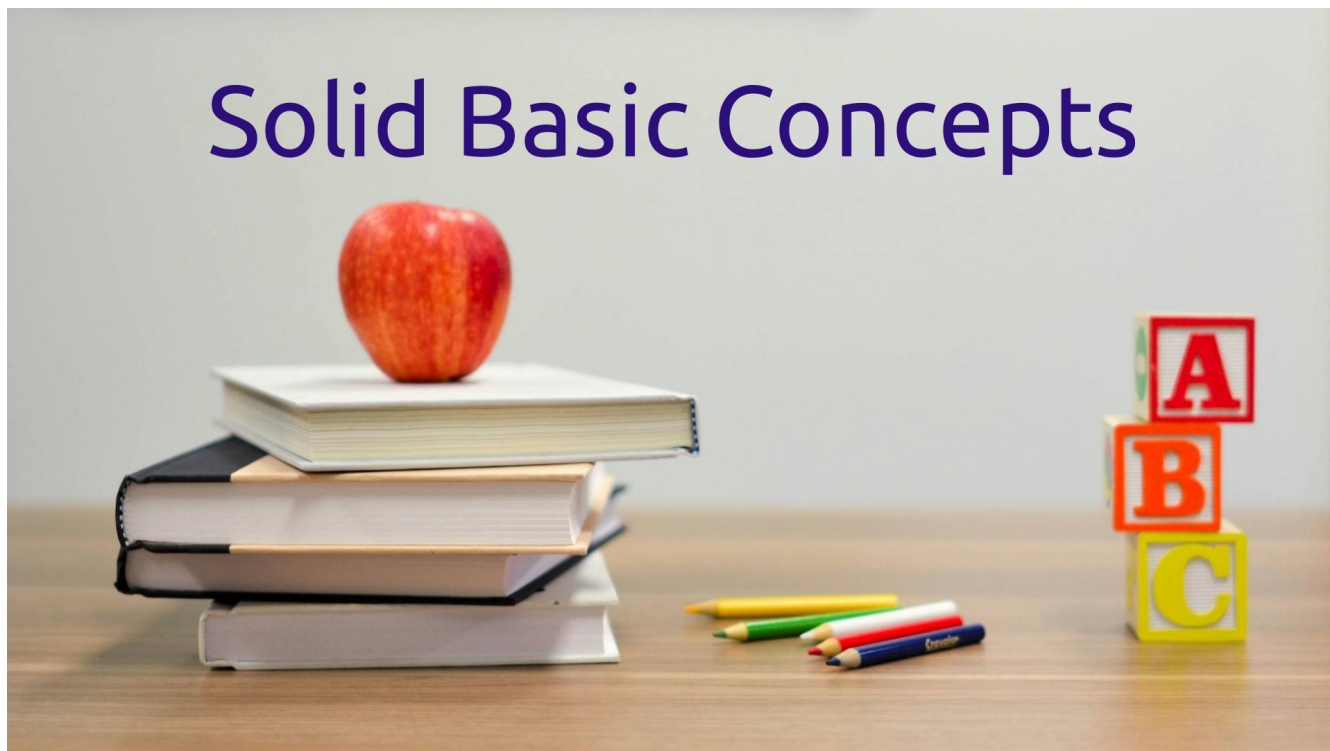
- A protocol for decentralized storage.
- Created by Tim Berners-Lee.
- Learn more:
 - solidproject.org
 - [Why Solid?](#)

noeldemartin.com/local-first-2026

进一步了解Solid • 画面 00:08:13

00:08:03 • Noel De Martin

想进一步了解 Solid，也可以去官网 solidproject.org。几年前，我写过一篇叫《Why Solid》的文章，介绍为什么为自己的应用选择 Solid，里面谈了它的优缺点，以及它能做什么。接下来，我会解释几个基本概念，让大家了解它在技术上如何工作。Solid 的核心概念是 Solid Pod。



Solid的基本概念 · 画面 00:08:28

Solid Basic Concepts

- PODs → Personal Object Datastore

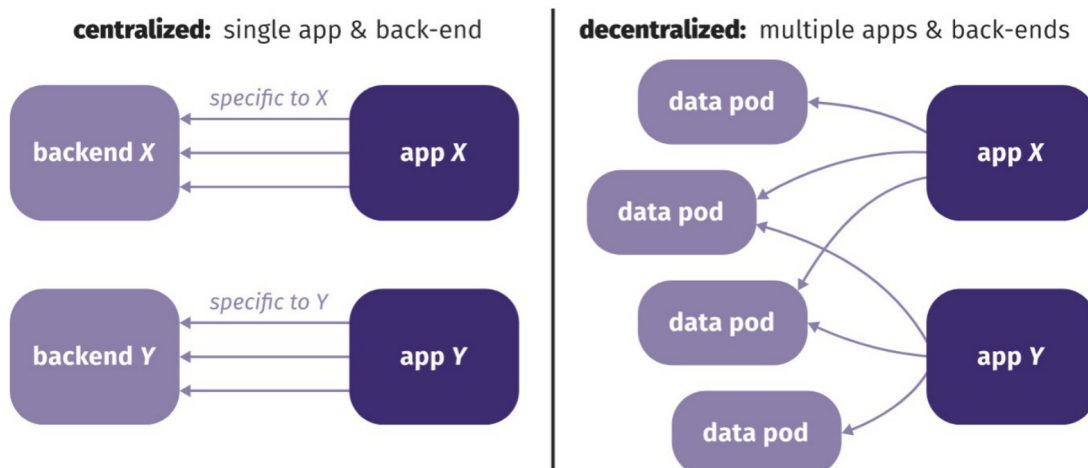


noeldemartin.com/local-first-2026

POD: 个人数据存储 · 画面 00:08:35

Solid Pod 是一个个人对象数据存储空间，也就是 Personal Object Datastore。可以想想 Dropbox、Nextcloud 或者其他云存储：基本想法就是，你有一个属于自己的数据存储空间。但这里不只有文件，还有语义数据，也就是软件能够理解的数据，而不只是一块块二进制内容。通常，我们熟悉的模式是：使用某个应用，数据就存进这个应用自己的后端。

Solid Basic Concepts



Ref: [Ruben Verborgh @ FOSDEM 2019](#)

noeldemartin.com/local-first-2026

应用与数据解耦前后 · 画面 00:09:59

比如，你往 Facebook 上传一张图片，再往 Twitter 或 X 上传一张，各个应用都有自己的图片存储。Solid 的想法是：东西存在 Pod 里，不同应用可以访问同一份数据。假设你有一个社交网络应用，通过它上传了图片，只要它使用的是 Solid 协议，换到另一个图片浏览器或图片管理应用里，就能看到、管理之前通过社交应用上传的所有图片。

Solid Basic Concepts

- PODs → Personal Object Datastore
- RDF → Resource Definition Framework
 - Data Graph + Custom “Ontologies” or “Vocabularies”
 - e.g schema.org
 - Semantic Web

noeldemartin.com/local-first-2026

RDF、图与词汇表 · 画面 00:10:44

00:09:51 • Noel De Martin

基本想法就是：数据属于你，应用只是处理这些数据的工具，两者不必绑在一起。这就是 Solid 的核心。技术上，它使用一种叫 RDF，也就是 Resource Definition Framework 的东西。它本质上是一张数据图，可以使用自定义的本体或词汇表。大家可能见过 schema.org，它常用于搜索引擎优化，也就是 SEO，但 RDF 的用途远不止 SEO。

编注：讲者口述及幻灯片均将 RDF 展开为 Resource Definition Framework。RDF 的标准全称是 Resource Description Framework（资源描述框架）。正文保留口述用词，此处单独说明术语更正。W3C《RDF 1.1 Concepts and Abstract Syntax》

00:10:28 • Noel De Martin

只是因为 SEO 这个用途，schema.org 变得更广为人知了。其基本思路是：用这些词汇表或本体定义不同类型的数据，应用就能知道怎样使用数据。这也叫语义网，你可能听过这个词。如前面所说，它建立在 Web 之上，并不是凭空出现的新协议，因此仍然使用大家熟悉的基本 HTTP 方法，比如 GET、POST 等。

Solid Basic Concepts

- PODs → Personal Object Datastore
- RDF → Resource Definition Framework
- Built on top of the Web (GET, POST, etc.)
- Permissions, Authentication, etc.

noeldemartin.com/local-first-2026

Solid基于Web构建 • 画面 00:11:02



noeldemartin.com/fosdem

Solid入门资料 • 画面 00:11:18

00:11:00 • Noel De Martin

它在 Web 上增加了一些功能，比如权限、身份验证，以及开发应用所需要的其他能力。如果想深入了解 Solid 是怎么工作的，可以看我几年前做的这场分享，里面从开发者的角度展开了技术细节。我几年前就开始做 Solid 相关工作了，那时大语言模型和 AI 还没有像今天这么流行。

AI 时代，我们还需要应用吗？

The Age of AI

- Generative AI
- Vibe Coding
- AI Agents
- ... but where are all the apps!?
- ... do we even want apps!?



noeldemartin.com/local-first-2026

AI时代，我们究竟想要什么应用 · 画面 00:13:10

00:11:38 • Noel De Martin

但今天，我们进入了另一种情境，也就是 AI 时代。似乎什么事情都有人用 AI 来做：生成式 AI、vibe coding、AI 智能体。软件的工作方式正在改变，理论上，制作软件比过去任何时候都容易。不过，看看 AI 真正产出了什么，再看看行业里发生的事情，至少就我个人而言，并没有发现自己开始使用大量新的应用。

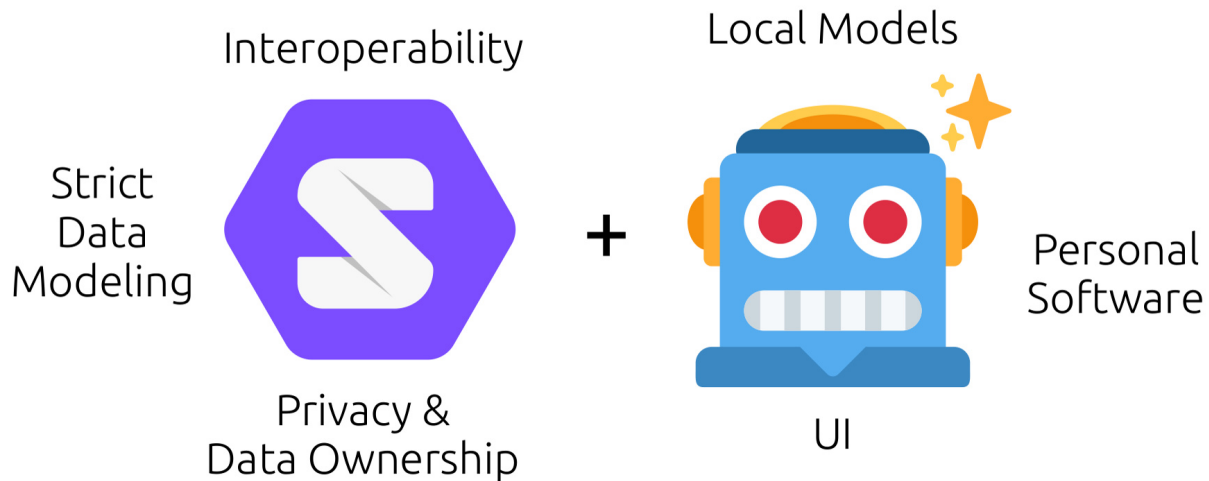
00:12:10 • Noel De Martin

确实，有很多人在讨论新应用，说自己五分钟就做出了一个。但我个人并不觉得，现在找到了比以前更多自己喜欢的应用。从我的角度看，情况与 AI 普及之前相差不大。我觉得不少人会同意，当然，你也可能不同意。不过，还有人跟我说，也许我们根本就不需要应用了。对于这个观点，我也能理解。

00:12:44 • Noel De Martin

也许没有更多新应用，是因为有了 AI，就不再需要应用了。你可以直接让 AI 完成目标、处理任务，那应用还有什么意义？我在一定程度上赞同，但我个人还是喜欢有应用可用，这也是我仍然在开发应用的原因。只是，我希望应用遵循一种不同的模式，也就是开头提到的三个理想。

Solid to the rescue?



noeldemartin.com/local-first-2026

Solid与AI如何结合 · 画面 00:14:40

00:13:15 • Noel De Martin

我认为，要实现这一点，也要让 AI 发挥更大作用，可以分成两层。第一层使用 Solid，或者你选择的其他协议，让你真正拥有数据，并保障数据隐私。这一层还要提供严格的数据建模。AI 虽然每天都在进步，但还是会产生幻觉。如果给它明确、严格的数据模型，而且你清楚自己有哪些数据、想使用哪些数据，AI 就更容易在此基础上构建东西。

00:13:53 • Noel De Martin

前面也说过，如果我们用 AI 做出来的东西都能直接互操作，那就很好。我认为，这正是 Solid 能带来的东西，也是它今天仍然有意义的原因，甚至可以说比过去更重要。同时，我们可以利用 AI 的新功能和新能力，扩展这个基本想法。比如，在使用 Solid 的基础上，用本地模型来处理自己的数据。

00:14:26 • Noel De Martin

我们还可以让 AI 根据已经定义好的数据结构和 schema 来构建用户界面，这是 AI 可以发挥作用的地方。如果给它数据 schema，并且复用已有的身份验证等功能，它产生幻觉的机会就会少一些，结果也会更好。另外，大家也经常谈到个人软件：很多人说，现在有了 vibe coding 和 AI，已经开始为自己制作软件。

Ànima: 把 Solid 与 AI 接在一起



Inspired by “Charlie, the AI that works for you”

👉 See w3.org/DesignIssues/Charlie.html

noeldemartin.com/local-first-2026

Ànima与Charlie的启发 · 画面 00:16:01

00:14:58 • Noel De Martin

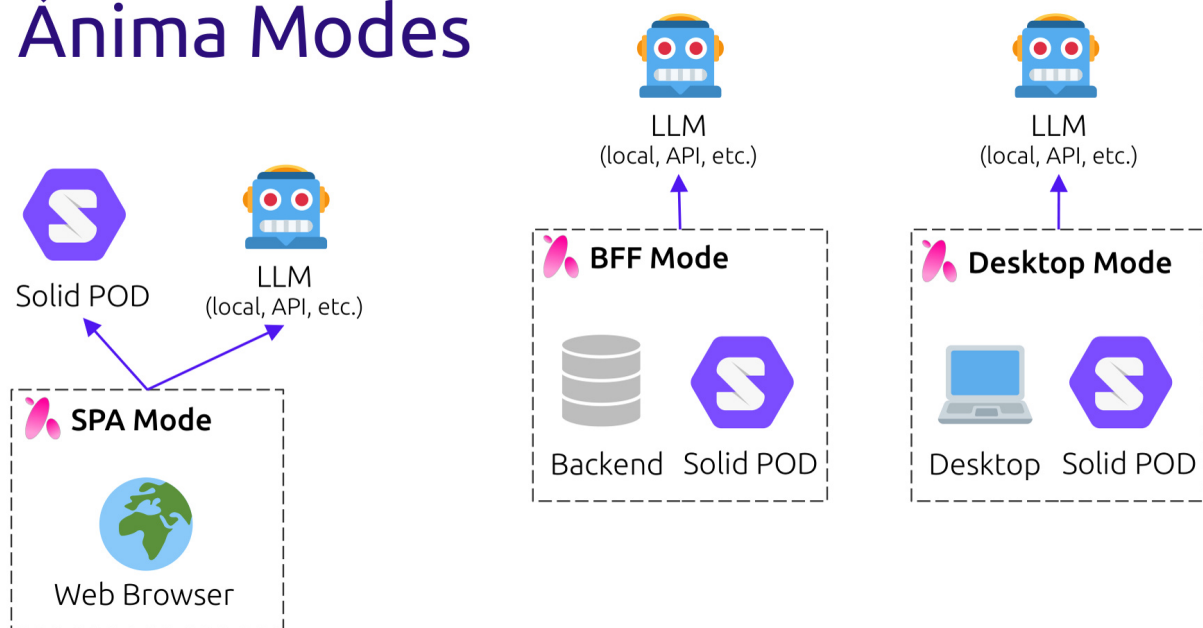
我觉得这个想法很好，也很适合使用 Solid。如果所有东西都建立在 Solid，或者遵循这些理念的技术之上，情况会更好，因为你通过 vibe coding 做出的应用，其数据都可以继续复用。你不必每做一个新应用、每开始一个新项目，都从零开始。数据可以慢慢积累，并持续使用。Ànima 是我这几个月一直在做的项目，也是我对这个问题的回答：怎样结合 Solid 和 AI，实现前面讲的这些事情。

00:15:40 • Noel De Martin

这个项目很大程度上受到 Tim Berners-Lee 另一个想法的启发，叫作 Charlie。他几年前写过一篇文章，设想了一个为你工作的 AI。它不是第三方公司拥有、服务于那家公司利益的 AI，而是为你工作、遵循你的利益。我认为，本地模型加上 Solid，正好适合实现这种场景。那么，今天的 Ànima 是怎样工作的呢？

三种运行模式：SPA、BFF 与桌面

Ànima Modes



noeldemartin.com/local-first-2026

Ànima的三种运行模式 · 画面 00:18:03

00:16:14 · Noel De Martin

上一页提到，它还在开发中，不过下面这些已经可以运行。我的想法是提供三种模式。第一种叫 SPA 模式，也就是单页应用。现在网址上部署的就是这种模式，访问 anima.noeldemartin.com 就可以使用。Ànima 在浏览器里运行，没有自己的后端，甚至可以离线运行。登录 Solid Pod 之后，应用产生的所有聊天记录和数据都会保存在这个 Pod 里。

00:16:53 · Noel De Martin

你也可以把它连接到外部的大语言模型。可以是笔记本电脑上运行的本地 AI，也可以是 Anthropic、Gemini 之类的服务，选择你想用的模型即可。这是一种容易上手、方便了解工作方式的模式。不过，还有一些我觉得更适合实际使用、也更想自己使用的模式。第二种叫 BFF，也就是 backend for frontend，基本思路是让 Ànima 运行在服务器上。

00:17:26 · Noel De Martin

既然这种模式有服务器，就可以利用它，直接内置一个 Solid Pod。Solid 目前的一个主要问题，是人们没有 Solid 账号，或者不知道该去哪里注册。借助 Ànima，就可以拥有自己托管的 Pod，并且与 AI 以及前面讲的能力集成在一起。它同样可以连接外部的大语言模型。基本想法与前一种一样，只不过 Pod 集成在后端里。

00:18:01 · Noel De Martin

最后一种是我最近在做、也觉得很有潜力的桌面模式。你下载 Ànima 的可执行应用，在自己的电脑上运行。上一种 BFF 模式需要服务器，或者需要自己托管服务，多数人不会去做，也不知道怎么做。而桌面模式只需要一台电脑。装好桌面应用之后，只要电脑在运行，Ànima 就能和 Solid Pod 一起工作，你可以自行决定 Pod 放在哪里。

已经实现的能力与实验性路线图

Ànima “Roadmap”

-  Model agnostic (Vercel's AI SDK)
-  Read Solid POD
-  Built-in Solid POD (BFF/Desktop Modes)
-  External reachability (Desktop Mode)
-  Write to Solid POD
-  Create and serve Apps/Views
-  Agentic Loop
-  Scheduled and background Jobs
-  3rd-party chat integration (Telegram, WeChat, etc.)
-  Plugins / Tools

noeldemartin.com/local-first-2026

Ànima 的开发计划 • 画面 00:21:14

00:18:41 • Noel De Martin

目前就是这三种模式，都已经能运行。不过，整个项目还很实验性，我没有做很多测试。按理说是可以工作的，至少我试过的部分能用。Ànima 接下来有什么路线图？这里的“路线图”要加引号，因为我不确定会不会把里面的事情都做完，它只是我现在的一些想法。已经实现的能力，首先是不绑定某个模型。它使用 Vercel 的 AI SDK，因此这个 SDK 支持的模型，它也能够支持。

00:19:15 • Noel De Martin

也就是大多数模型。它还可以读取 Solid Pod：连接 Solid 之后，就能与 Pod 交互。BFF 模式和桌面模式也已经可以提供内置的 Solid Pod。这些都是今天已经能用的功能。我正在做、或者正在考虑的一件事，是让它能够从外部访问。对许多人来说，桌面模式可能更有吸引力，因为不是每个人都有服务器，也不是每个人都愿意自己托管服务。

00:19:49 • Noel De Martin

安装一个桌面应用很容易，但问题是，电脑之外访问不到它。所以，我在考虑使用 Tailscale 或者 HTTP 隧道一类的方式，让你可以按自己的意愿，把笔记本上运行的 Ànima 开放到互联网上。另外，目前它只读取 Pod。实现写入能力并不难，但在动手之前，我想先考虑清楚，因为 Solid 的设想是承载你所有的信息。

00:20:22 • Noel De Martin

所以，把 Solid Pod 的写入权限交给 AI 之前，需要很谨慎。这件事我确实想做，但要先想好怎么做才合适。另外，我想让它能够创建、提供应用，或者我称为“视图”的东西。你不一定每次都需要一个新应用，有时只是想用一个小界面查看已有数据。结合 Solid 和 AI，我觉得可以实现这样的东西。

00:20:55 • Noel De Martin

这也更接近有些人所说的“不想要应用”：也许他们想要的是视图。毕竟，你并不总想通过聊天机器人来操作数据。因此，我希望赋予 Ànima 创建应用或者视图的能力。最后还有一些事情，也许以后会做，但我不确定，也不作承诺。其中一个智能体循环。现在，你跟 Ànima 说一句话，它回复一次，到这里就结束了。

00:21:29 • Noel De Martin

但对很多任务来说，循环是有用的：你给它一个目标，它就一直做，直到判断目标已经完成。还有后台任务调度，这也是 AI 可以做的实用功能；如果一切都与 Solid 集成，能力可能会更强。再一个是接入第三方聊天工具。现在只能通过网站上的聊天界面与 Ànima 对话，如果也能从不同的聊天应用里跟它说话，就会更方便。

现场演示：Focus、本地 Pod 与 Ànima

DEMO

noeldemartin.com/local-first-2026

现场演示 • 画面 00:22:16

00:22:03 • Noel De Martin

最后还有插件和工具，让大家按自己的需求扩展功能。这就是我对 Ànima 的基本设想。前面讲了这么多，接下来实际演示一下。这里是我做的应用 Focus，一个任务管理器。这边运行着 Ànima。我在这台电脑上安装了 Ànima 的桌面模式，因此现在有一个本地运行的 Pod。

Press "+" to add a new task

☐ Install Ànima

Focus中的本地任务 · 画面 00:23:10

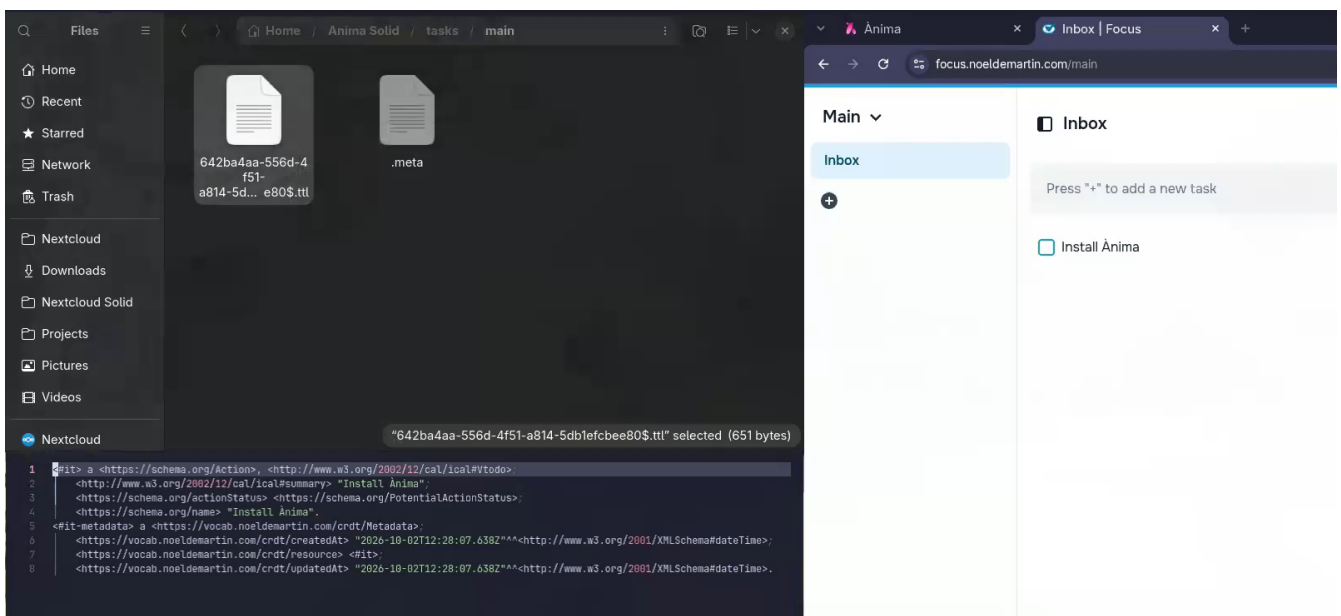
00:22:36 · Noel De Martin

先用一下 Focus 这个任务管理器。我们添加一个任务：“安装 Ànima”。大家也能看出来，这个演示我之前排练过。现在，任务管理器里有了一条待办事项。它是本地优先的，你可能注意到了，我没有登录，也没做别的操作，所有数据都在本地。我的所有应用都可以不注册账号，一直这样用下去，数据全部保存在你的设备上。

00:23:09 · Noel De Martin

但到了某个时候，你可能想在设备之间同步数据，或者把它安全地存进云端，避免丢失。准备好之后，就可以连接账号，具体方式是连接一个 Solid Pod。任何 Solid Pod 都可以，并不局限于 Ànima 或者我提供的 Pod，因为应用遵循的是 Solid 协议。这里我们登录这台电脑上运行的 Pod，也就是刚才安装好的那个。先复制本地 Ànima 实例的网址。

登录、同步与可读的 Turtle 文件



任务与本地Solid文件同步 · 画面 00:24:30

00:23:43 • Noel De Martin

接着会跳回 Ànima，让你登录并授予权限。我先登录，再允许访问。它询问是否要备份数据，我们选“是”。现在已经完成同步，数据存进了 Solid Pod。我给大家看看具体是什么样。这里是我电脑上的一个文件夹，安装 Ànima 的时候，我指定了用哪个文件夹来存数据。这就是 Pod 里的全部数据。目前东西不多，只有个人资料、一些设置，以及任务。

00:24:26 • Noel De Martin

这些就是刚刚创建出来的内容。打开文件，就能看到 Solid 的工作方式。可以把它想成类似 Obsidian 那样的应用：一组 Markdown 文件就是实际的数据来源，你也可以直接修改 Markdown。Solid 的想法类似。这里的数据放在 Turtle 文件里，Turtle 是 RDF 的一种格式，人可以读懂，至少开发者比较容易读懂。

00:24:59 • Noel De Martin

你可以直接打开文件，查看里面的内容。比如，我们把任务标记为完成，应用就会同步，这个文件也会更新。可以看到，它已经更新了，任务现在是完成状态。另外，因为使用了 CRDT，它还增加了一些数据。我的应用是本地优先的，每次修改都会产生相应的 CRDT 数据。现在，这些内容已经在 Pod 里了。

让 AI 读取任务数据

00:25:29 • Noel De Martin

应用已经连上了 Pod，来看看 Ànima 能做什么。前面说过，它可以使用不同的模型。我这里也有几个本地运行的模型，不过因为电脑内存不多，效果不是很好。所以，这次演示使用 Anthropic 的 Sonnet。你也可以换成本地模型或者其他第三方模型。我们问 Ànima 一个关于 Pod 的问题：“我目前有哪些尚未完成任务？”

Hello <http://localhost:1191/pod/noeldemartin/profile/card#me>, how can I help you today?

10/2/2026, 2:31:33 PM

Which tasks do I have currently open?

10/2/2026, 2:31:33 PM - Anthropic / Sonnet 5

➤ Read type index

...

Ànima读取个人数据 • 画面 00:26:18

00:26:03 • Noel De Martin

看得重新登录一下。“我有哪些任务？”对，可能是这个页面开了一段时间，会话过期了。发出消息之后，因为 Ànima 已经连接到 Pod，它就能读取数据。可以看到，它回答说：你有一个“安装 Ànima”的任务，但已经完成了，所以目前没有未完成任务。现在我在这里新建一个，比如“学习中文”。它会同步，因此这条任务也进入 Pod 了。大家可以看到，新文件已经创建出来。

I checked your tasks, and it looks like you only have one task recorded — "Install Ànima" — but it's already marked as **completed** (finished on October 2, 2026).

You currently have **no open tasks**. 🎉

10/2/2026, 2:32:14 PM

Can you take another look, do we have any new tasks?

10/2/2026, 2:32:14 PM - Anthropic / Sonnet 5

➤ Listed files from <http://localhost:1191/pod/noeldemartin/tasks/main/>

There is a new one! Let me check its details.

➤ Read <http://localhost:1191/pod/noeldemartin/tasks/main/cc093f82-db0e-4b51-a5ae-1727607155c2>

Yes, you've got a new one! 🎉

"**Learn Chinese**" — this task is currently **open** (not yet started/completed), created on October 2, 2026.

So to sum up, your open tasks now are:

- **Learn Chinese** (open)

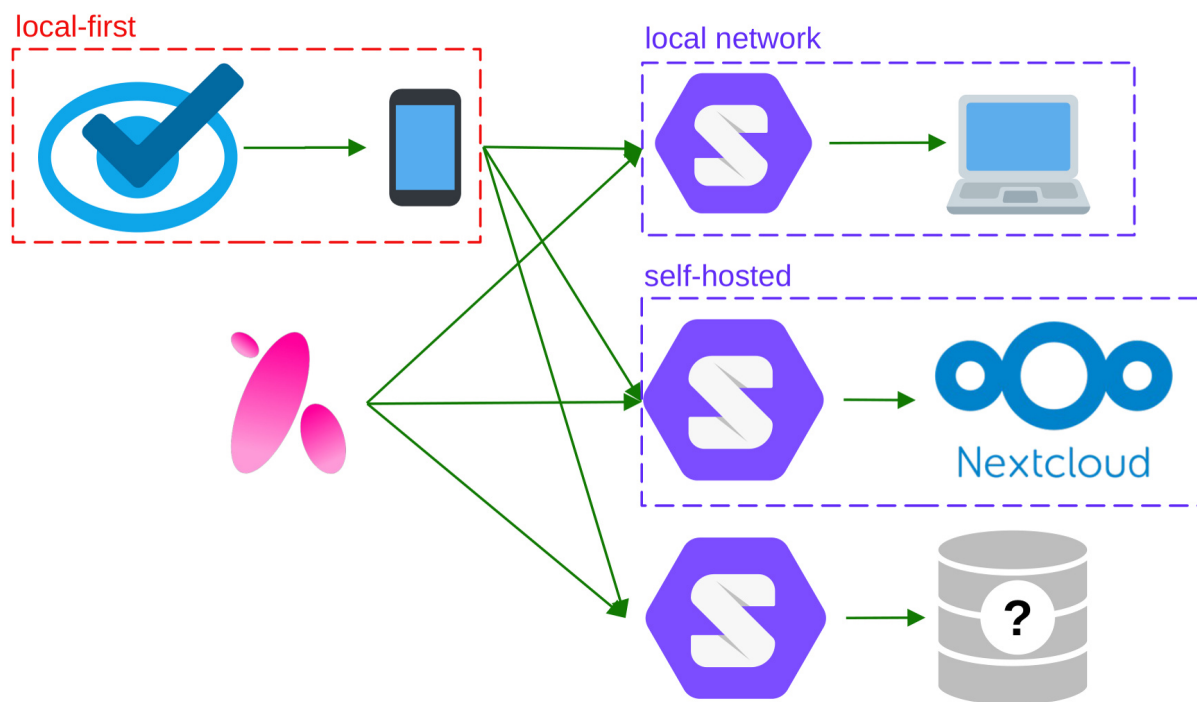
Your "Install Ànima" task is still completed, so no action needed there. Want me to help break down the Chinese learning goal into smaller steps, or set any details for it?

Ànima根据个人数据回答问题 · 画面 00:27:18

00:26:43 • Noel De Martin

现在有两条任务了。我可以再问它：“你能再看一下吗？有没有新的任务？”它应该会重新读取数据，找到刚才新增的任务。对，它现在说有一条“学习中文”，而且尚未完成。基本想法就是，你照常使用 AI，模型也由自己选择，但它连接到了你的 Solid Pod，因此能够访问你的数据。

隐私成立的条件，以及数据怎样流动



本地优先、本地网络与自托管 · 画面 00:28:48

00:27:14 • Noel De Martin

这次我使用的是 Anthropic，也就是第三方模型。但如果只使用本地模型，或者自己控制的模型，一切就可以完全私密。实际上，断开互联网也能工作，因为所有东西都可以在本地运行。这就是 Ànima 和 Solid 应用的基本工作方式。回顾一下刚才的过程：首先，我们开始使用 Solid Focus。它是本地优先的，可以完全在自己的设备上使用。

00:27:46 • Noel De Martin

你甚至不需要创建账号，想这样使用我的应用，完全可以。不过，到某个时候，你可能想同步数据，或者把它存到别的地方。在刚才的演示里，我们用的是本地网络，一切都发生在 localhost，数据存进了笔记本电脑上的 Pod。接着打开 Ànima，它就能与这个 Pod 通信。这里的 Pod 也是 Ànima 提供的，但也可以是其他任何 Pod。

选择存储：以 Nextcloud 为例

00:28:18 • Noel De Martin

Ànima 可以连接任何 Solid Pod。比如，在我自己实际使用的环境里，有一个自己托管的 Pod，底层把数据存进 Nextcloud。我就是这样处理自己的数据的。因为两者都是 Solid Pod，我的所有应用，包括 Ànima、Focus 等，也都能连接到那个 Pod。因此，我在实际环境中使用 Nextcloud 存储数据。Solid 是一个协议，所以底层存储可以由你选择。

不过，取决于你想用什么存储，可能还没有对应的适配器，需要自己实现。就我的体验而言，这并不算难，比如 Nextcloud 的适配器就是我自己写的。我的 Solid Pod 实现也可以在 GitHub 上看到。Solid 有一个优点，就是 API 不太复杂。这就是整体工作方式。如果你想自己开发应用，或者了解我的应用是怎么做出来的，下面有两个我一直在开发的库。

Soukai: 本地优先的数据层

My Tools



Framework agnostic

TypeScript

Models
(Active Record)

Engines
(Solid POD, IndexedDB, ...)

Interoperability
(Type Indexes, Schemas, RDF aliases)

Local-first
(CRDTs)

noeldemartin.com/local-first-2026

Soukai的数据模型与同步工具 • 画面 00:30:50

00:29:31 • Noel De Martin

第一个是 SoukaiJS。它不依赖特定框架，是一个 TypeScript 库。Ànima 的服务器端和前端都在用它。它用纯 TypeScript 编写，所以只要能运行 JavaScript，就能使用这个库，没有其他要求。因为是 TypeScript，也有完整的类型定义。它采用的是 Active Record 模式。

00:30:01 • Noel De Martin

如果用过 Ruby on Rails 或 Laravel，你可能已经熟悉它。这是一种处理数据的编程模式，其中一个好处是可以替换存储引擎。我的应用默认使用 IndexedDB，因此在应用里写入的内容会先存在 IndexedDB 中。如果要同步到 Pod，就可以换一个引擎，这些工作可以自动完成。

00:30:33 • Noel De Martin

你的代码不必知道数据究竟存进了 Pod，还是 IndexedDB，这就是 Active Record 的作用。它也提供了配合 Solid 实现互操作的功能。前面提过，它是本地优先的，因此也内置了 CRDT，不需要自己再实现一遍。另一个正在做的工具叫 Aerogel。它的实验性质更强，完成度也低一些，不过我的所有应用已经都在使用它。

Aerogel: 完整的应用开发框架

My Tools

Opinionated
(VueJS + TailwindCSS + Soukai)

UI Components
(Login, local-first, ...)

Authentication

Error handling

Cloud Sync

Aerogel The Lightest Solid

Work in Progress!

aerogel.js.org

noeldemartin.com/local-first-2026

Aerogel应用框架 • 画面 00:31:57

00:31:10 • Noel De Martin

Aerogel 是一个完整的应用开发框架。Soukai 只负责数据层，Aerogel 则用来构建整个应用。它有自己的技术选型，使用 Vue.js、Tailwind CSS，以及负责数据的 Soukai。框架包含许多界面组件。比如，通过 Solid 登录、显示数据同步界面，这些都已经做好了，不必重新实现。它也内置了身份验证，以及部分错误处理。

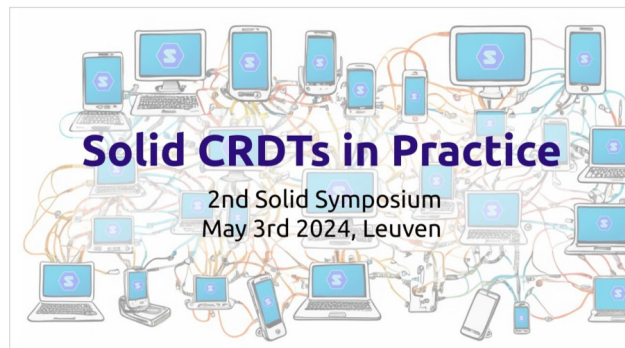
00:31:41 • Noel De Martin

有些错误是 Solid 特有的，框架会处理这些问题，向用户显示合适的提示，或者采取相应措施。云端同步也已经实现了。使用 Aerogel 时，不必自己操心数据同步，相关机制会自动运行，当然也可以配置。如果想进一步了解这些工具，我以前做过几场分享。

My Tools



noeldemartin.com/solid-symposium-dx



noeldemartin.com/solid-symposium-crdts

noeldemartin.com/local-first-2026

开发体验与CRDT的延伸资料 • 画面 00:32:10

00:32:12 • Noel De Martin

大家可以看这些分享，继续深入了解。其中一场讲开发 Solid 应用的开发者体验，另一场讲 CRDT：它在这个库里如何实现，以及我怎样把它与 Solid 一起使用。主要内容就是这些。大家可以在线上关注我。前面提过，我会在自己的网站上记录正在做的事情，也可以关注开发日志，了解这些项目的进展。我的分享就到这里，谢谢。

See you online!



Follow my work at
noeldemartin.com/now

You can also find me on the fediverse,
Bluesky, Twitter/X, etc.

noeldemartin.com/local-first-2026

继续关注Noel的开发 • 画面 00:32:28

00:32:46 • Noel De Martin

我先停止共享屏幕。

00:32:50 • Noel De Martin

非常感谢。

现场问答

00:32:53 • 主持人

好的。有问题吗？有没有人有什么问题？

00:33:06 • Noel De Martin

希望刚才讲清楚了，不知道是不是讲得有点快。

00:33:11 • 主持人

我觉得很清楚。

Solid 本身提供同步吗？

00:33:15 • 听众

你好，我可以问个问题吗？我的理解是，你在做本地优先应用，并且把 Solid 用作协调层和存储层，是这样吗？

00:33:34 • Noel De Martin

对，这些应用是本地优先的，所有内容都在浏览器里，通过 Solid 来同步数据。我实现了一个能与 Solid 配合使用的 CRDT 算法。如果想进一步了解，可以看我专门讲它的那场分享，也就是最后列出来的那个。

00:33:52 · 听众

Solid 协议本身提供客户端与 Pod 服务器之间的实时同步吗？

00:34:11 · Noel De Martin

Solid 本身并不直接提供同步。如前面所说，它像一个数据库，或者像 Dropbox，里面是一组文件。我是在 Solid 之上构建了这个 CRDT。Solid 自身允许你读取、写入数据，但处理的是通用数据。这就像有一个支持 SQL 的数据库：你可以在 SQL 之上构建 CRDT，也可以做一个普通的、并非本地优先的应用。

00:34:43 · Noel De Martin

在这里，Solid 只承担数据库或文件存储的角色。同步是我在自己的库里实现的。

RDF 业务数据与 CRDT 操作如何共存

00:34:53 · 听众

在 Solid 里，RDF 描述的是所存数据的结构或者语义。但如果使用 CRDT，传输的数据更像片段，或者增量操作。那么，你这儿的 RDF 描述的是 CRDT，还是数据本身的 schema？

00:35:32 · Noel De Martin

我为 CRDT 创建了一套 RDF 词汇表，所以 CRDT 本身也用 RDF 描述。Solid Pod 并不知道 CRDT 是什么，它只知道我往里面写了 RDF。我尽量让其他应用即使不是本地优先的，也能使用同一份数据。因为使用 CRDT 或自定义词汇表时有一个问题：如果做得太特殊，别的应用就看不懂这些数据，除非也使用同一个库。

00:36:16 · Noel De Martin

我的实现方式是：基础数据仍然是普通 RDF。比如创建一个任务，它就是一个 schema.org 的 Action；在此基础上，再生成 CRDT 操作。我重新共享一下屏幕，给大家看刚才那条任务。打开之后，可以看到这就是 RDF，基础数据是一个 schema.org 的 Action。

```
1 <#it> a <https://schema.org/Action>, <http://www.w3.org/2002/12/cal/ical#Vtodo>.  
2 <http://www.w3.org/2002/12/cal/ical#summary> "Install Anima";  
3 <https://schema.org/actionStatus> <https://schema.org/CompletedActionStatus>;  
4 <https://schema.org/name> "Install Anima";  
5 <http://www.w3.org/2002/12/cal/ical#completed> "2026-10-02T12:30:27.425Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>.  
6 <#it-metaData> a <https://vocab.noeldemartin.com/crdt/MetaData>.  
7 <https://vocab.noeldemartin.com/crdt/createdAt> "2026-10-02T12:28:07.638Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;  
8 <https://vocab.noeldemartin.com/crdt/resource> <#it>;  
9 <https://vocab.noeldemartin.com/crdt/updatedAt> "2026-10-02T12:30:27.436Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>.  
10 <#it-operation-2678f654-637e-4ebe-81e0-421c1ffc2619> a <https://vocab.noeldemartin.com/crdt/SetPropertyOperation>;  
11 <https://vocab.noeldemartin.com/crdt/resource> <#it>;  
12 <https://vocab.noeldemartin.com/crdt/date> "2026-10-02T12:28:07.638Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;  
13 <https://vocab.noeldemartin.com/crdt/property> <https://schema.org/name>;  
14 <https://vocab.noeldemartin.com/crdt/value> "Install Anima";  
15 <#it-operation-666e4d6f-de76-4385-a7ab-a1aaf013581> a <https://vocab.noeldemartin.com/crdt/SetPropertyOperation>;  
16 <https://vocab.noeldemartin.com/crdt/resource> <#it>;  
17 <https://vocab.noeldemartin.com/crdt/date> "2026-10-02T12:28:07.638Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;  
18 <https://vocab.noeldemartin.com/crdt/property> <http://www.w3.org/2002/12/cal/ical#summary>;  
19 <https://vocab.noeldemartin.com/crdt/value> "Install Anima".  
20 <#it-operation-ce13565e-c5b4-40ba-95d1-42793e7cb604> a <https://vocab.noeldemartin.com/crdt/SetPropertyOperation>;  
21 <https://vocab.noeldemartin.com/crdt/resource> <#it>;  
22 <https://vocab.noeldemartin.com/crdt/date> "2026-10-02T12:28:07.638Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;  
23 <https://vocab.noeldemartin.com/crdt/property> <https://schema.org/actionStatus>;  
24 <https://vocab.noeldemartin.com/crdt/value> <https://schema.org/PotentialActionStatus>.  
25 <#it-operation-0515ee6b-750f-4620-9d69-f97fb6684d96> a <https://vocab.noeldemartin.com/crdt/SetPropertyOperation>;  
26 <https://vocab.noeldemartin.com/crdt/resource> <#it>;  
27 <https://vocab.noeldemartin.com/crdt/date> "2026-10-02T12:30:27.436Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;  
28 <https://vocab.noeldemartin.com/crdt/property> <http://www.w3.org/2002/12/cal/ical#completed>;  
29 <https://vocab.noeldemartin.com/crdt/value> "2026-10-02T12:30:27.425Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>.  
30 <#it-operation-a02f19dc-3cce-4261-b025-c41df4da2d3c> a <https://vocab.noeldemartin.com/crdt/SetPropertyOperation>;  
31 <https://vocab.noeldemartin.com/crdt/resource> <#it>;  
32 <https://vocab.noeldemartin.com/crdt/date> "2026-10-02T12:30:27.436Z"^^<http://www.w3.org/2001/XMLSchema#dateTime>;  
33 <https://vocab.noeldemartin.com/crdt/property> <https://schema.org/actionStatus>;  
34 <https://vocab.noeldemartin.com/crdt/value> <https://schema.org/CompletedActionStatus>.
```

RDF 文件中的任务状态与 CRDT 操作 · 画面 00:37:00

00:36:46 • Noel De Martin

这应该是任何 Solid 应用都能理解的东西，不需要知道 CRDT 是什么。这里同时也有操作记录，使用的是我创建的 CRDT 词汇表。如果你用 Solid 开发任务管理器，很可能会选择 schema.org，因为它是最常见、大家也比较熟悉的本体之一。所以，即使不知道我的应用在做什么，也不了解 CRDT，你仍然应该能理解这部分数据。

00:37:22 • Noel De Martin

我就是通过这样的实现，让它也能与其他应用互操作。CRDT 操作会一并写进文档。如果想了解更多细节，可以看这场《Solid CRDTs in Practice》，里面解释了整套机制是如何工作的。

00:37:48 • 听众

哇，太棒了。

00:37:51 • Noel De Martin

谢谢。还有其他问题、想法，或者想聊的内容吗？

协议稳定为什么会成为优点

00:38:01 • 主持人

我想问一个关于 Solid 的问题。Solid 已经存在、发展了很长时间，可能快十年了。我想它背后一定有不少有趣的想法和功能。就你个人而言，在 Solid 之上开发应用，最喜欢的是什么？

00:38:32 • Noel De Martin

对，Solid 已经存在一段时间了。如果看看它的历史、它是怎么形成的，就会发现它并不是一夜之间出现的，而是 Tim Berners-Lee 对 Web 的想法不断演进的结果。他先创造了 Web，之后提出与 Web 配合的语义数据，延续的是同一种理念，再后来用 Solid 来交换、编辑和存储语义数据。

00:39:04 • Noel De Martin

所以，Solid 究竟从什么时候开始算，界线有点模糊，并不是非黑即白的。但与此同时，我也会说，Solid 是一个很稳定的东西。坦白讲，一开始这反而让我有点失望。我刚使用它的时候，第一印象是：Tim Berners-Lee 做了一个新的协议，要来解决数据存储这些事情。

00:39:35 • Noel De Martin

既然是新东西，我就以为它会不断进步、改变，增加很多新功能。但实际情况是，我做 Solid 相关工作的这些年，协议层面几乎没有发生什么新变化。新的项目、参与者、库确实很多，但 Solid 协议本身没有演进太多。最初我有点失望，现在却觉得这是一件好事，因为我开始把 Solid 看成类似 RSS 的东西。

00:40:09 • Noel De Martin

RSS 订阅源的规范很多年没变了，这也是一件好事，因为很多应用因此能兼容 RSS。后来有了播客应用，大家都在做播客，但它们仍然基于最基本的 RSS 规范运转。我眼中的 Solid，确实还是一个正在发展的协议。

00:40:40 • Noel De Martin

有些东西可能还会变化，但我不认为基本理念会改变太多。Solid 的基本概念，就是一个类似 Dropbox、Nextcloud 的存储空间，只不过里面的文件是 RDF。因此，它不只是文件，而是语义数据。这就是 Solid 的基石，我不觉得协议本身还会有特别大的变化。

00:41:11 • 主持人

非常感谢。我看到聊天框里有一个问题。

00:41:22 • Noel De Martin

好的。我想可能只有你能看到，因为我这里看不到。

为什么开始做：那些突然消失的应用

00:41:27 • 主持人

问题是：你最初是怎么接触到 Solid 的？什么促使你开始基于它做东西？

00:41:41 • Noel De Martin

这挺有意思，因为如果你想知道我最初怎么看 Solid，可以直接去我的网站读。从开始学习 Solid 起，我就在开发日志里记录了相关想法，你可以往前翻。事情大致是这样的：我原本在做一个相近的想法，叫作 autonomous data，也就是自主数据。那些突然不能用的应用实在让我很烦。比如，我之前用过 Wunderlist，一个后来被微软收购的任务管理器。

00:42:19 • Noel De Martin

我也用过 Evernote，更准确地说，是 Evernote Food，它相当于 Evernote 之上的一层，用来管理食谱。这两款应用后来都停用了。我非常不高兴，因为原来的应用对我已经够用了：我不需要新功能，也不希望改变什么。明明什么新功能都没要求，应用却会消失，我实在受够了这种事情。顺着这个想法，我开始做自己所说的“自主数据”。

00:42:54 • Noel De Martin

研究这个方向时，我发现了 Solid，它和我想做的事情很接近。于是，我决定停下自己的那套方案，转而沿着 Tim Berners-Lee 的思路做，因为它已经很接近我想实现的目标了。主要的不足是，当时的开发者体验不太好。这也是我后来自己实现 Soukai、Aerogel 这些库的原因：我对那时能用的库不太满意。

00:43:30 • Noel De Martin

现在可选的库多了，不过我仍然使用自己的，因为它们已经提供了我喜欢的开发体验。这就是我开始做这些事情的原因：我想解决应用突然消失的问题。

00:43:51 • 主持人

好的，非常感谢。还有其他听众想提问吗？大家想到的话，都可以提问题。

Focus、Umai 与 MediaKraken

00:44:19 • Noel De Martin

如果大家对这些应用感兴趣，我可以补充一点刚才没讲的内容。其中一个是 Focus，也就是任务管理器。另外两个，一个叫 Umai，是食谱管理器，我用它替代了 Evernote Food，主要就是管理做饭用的食谱。另一个叫 MediaKraken，是记录观影的网站。我把看过的、想看的电影都放在里面，管理自己的电影收藏。

00:44:53 • Noel De Martin

我在那里记录这些电影。目前，我正在重写 MediaKraken，希望它也能支持电视剧。如果现在去试用，你会发现它没有另外两款那么顺畅，因为它目前还是较老的版本，需要继续改进。如果你想试试这些应用，这就是它们各自的用途。

结束致谢

00:45:32 • 主持人

好的。还有其他问题吗？如果没有，那就谢谢 Noel。

00:45:53 • Noel De Martin

今天很愉快。谢谢你们组织活动，也谢谢大家参加。

Noel De Martin: <https://noeldemartin.com/> • Slides: <https://noeldemartin.com/local-first-2026/> • Solid: <https://solidproject.org/>