A Demonstration of the Solid Platform for Social Web Applications

面向社交 web 应用的实体平台演示

Essam Mansour[1] Andrei Vlad Sambra[1] Sandro Hawke[2] Maged Zereba[1]
Sarven Capadisli[2] Abdurrahman Ghanem[1] Ashraf Aboulnaga[1] Tim Berners-Lee[2,1]

Qatar Computing Research Institute, HBKU [2]
Decentralized Information Group, MIT CSAIL

ABSTRACT/诸论

Solid is a decentralized platform for social Web applications. In the Solid platform, users' data is managed independently of the applications that create and consume this data. Each user stores their data in a Web-accessible personal online datastore (or pod). Each user can have one or more pods from different pod providers, and can easily switch between providers. Applications access data in users' pods using well defined protocols, and a decentralized authentication and access control mechanism guarantees the privacy of the data. In this decentralized architecture, applications can operate on users' data wherever it is stored. Users control access to their data, and have the option to switch between applications at any time. We will demonstrate the utility of Solid and how it is experienced from the point of view of end users and application developers. For this, we will use a set of Solid servers and multiple Web applications that use these servers. We believe that experience with a concrete platform such as Solid is highly valuable in truly appreciating the power of a decentralized social Web.

solid 是社交 web 应用程序的分散平台。在 solid 平台中, 用户的数据独立于创建和使用此数据的应用程序进行管理。每个用户都将其数据存储在 web 可访问的个人在线数据存储区 (或 pod) 中。每个用户可以有一个或多个来自不同 pod 提供程序的 pod, 并且可以轻松地在提供程序之间切换。应用程序使用定义良好的协议访问用户 pod 中的数据, 分散的身份验证和访问控制机制保证了数据的隐私。在这种分散的体系结构中, 无论用户数据存储在何处, 应用程序都可以对其进行操作。用户控制对其数据的访问, 并可以随时在应用程序之间切换。我们将从最终用户和应用程序开发人员的角度演示 solid 的效用以及它是如何实现的。为此, 我们将使用一组 solid 服务器和使用这些服务器的多个 web 应用程序。我们相信, 使用 solid 这样的具体平台的经验对于真正了解分散的社交网络的力量非常有价值。

1. INTRODUCTION/介绍和背景

Social Web applications, such as Facebook, Twitter, Doodle, Wikipedia, Craigslist, and many more store data in centralized repositories that can be thought of as "data silos". Each application (or set of applications based on one social network platform) controls its own data and often has its own authentication and access control mechanisms. As a result, users cannot easily switch between similar applications that could allow reuse of their data, or switch from one data storage service to a different one. Developers are restricted to the data access APIs provided by a specific platform, and cannot easily develop applications that can run on multiple platforms. These and other problems of centralization have been recognized for a long time, and there have been many proposals for "re-decentralizing" the social Web such as Diaspora[1], Musubi [3], and WebBox [10], among others. None of these proposals has been widely adopted yet, and

社交网络应用程序, 如 Facebook, Twitter, Doodle, Wikipedia, Craigslist, 他们中多数将数据存储在可被视为"数据孤岛" 的集中存储存储库中。每个应用程序 (或基于一个社交网络平台的一组应用程序) 控制自己的数据, 并且通常有自己的身份验证和访问控制机制。因此, 用户无法轻松地在允许重用其数据的类似应用程序之间切换, 也无法从一个数据存储服务切换到另一个数据存储服务。开发人员仅限于特定平台提供的数据访问 api, 并且无法轻松开发可在多个平台上运行的应用程序。这些和其他集中化的问题已经认识到很长时间了, 有许多关于 "重新下放" 社会网络的建议, Diaspora1, Musubi [3], and WebBox [10], 等。但这些建议都尚未被广泛通过,

[1] https://diasporafoundation.org

[1] https://www.w3.org/Social/WG

ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2890529.

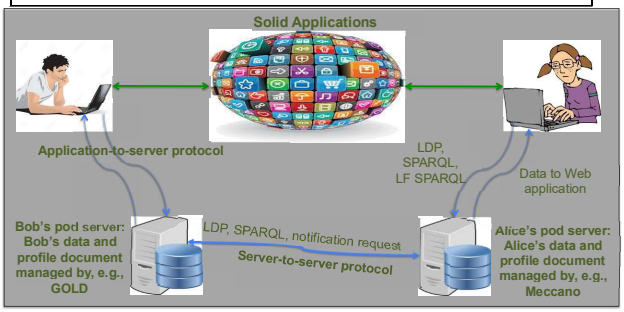

Figure 1: The Solid platform. A user stores their data in a personal online datastore (pod) that resides on a pod server. The user controls their identity using an RDF profile document stored in their pod. To use a Solid application, the user loads the application from an application provider. The application obtains the user's pod from the identity profile. It then follows links from the profile to discover data on the user's pod, as well as on other pods, performing authentication when needed.

Solid 平台。用户将他们的数据存储在个人在线数据存储（pod）中，该个人在线数据存储驻留在 pod 服务器上。用户使用存储在其 pod 中的 RDFprole 文档控制他们的身份。要使用 Solid 应用程序，用户从应用程序提供程序加载应用程序。应用程序从用户的身份信息获取用户的 pod。然后，它遵循身份信息的链接来发现用户 pod 以及其他 pod 上的数据，并在需要时执行身份验证。

the technical details of the decentralization platform are still a topic of investigation among researchers and practitioners. For example, W3C has Social Web Working Group actively investigating decentralization standards[2].
权力下放平台的技术细节仍然是研究人员和从业人员调查的课题。例如, w3c 让社会网络工作组积极调查权力下放标准 [2].

Our current activity in this space centers around a platform that we call *Solid*[2] (for *Social Linked Data*) [7]. The Solid platform supports decentralized social Web applications, relying as much as possible on W3C standards and Semantic Web technologies to realize the architecture shown in Figure 1. The platform specifies all the protocols required in the figure, such as authentication, application-to-server and server-to-server communications. This demonstration showcases the Solid platform, focusing on the experience of the end user and the application developer. In particular, we demonstrate through a set of applications and servers supporting these applications that Solid enables a high degree of interoperability between applications, easy sharing of data and the social graph between applications, and portability of data between servers.

我们目前在这个空间的活动围绕着一个平台，我们称之为 solid (用于社会链接数据) [7]。solid 平台支持分散的社交 web 应用程序，尽可能依靠 w3c 标准和语义 web 技术来实现图 1 所示的体系结构。该平台指定图中所需的所有协议，如身份验证、应用程序到服务器和服务器到服务器的通信。本演示展示了 solid 平台，重点介绍了最终用户和应用程序开发人员的体验。特别是，我们通过一组支持这些应用程序的应用程序和服务器来证明，solid 支持应用程序之间的高度互操作性、应用程序之间轻松共享数据和社交图，以及服务器之间数据交换的便捷性。

These features are demonstrated through standard applications such as maintaining the list of contacts of a user, and more novel applications enabled by Solid such as collaborative authoring of scholarly articles. We are developing the Solid platform as part of our Crosscloud project[3], which aims to address the research challenges

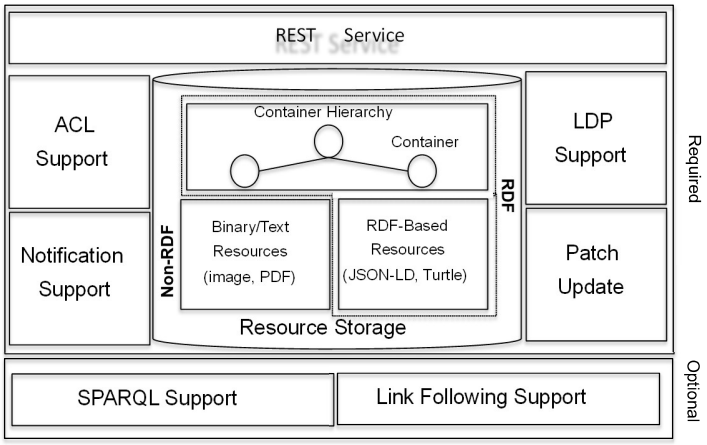

Figure 2: Overview of a pod server. A pod stores RDF and non-RDF resources. The server support LDP, patching resources, access control, live updates, and optionally SPARQL.

pod 服务器概述。pod 存储 rdf 和非 rdf 资源。服务器支持 ldp、修补资源、访问控制、实时更新和 sparql。

*Sparql 是 RDF 查询语言

http://www.chinaw3c.org/REC-sparql11-overview-20130321-cn.html

*REST 是一 种万维网软件架构风格

[2] https://github.com/solid/solid

[3] http://crosscloud.org

http://www.drdobbs.com/web-development/restful-web-services-a-tutorial/240169069

related to building a decentralized social Web.
这些功能通过标准应用程序 (如维护用户的联系人列表) 和由 solid 启用的更新颖的应用程序 (如协作创作学术文章) 进行演示。我们正在开发 solid 平台，作为我们 Crosscloud 项目的一部分，该项目旨在应对与构建分散的社交网络有关的研究挑战。
A good choice of protocols on the wire is essential to Crosscloud, and Solid provides such protocols. In addition to designing protocols, the Crosscloud project must address several research questions. For example, what are the data models and design patterns that applications should use to store data? How can we ensure that applications agree on a vocabulary for the concepts that they use, and how to integrate data from different applications when needed? What is the best way to support notifications from application to application? An interesting topic of investigation is the extent to which Web traversal and complex data retrieval can be offloaded from client to server. Supporting application developers is also important to build momentum around the Crosscloud ecosystem. In addition, suitable models for security and privacy are essential for the social Web, and decentralization makes the questions around these models more complicated [1, 9]. This demonstration of Solid provides a framework for appreciating these questions.
在线上选择好协议对于交叉云来说是必不可少的，solid 提供了这样的协议。除了设计协议外，crosscloud 项目还必须解决几个研究问题。例如，应用程序应使用哪些数据模型和设计模式来存储数据？我们如何确保应用程序就其使用的概念的词汇达成一致，以及如何在需要时集成来自不同应用程序的数据？支持从应用程序到应用程序的通知的最佳方法是什么？一个有趣的研究主题是 web 遍历和复杂数据检索在多大程度上可以从客户端卸载到服务器。支持应用程序开发人员对于围绕交叉云生态系统建立势头也很重要。此外，适当的安全和隐私模型对于社交网络至关重要，而权力下放使围绕这些模型的问题变得更加复杂 [1, 9]。而 Solid 为解决这些问题提供了思路。

The rest of this document is organized as follows. We present a brief overview of the Solid platform in Section 2. Section 3 then discusses application development in Solid and presents some of the Solid applications that will be used in the demonstration. In Section 4, we describe some possible demonstration scenarios. Section 5 concludes.

本文档的其余部分按如下方式进行组织。我们在第 2 节中简要介绍了 solid 平台。第 3 节随后讨论了 solid 中的应用程序开发，并介绍了将在演示中使用的一些 solid 应用程序。在第 4 节中，我们将介绍一些可能的演示方案。第 5 节结束。

2. OVERVIEW OF SOLID /Solid 的概念

In the Solid platform, each user stores their data in a Web-accessible *personal on-line datastore* (or *pod*). Applications run as client-side Web applications in a browser or as mobile applications. These applications use an authentication protocol to discover the user's identity and profile data, as well as relevant links that point to the user's pod, which contains application data. Solid supports decentralized authentication and access control, and it also supports standardized data access mechanisms. We describe these two aspects next.
在 solid 平台中，每个用户都将其数据存储在 web 可访问的个人在线数据存储 (或 pod) 中。应用程序在浏览器中作为客户端 web 应用程序运行，或作为移动应用程序运行。这些应用程序使用身份验证协议来发现用户的标识和配置文件数据，以及指向用户的 pod (其中包含应用程序数据) 的相关链接。solid 支持分散身份验证和访问控制，还支持标准化的数据访问机制。接下来我们将描述这两个方面。

Decentralized authentication, a global ID space, and global single sign-on are a critical part of the Solid ecosystem. Solid uses WebID [8] to provide these features, although other solutions exist and can potentially interoperate with Solid. In Solid, a user has to register with an identity provider, and this identity provider stores the user's WebID profile document associated with a cryptographic key. In most cases, a pod provider would also operate as an identity provider, offering WebID "accounts" to its users.
分散身份验证、全局 id 空间和全局单点登录是 solid 生态系统的重要组成部分。solid 使用 webid [8] 来提供这些功能，尽管存在其他解决方案，并且可能与 solid 进行互操作。在 solid 中，用户必须向标识提供程序注册，并且此标识提供程序存储与加密密钥关联的用户的 weid 配置文件文档。在大多数情况下，pod 提供商也将作为标识提供商运行，向其用户提供 webid "帐户"。

Table 1: Pod servers. databox.me, meccano.io, and rww.io act as public pod servers as well as identity providers, allowing users to create WebIDs.
表 1: pod 服务器。databox. me、meccano. io 和 rwww. io 充当公共 pod 服务器以及标识提供程序，允许用户创建 web id。

Name	Platform	Running Service
gold	golang	*https://databox.me/*
meccano	Java+Jena	*https://meccano.io/*
ldphp	PHP	*https://rww.io/*
ldnode	node.js	not public

Application data in Solid is stored in users' pods and pods are stored on *pod servers*. Data is managed in a RESTful way, as defined by the Linked Data Platform (LDP) recommendation [6]. LDP enables applications to manage data items within hierarchical containers (which can also be called collections or directories). Each data item and container has a URI, and LDP defines the protocol for manipulating the data items and containers through HTTP requests on their URIs; for example, POST/PUT to create, PUT/PATCH to update, and GET to retrieve. Items can be found through their URIs, or by following links from other items. Solid distinguishes between structured data, which is represented in Solid using RDF [5], and unstructured data that can be of any type, e.g., videos, images, Web pages. This allows structured data to be parsed and serialized in various formats such as Turtle or JSON-LD.
solid 中的应用程序数据存储在用户的 pod 中，pod 则存储在 pod 服务器上。根据链接数据平台 (ldp) 建议 [6] 的定

义, 以 restful 方式管理数据。ldp 使应用程序能够管理分层容器中的数据项 (也可以称为集合或目录)。每个数据项和容器都有一个 uri, ldp 定义了通过其 uri 上的 http 请求操作数据项和容器的协议;例如, 要创建的 psts/put、要更新的 put/patch 和要检索的 get。项目可以通过其 uri 找到, 也可以通过以下其他项目的链接找到。solid 区分了使用 rdf [5] 在 solid 中表示的结构化数据和可用于任何类型的非结构化数据 (如视频、图像、网页)。这允许以各种格式 (如 Turtle 或 JSON-LD) 对结构化数据进行分析和序列化。

Additional to LDP support, pod servers may offer optional SPARQL support. Servers that support SPARQL allow applications to express complex data retrieval operations, including operations that require server-to-server communication via link-following SPARQL. This simplifies Solid application development, since it enables a developer to delegate complex, multi-pod data retrieval operations to the server.

除了 ldp 支持外, pod 服务器还可以提供可选的 sparql 支持。支持 sparql 的服务器允许应用程序表示复杂的数据检索操作, 包括需要通过链路跟踪 sparql 进行服务器到服务器通信的操作。这简化了 solid 应用程序开发, 因为它使开发人员能够将复杂的多 pod 数据检索操作委托给服务器。

Pod servers in Solid are application-agnostic, so that new applications can be developed without having to modify the servers. For example, even though LDP 1.0 contains nothing specific to "social", many of the W3C Social Web Working Group user stories[4] can be implemented in Solid, using only LDP and application logic, with no changes to the server.

The requirements of a pod server are illustrated in Figure 2. A pod server needs to store RDF and non-RDF resources, and it needs to support basic LDP access to these resources, patching resources, access control lists (ACLs), live updates, and optionally SPARQL. There are several ways in which the underlying storage for RDF data can be implemented in a pod server, e.g., using the file system, a key-value store, a relational database system, or a graph database system (i.e., a triple/quad store).

solid 中的 pod 服务器与应用程序无关, 因此无需修改服务器即可开发新的应用程序。例如, 即使 ldp 1.0 不包含特定于 "社交" 的内容, 但许多 w3c 社交 web 工作组用户情景可以在 solid 中实现, 仅使用 ldp 和应用程序逻辑, 而不会对服务器进行任何更改。

pod 服务器的要求如图 2 所示。pod 服务器需要存储 rdf 和非 rdf 资源, 并且需要支持对这些资源的基本 ldp 访问、修补资源、访问控制列表 (acl)、实时更新和 (可选 sparql)。有几种方法可以在 pod 服务器中实现 rdf 数据的基础存储, 例如, 使用文件系统、键值存储、关系数据库系统或图形数据库系统 (即 triple/quad 存储)。

We have implemented several prototype servers, listed in Table 1. Our **ldphp**[5], **gold**[6], and **ldnode**[7] servers store all their data in the file system. In this case, both RDF and non-RDF resources are stored as files, including the RDF resources representing ACLs and the metadata documents corresponding to non-RDF resources (all of which are defined by LDP). Our **meccano** server stores RDF data in a graph database system (currently we use Jena[8]), and it stores non-RDF data in the file system. **Meccano** implements all Solid operations via SPARQL queries, and it also implements complex data retrieval using link-following SPARQL. Using an RDF database simplifies querying large data sets, efficient data retrieval (i.e., subsets of graphs), as well as patch operations.

我们已经实现了几个原型服务器, 如表 1 所示。我们的 **ldphp**、**gold** 和 **ldnode** 服务器将其所有数据存储在文件系统中。在这种情况下, rdf 和非 rdf 资源都存储为文件, 包括表示 acl 的 rdf 资源和与非 rdf 资源相对应的元数据文档 (所有这些都由 ldp 定义)。我们的 **meccano** 服务器将 rdf 数据存储在图形数据库系统中 (目前我们使用 jena), 并将非 rdf 数据存储在文件系统中。**meccano** 通过 sparql 查询实现所有 solid 操作, 它还使用链接跟踪 sparql 实现复杂的数据检索。使用 rdf 数据库简化了大型数据集的查询、高效的数据检索 (即图形子集) 以及修补程序操作。

3. APPLICATION DEVELOPMENT/（Solid）技术的发展

In this section, we discuss application development in Solid and give examples of the Solid applications that we have implemented. The intention is to demonstrate the flexibility of the underlying architecture of Solid and the benefits of decentralization.

在本节中, 我们将讨论 solid 中的应用程序开发, 并举例说明我们已经实现的 solid 应用程序。目的是展示 solid 的基本结构的灵活性和权力下放的好处。

Solid application development is supported by a set of libraries and components. For example, all the applications that we have developed use the rdflib.js library (the core library from Tabulator [2]) to handle RDF resources. Another library is solid.js[9], which simplifies the development of Solid applications by abstracting some of the more complex operations. We have also provided modules for authentication[10]and signup[11] that are designed for reuse as Web Components [4]. We are continuously growing the set of libraries and components in the Solid ecosystem, and we expect this to significantly accelerate the adoption of Solid.

Solid 技术的发展依赖于可靠的组库和组件的支持。

[4] http://www.w3.org/wiki/Socialwg/Social API/User stories
[5] https://github.com/linkeddata/ldphp
[6] https://github.com/linkeddata/gold
[7] https://github.com/linkeddata/ldnode
[8] http://jena.apache.org
[9] https://github.com/solid/solid.js
[10] https://github.com/linkeddata/webid-login
[11] https://github.com/linkeddata/webid-signup

例如，我们开发的所有应用程序都使用 rdflib. js 库（来自 tabulator [2] 的核心库）来处理 rdf 资源。另一个库是 solid. js，它通过抽象一些更复杂的操作来简化 solid 应用程序的开发。我们还提供了用于身份验证和注册的模块，这些模块设计用于作为 web 组件重用 [4]。我们不断地扩展 solid 生态系统中的库和组件集，我们预计这将显著加快 solid 的采用。

We have developed several Solid applications for common day-to-day tasks, listed in Table 2. Some of these applications use the AngularJS and jQuery frameworks, which provides a proven set of features in terms of application interactivity. For this demonstration, all applications were developed as responsive Web applications and tested in Firefox and Chrome. Screenshots of the **contacts** and **dokieli** applications are shown in Figures 3 and 4. We describe these two applications next.

我们已经为常见的日常任务开发了几种 solid 应用程序，如表 2 所示。其中一些应用程序使用 angularjs 和 jquery 框架，它在应用程序交互性方面提供了一组经过验证的功能。在此演示中，所有应用程序都是作为响应式 web 应用程序开发的，并在 firefox 和 chrome 中进行了测试。contacts 和 dokieli 应用程序的屏幕截图如图 3 和图 4 所示。接下来我们将介绍这两个应用程序。

The **contacts** application manages a list of contacts stored on a user's pod. In Solid, a user's social graph is made up of the contacts stored on their pod, the contacts of these contacts, and so on, where each user is identified by a WebID. Thus, **contacts** can be viewed as an interface for managing a user's distributed social graph. The **contacts** application maintains a set of vCards for a user's contacts using the vCard ontology[12]. Each vCard is a resource with a unique URI, and can contain the WebID of the user that it represents in addition to other fields such as name and e-mail. A user may mark a vCard as public or allow a vCard to be accessed by an individual or a group of people (identified by WebIDs).

* vCards 是一个/一组电子名片

联系人应用程序管理存储在用户 pod 上的联系人列表。在 solid 中，用户的社交图由存储在其 pod 上的联系人、这些联系人的联系人等组成，其中每个用户都由 webid 标识。因此，联系人可以被看作是管理用户分布式社交图的界面。联系人应用程序使用 vCards 本体为用户的联系人维护一组 vCards。每个 vcard 都是具有唯一 uri 的资源，除了名称和电子邮件等其他字段外，还可以包含它所代表的用户的 web id。用户可以将 vcard 标记为公共，也可以允许个人或一组人访问 vcard（由 webid 标识）。

One of the interesting social features in our **contacts** application, enabled by Solid, is the ability to search in the "contacts of your contacts"using link-following SPARQL. A user can search in their pod for a vCard matching search criteria such as name, email, or address. In addition, the **contacts** application can use a link-following SPARQL query to search for a contact in the public contacts on pods that can be reached from WebIDs in the user's vCards (via link following). The user gets a list of vCards matching the search criteria, and the URI of each answer vCard indicates the source of this card. This search capability provides an example of the innovative social features supported by a decentralized social platform such as Solid.

在我们的联系人应用程序中，由 solid 启用的一个有趣的社交功能是能够使用链接跟踪 sparql 在 "您的联系人的联系人" 中搜索。用户可以在其 pod 中搜索符合姓名、电子邮件或地址等搜索条件的 vcard。此外，联系人应用程序可以使用链接跟踪 sparql 查询在 pod 上的公共联系人中搜索可从用户 vCards 中的 webid 访问的联系人（通过以下链接）。用户获取与搜索条件匹配的 vCards 列表，每个应答 vCards 的 uri 指示此名片的来源。这种搜索功能提供了一个由分散的社交平台（如 solid）支持的创新社交功能的示例。

dokieli is a decentralized article authoring, annotation, and social notifications application[13]. While it is a general purpose tool to write and manage articles, it is compliant

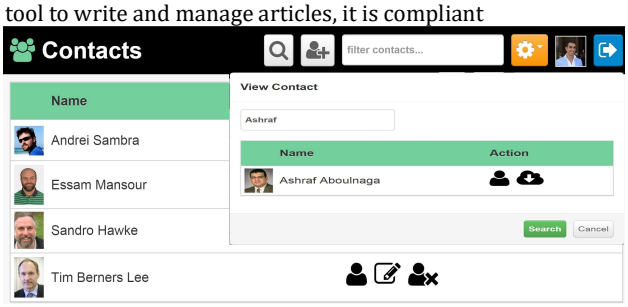

Figure 3: The contacts application maintains a set of vCards using LDP and uses link-following SPARQL to search for vCards in the user's pod and pods reached from WebIDs in these vCards.

图 3: 联系人应用程序使用 ldp 维护一组 vCards，并使用链接跟踪 sparql 在用户的 pod 和从这些 vCards 中的 webid 到达的 vCards 中搜索 vCards。

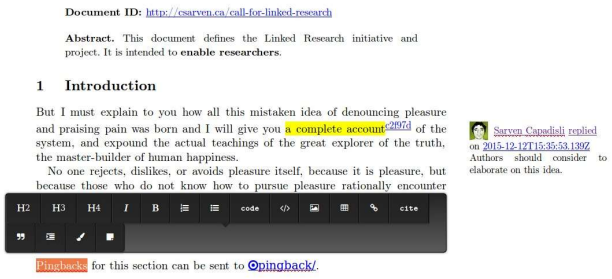

Figure 4: dokieli is a general purpose authoring and interaction tool among different pods.

图 4:dokieli 是一种通用的创作和交互工具。

with the Linked Research initiative[14], and provides social features and interactions for scholarly communication. Articles can also be semantically annotated, anywhere from a fragment in a sentence to descriptions, e.g., hypothesis, workflows, evaluations, and have their own URIs to foster discovery and reuse. All URIs can be dereferenced and have their content representations in HTML and RDF.

[12] http://www.w3.org/2006/vCard
[13] https://github.com/linkeddata/dokieli
[14] https://github.com/csarven/linked-research

dokieli 是一个分散的文章创作、注释和社交通知应用程序。 在 solid 应用中使用 sparql 也相对简单。开发人员可以

Table 2: Solid applications. An asterisk (*) indicates a third-party application, not developed by us.
表 2: solid 应用。星号 (*) 表示我们未开发的第三方应用程序。

Name	Function/功能	Usable At/网址
contacts	Manage a list of contacts/管理联系人列表	*http://mzereba.github.io/contacts*
contacts	Manage a list of contacts/管理联系人列表	*http://linkeddata.github.io/contacts*
calendar	Event manager/管理日程安排	*http://mzereba.github.io/calendar*
dokieli	Decentralized authoring, annotation, and social notifications 分散的创作、注释和社交通知	*https://dokie.li*
pad	Shared collaborative editing/共享协同编辑	*https://github.com/timbl/pad*
profile-editor	View and update a user's profile/查看和更新用户的配置文件	*http://linkeddata.github.io/profile-editor*
warp	Solid file browser/Solid 文件浏览器	*http://linkeddata.github.io/warp*
cimba	Microblogging (cf. Twitter)/博客（就像 Twitter）	*http://cimba.co*
zagel	Instant messaging/group chat/即时消息/群聊	*https://solid.github.io/solid-zagel*
***webid.im**	Instant messaging/chat/即时消息/聊天	*http://webid.im*
***shamblokus**	Strategy game (cf. Blokus)/策略游戏(就像 Blokus)	*http://deiu.github.io/Shamblokus*

虽然它是撰写和管理文章的通用工具, 但它符合链接研究倡议, 并为学术交流提供社会特征和互动。文章也可以在语义上进行注释——从句子中的片段到描述 (例如, 假设、工作流、评估), 并有自己的 uri 来促进发现和重用。可以取消引用所有 uri, 并在 html 和 rdf 中使用其内容表示形式。

dokieli employs the WebIDs and pods of authors, reviewers, and commenters for instance to store the information created by these participants and can assign them different access controls. For example, annotations and social notifications like replies, peer-reviews, liking, resharing, can reside in the contributor's pod.

dokieli 使用作者、审阅者和评论员的 weid 和 pod 来存储这些参与者创建的信息, 并可以为他们分配不同的访问控制。例如, 批注和社交通知 (如回复、同行审阅、赞、重新共享) 可以驻留在贡献者的 pod 中。

Our experience with Solid application development confirms that Solid provides a feature-rich platform that supports portability and interoperability. Applications can work with multiple pod server implementations, and it is easy to change the applications that use data without changing the data (e.g., by forking and adding features).

我们在 solid 应用程序开发方面的经验证实, solid 提供了一个功能丰富的平台, 支持可移植性和互操作性。应用程序可以使用多个 pod 服务器实现, 并且可以轻松地更改使用数据的应用程序, 而无需更改数据 (例如, 通过分叉和添加功能)。

Implementing social features in Solid applications is simple and requires a thin layer of reusable code. Applications and pods interact with each other by taking advantage of the HTTP/1.1 methods, which do the heavy-lifting.

在 solid 应用程序中实现社交功能非常简单, 需要一层薄薄的可重用代码。应用程序和 pod 通过利用 httpx1.1 方法相互交互, 这些方法起到了很大的作用。

Using SPARQL in Solid applications is also relatively simple. Developers can write SPARQL queries to express data access features such as search, filtering, or fetching top results. Developers can also write link-following SPARQL queries to follow links between pods.

编写 sparql 查询来表示数据访问功能, 如搜索、筛选或获取顶级结果。开发人员还可以编写链接跟踪 sparql 查询, 以跟踪 pod 之间的链接。

4. DEMONSTRATION PLAN/应用实例展示

Demonstration participants will be able to interact with all the applications shown in Table 2, and will be able to store data on two different pod servers: databox.me and meccano.io. This section provides a specific demonstration scenario using these servers.

使用者将能够与表 2 中显示的所有应用程序进行交互, 并能够将数据存储在两个不同的 pod 服务器上: databox.me and meccano.io。本节提供了使用这些服务器的特定应用方案。

The demonstration scenario involve two users, Alice and Bob, using different pod servers. Alice will use the **gold** server at databox.me, and Bob will use the **meccano** server at meccano.io. We will show that although these are two totally different servers, both users can use the same applications to access and maintain their data. This can be shown using any of the applications in Table 2. An application will be able to create, modify, delete, and retrieve resources in the user's pod. Demonstration participants can view these resources using the **warp** file browser, and can also see the client-server interaction involved.

演示方案涉及两个用户, alice 和 bob, 使用不同的 pod 服务器。alice 将使用 databox.me 中的 gold 服务器, bob 将使用 meccano.io 的 meccano 服务器。我们将展示, 虽然这是两个完全不同的服务器, 但这两个用户可以使用相同的应用程序来访问和维护他们的数据。这可以使用表 2 中的任何应用程序来显示。应用程序将能够在用户的 pod 中创建、修改、删除和检索资源。演示参与者可以使用 warp 文件浏览器查看这些资源, 还可以查看所涉及的客户端-服务器交互。

Besides the basic Solid functionality, the demonstration will turn to interoperability and access control. Interoperability will be demonstrated through the **dokieli** application enabling social interactions among users, and through applications using link-following queries. For example, we will demonstrate how Alice

can use link-following queries in the **contacts** application to search in the public contacts of Bob. In addition to demonstrating interoperability, these examples will also demonstrate access control. They will also demonstrate other featured of Solid such as delegation[15] in order to allow a pod to speak on behalf of its owner. As before, demonstration participants can view the resources being created, observe client-server interactions, and also server-to-server interactions.

除了基本的 solid 功能外, 演示还将转向互操作性和访问控制。互操作性将通过 dokieli 应用程序进行演示, 该应用程序可在用户之间实现社会互动, 并通过使用链接跟踪查询的应用程序进行演示。例如, 我们将演示 alice 如何在联系人应用程序中使用链接跟踪查询在 bob 的公共联系人中进行搜索。除了演示互操作性外, 这些示例还将演示访问控制。他们还将展示 solid 的其他特征, 如授权, 以便允许一个 pod 代表其所有者发言。与以前一样, 演示参与者可以查看正在创建的资源, 观察客户端-服务器交互, 以及服务器到服务器的交互。

Another form of interoperability is having multiple applications use the same data. We will show that a user can use two different **contacts** applications to manage the same set of contacts. We will also demonstrate the portability provided by Solid by showing how Alice can easily migrate her pod from databox.me to meccano.io. After this migration, Alice needs to change her WebID profile to point to the new storage, and her applications will be redirected to the new pod.

另一种形式的互操作性是让多个应用程序使用相同的数据。我们将显示用户可以使用两个不同的联系人应用程序来管理同一组联系人。我们还将展示如何将 alice 的 pod 从 databox.me 迁移到 meccano. io, 从而展示 solid 提供的可移植性。迁移后, alice 需要更改其 web id 配置文件以指向新存储, 她的应用程序将被重定向到新的 pod。

5. CONCLUSION/结论

Re-decentralizing the social Web is an important topic and an active area of research. The Solid platform is a concrete instance of a decentralized platform for social Web applications, providing decentralized authentication, decentralized data management, developer support in the form of libraries and web components, and a suite of running servers and example applications. This demonstration will show how the Solid platform can enable social applications while allowing each user to retain control of their pod. Demonstration participants will experience Solid from a user and application developer perspective. They will gain insights into the interoperability and portability features provided by Solid, the rich social features that it can enable, and the client and server machinery behind these features. A concrete appreciation of such a platform is very valuable in the ongoing discussion on re-decentralization.

社会网络的重新分散是一个重要的课题, 也是一个活跃的研究领域。solid 平台是社交 web 应用程序分散平台的具体实例, 它提供分散身份验证、分散的数据管理、以库和 Web 组件形式提供的开发人员支持, 以及一套运行服务器和示例应用程序。本演示将展示 solid 平台如何启用社交应用程序, 同时允许每个用户保留对其 pod 的控制。演示参与者将从用户和应用程序开发人员的角度体验 solid。他们将深入了解 solid 提供的互操作性和可移植性功能、它可以启用的丰富社交功能以及这些功能背后的客户端和服务器机制。对这一平台的具体认识在正在进行的关于重新权力下放的讨论中非常有价值。

[15] https://github.com/solid/solid-spec#webid-delegatedrequests

6. REFERENCES/参考文献

[1] L. M. Aiello and G. Ruffo. LotusNet: Tunable privacy for distributed online social network services. *Computer Communications*, 35(1), 2012.

[2] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic Web. In *Proc. Int. Semantic Web User Interaction*, 2006.

[3] B. Dodson, I. Vo, T. J. Purtell, A. Cannon, and M. S. Lam. Musubi: Disintermediated interctive social feeds for mobile devices. In *Proc. World Wide Web Conf. (WWW)*, pages 211–220, 2012.

[4] D. Glazkov and H. Ito. Introduction to Web components. *W3C Working Group Note*, 14, 2014.

[5] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. 2006.

[6] A. Malhotra, J. Arwe, and S. Speicher. Linked Data Platform Specification. W3C Recommendation, 2015. http://www.w3.org/TR/ldp/.

[7] A. Sambra, A. Guy, S. Capadisli, and N. Greco. Building decentralized applications for the social Web. Tutorial at the World Wide Web Conf. (WWW), 2016.

[8] A. V. Sambra, H. Story, and T. Berners-Lee. WebID Specification. 2014. http://www.w3.org/2005/Incubator/webid/spec/identity/.

[9] S. Schulz and T. Strufe. d2 deleting Diaspora: Practical attacks for profile discovery and deletion. In *Proc. IEEE Int. Conf. on Communications (ICC)*, 2013.

[10] M. Van Kleek, D. A. Smith, N. R. Shadbolt, and mc schraefel. A decentralized architecture for consolidating personal information ecosystems: The WebBox. In *Proc. Workshop on Personal Information Management (PIM)*, 2012.