



面向开发人员的区块链

S



灵宗, 博士
ibm almaden 研究中心
美国加利福尼亚州圣何塞

目标

在此会话你将了解超级分类帐编辑器和其他可用于与区块链网络交互的工具。

您还将了解典型区块链解决方案的组件, 如分类帐、智能合同、对等网络和钱包。

最后, 您将了解区块链开发人员和管理员的主要职责。

目标

完成后, 您应该了解:

- 使用超级分类帐编辑器的结构和好处, 它是一个开源项目, 也是业务网络的一套高级抽象
- 如何使用超级分类分类编辑器建模和构建一个简单的业务网络
- 超分类分类结构的组件和结构, 它是一个区块链框架, 使用容器来承载称为链码构成了系统的应用程序逻辑
- 区块链解决方案组件, 如钱包、分类帐、参与者、共识、安全性和智能合同
- 帮助构建、建模、运行和维护区块链业务网络的人员的主要考虑因素和责任:
开发人员、管理员
- 达成共识的方法
- 可用于与网络交互的工具和应用程序

演示和我Abs

- **演示:** 如何使用本地安装的超级分类帐编辑器对业务网络建模、将其部署到本地结构、将其公开为 rest api 以及如何创建可与网络交互的简单 web 应用程序。演示还演示了如何查找文档、如何安装编辑器及其命令行界面。
- **实验室1:** 创建超级分类帐编辑器解决方案
- **实验室2:** 编写您的第一个区块链应用程序
- **实验室3:** 构建自己的网络

本节的目标

完成此部分后, 您将了解:

- 什么是超级分类帐编辑器及其工作原理
- 如何使用 "作曲家" 对简单的区块链网络进行建模

观看视频

什么是超级分类帐编辑器--超级分类帐编辑器。

实验室 1: 创建超级分类帐编辑器解决方案

在本练习中, 您将使用超级分类帐编辑器从头开始建模区块链解决方案, 并创建一个与区块链网络交互的基于角度2的示例应用程序。

编辑器可帮助您根据自己的用例对网络进行建模和测试。

您必须安装超级分类帐编辑器、docker 和其他软件。有关详细信息, 请参阅实验室。

<https://hyperledger.github.io/composer/latest/tutorials/developer-tutorial>

用于创建超级分类帐编辑器解决方案的开发人员教程

- 本教程将引导您从零开始构建超级分类帐管理程序区块链解决方案。
- 在几个小时的时间里, 您将能够从颠覆性区块链创新的想法, 到针对真正的超分类分类分类器结构区块链网络执行事务, 并生成一个示例角度2应用程序, 该应用程序与区块链网络。
- 本教程概述了可应用于您自己的用例的技术和资源。
- **注意:** 本教程是针对最新的超级分类帐编辑器构建的**ubuntu linux**使用超分类分类帐结构 v1.1 运行, 其中下面引用了该版本, 并针对 mac 环境进行了测试。

用于创建超级分类帐编辑器解决方案的开发人员教程。医生

区块链解决方案的组件

现在, 让我们更深入地了解区块链解决方案的结构、参与者、角色和组件。当您为您的企业创建区块链网络解决方案时, 您可能需要应用程序开发人员、架构师、网络运营商和监管机构。您的解决方案可能还需要传统的处理平台和数据源。

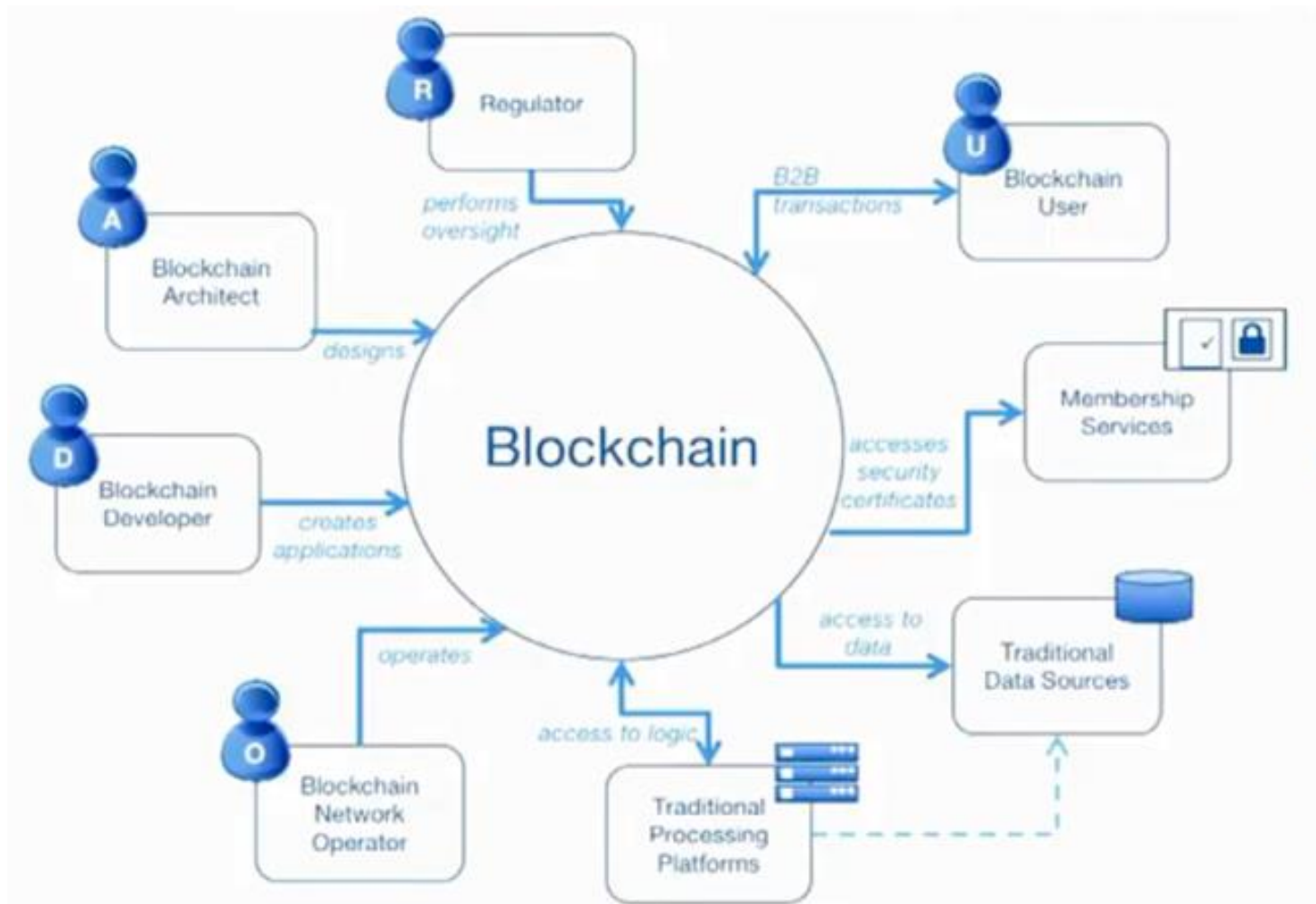
目标

完成此部分后, 您将了解:

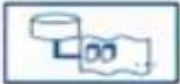


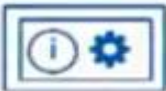
- 区块链解决方案中的参与者及其角色
- 区块链解决方案中的组件, 如分类帐、智能合同、对等网络和钱包
- 三个主要角色的具体注意事项: 应用程序开发人员、架构师和网络操作员
- 如何创建第一个区块链应用程序

参与者和组成部分概述

谁是区块链网络的参与者

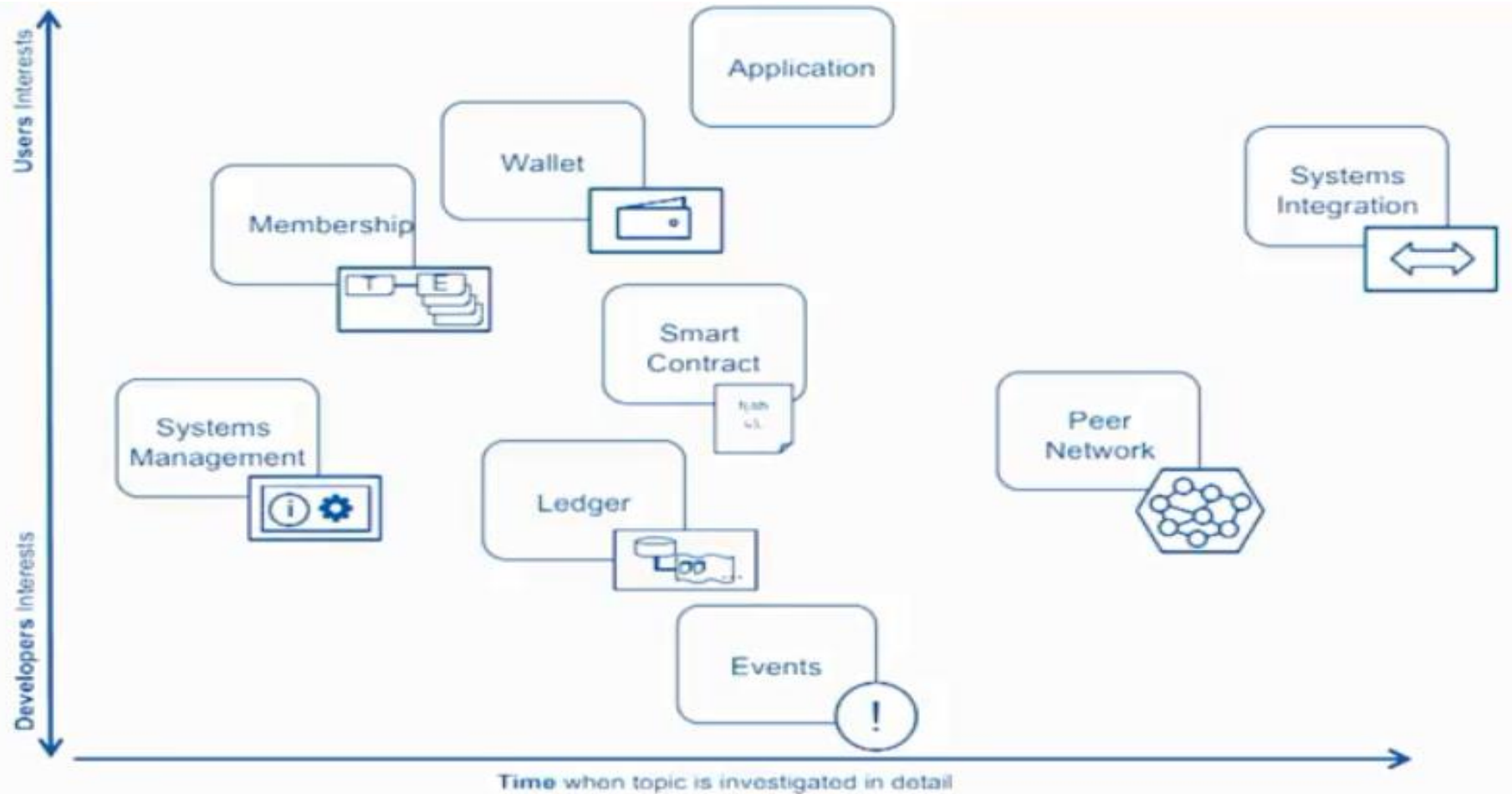


区块链中的组件

Ledger		A ledger is a channel's chain and current state data which is maintained by each peer on the channel.
Smart Contract		Software running on a ledger, to encode assets and the transaction instructions (business logic) for modifying the assets.
Peer Network		A broader term overarching the entire transactional flow, which serves to generate an agreement on the order and to confirm the correctness of the set of transactions constituting a block.
Membership		Membership Services authenticates, authorizes, and manages identities on a permissioned blockchain network.
Events		Creates notifications of significant operations on the Blockchain (e.g. a new block), as well as notifications related to smart contracts. Does not include event distribution.
Systems Management		Provides the ability to create, change and monitor Blockchain components
Wallet		Securely manages a user's security credentials
Systems Integration		Responsible for integrating Blockchain bi-directionally with external systems. Not part of Blockchain, but used with it.

智能合约定义了管理整个区块链网络的规则, 就像管理纸牌或棋盘游戏的规则一样。

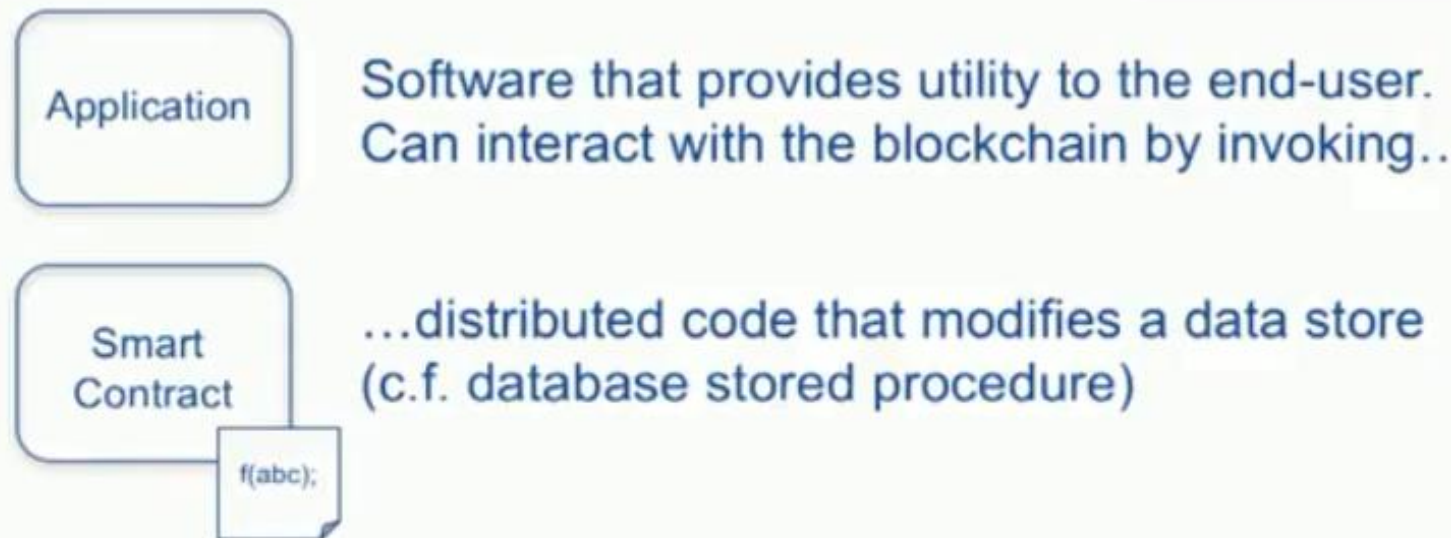
区块链中的组件



证书 (注册证书) 标识网络中的个人。tcerts (事务证书) 捕获事务的事件。网络上的任何人都可以查找埃克特或特克特。

区块链开发人员的注意事项

Blockchain developers' primary interests are:



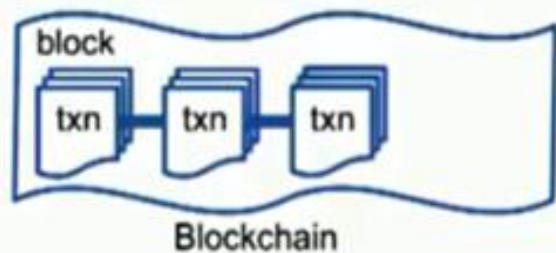
They should NOT have to care about operational concerns, such as:



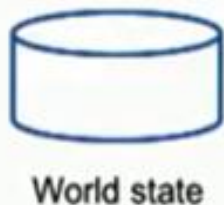
在超分类帐结构中, 开发人员需要真正了解应用程序和智能协定。在这里, 我们来看看一个例子, 区块链和世界状态在分类帐, 如何集成与智能合同和应用程序。开发人员还将理解在集成方法 (如事件、转换和调用直接从智能合同中调用) 中构建的内容, 最后一个选项可能会出现不确定结果的问题, 而这一问题将达不到共识。

区块链开发人员的注意事项

分类帐通常由两个数据结构组成



- A linked list of blocks
- Each block describes a set of transactions (e.g. the inputs to a smart contract invocation)
- Immutable – blocks cannot be tampered

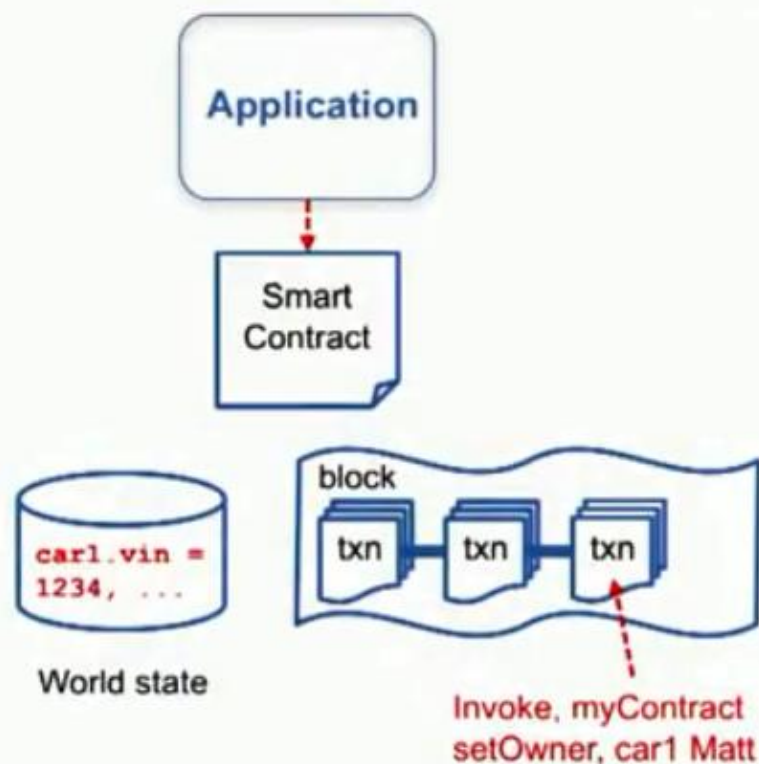


- An ordinary database (e.g. key/value store)
- Stores the combined outputs of all transactions
- Not usually immutable

区块链开发人员的注意事项

分类帐示例

"Change the owner of car1 to Matt"



Transaction input - sent from application

```
invoke(myContract, setOwner,  
       car1, Matt)
```

Smart contract implementation

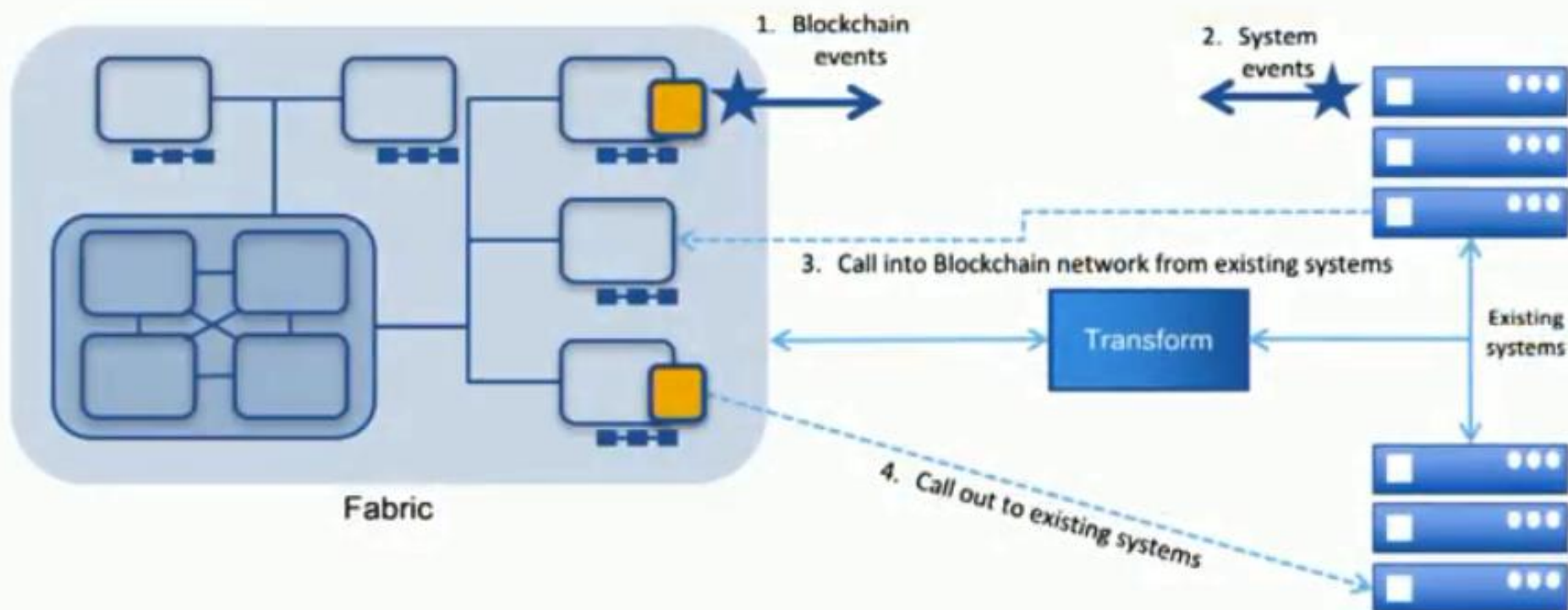
```
setOwner(Car, newOwner) {  
    set Car.owner = newOwner  
}
```

World state - new contents

```
car1.vin = 1234  
car1.owner = Matt  
car1.make = Audi  
...
```

区块链开发人员的注意事项

与现有系统集成



实验室 2: 编写您的第一个区块链应用程序

在本练习中, 您将使用命令行工具在 javascript 中构建一个简单的应用程序, 该应用程序查询和更新预先填充的区块链网络分类帐。您的应用程序将使用 api 来调用智能协定或链代码。您将使用超分类帐结构节点 sdk 访问 api。

您将在下一个实验室中更详细地使用网络。

"编写您的第一个应用程序" 模块的通知: 版权超级分类帐2017。修订 b33fb974。
根据知识共享归因4.0 国际许可获得许可

<https://creativecommons.org/licenses/by/4.0/legalcode>. 模块可在
http://hyperledger-fabric.readthedocs.io/en/latest/write_first_app.html.

写你的第一个应用程序。

实验室 2: 编写您的第一个区块链应用程序

我们将经历三个主要步骤:

1. 设置开发环境。

我们的应用程序需要一个网络进行交互, 因此我们将下载一个被剥离的应用程序, 以便仅下载到注册/注册、查询和更新所需的组件:

2. 学习我们的应用程序将使用的示例智能合同的参数。

我们的智能合同包含各种功能, 使我们能够以不同的方式与分类帐进行交互。我们将进入并检查该智能合同, 以了解我们的应用程序将使用的功能。

3. 开发应用程序, 以便能够查询和更新分类帐上的资产。

我们将进入应用代码本身 (我们的应用程序已在 javascript 中编写), 并手动操作变量以运行不同类型的查询和更新。

实验室 2: 编写您的第一个区块链应用程序

完成本教程后, 您应该对如何将应用程序与智能合同一起编程以与结构网络上的分类帐 (即对等) 进行交互有一个基本的了解。

设置开发环境

如果你已经跑过[构建您的第一个网络](#), 您应该有您的开发环境设置, 并将已下载[面料样品](#)以及随附的文物。要运行本教程, 您现在需要执行的操作是拆除了您拥有的任何现有网络, 您可以通过颁发以下命令来执行此操作: `./比芬.嘘下来`

如果您没有网络 and 应用程序的开发环境以及随附的工件, 请访问[先决条件](#)页, 并确保您的计算机上安装了必要的依赖项。

实验室 2: 编写您的第一个区块链应用程序

接下来, 如果您还没有这样做, 请访问[安装示例](#)、[二进制文件和 docker 映像](#)页, 然后按照提供的说明进行操作。克隆结构图库并下载最新的稳定结构映像和可用实用程序后, 返回本教程。此时, 应安装所有内容。导航到您的面料样本存储库中的 fabcar 子目录, 并查看里面的内容:

```
cd 面料-样品/法卡尔&&l
```

您应该看到以下内容:

```
注册管理员.js 调用.js 套餐.杰森查询.js 注册用户.js startfabric.Sh
```

实验室 2: 编写您的第一个区块链应用程序

在开始之前, 我们还需要做一点家务。运行以下命令以杀死任何陈旧或活动的容器:

码头工人 `rm-f $(docker s-aq)`

清除所有缓存的网络:

当命令提示时按 "y" 码头工人网络修剪

最后, 如果您已经完成了本教程, 您还需要删除 fabcar 智能合约的基础链码映像。如果您是首次浏览此内容的用户, 则系统上不会有此链代码映像:

码头工人 `rmi dev-peer0.org1.例子.Com-法卡尔-1。0-5c906e402ed29f202600e4283216a75549e551e2e380f368265d8269ba`

实验室 2: 编写您的第一个区块链应用程序

安装客户端并启动网络

注意

以下说明要求您在工厂内的工厂分版目录中, 在您的本地克隆的设备样品存储库。在本教程的其余部分中, 请保留在此子目录的根目录中。

运行以下命令以安装应用程序的结构依赖项。我们关注的是结构 `ca-verent`, 这将允许我们的应用程序与 `ca` 服务器通信并检索标识材料, 以及与织物客户端进行通信, 从而使我们能够加载标识材料并与同行交谈并订购服务。

npm 安装

实验室 2: 编写您的第一个区块链应用程序

使用 `startfabrich shell` 脚本启动您的网络。此命令将启动我们的各种结构实体, 并启动一个智能合约容器, 用于使用 `golang` 编写的链码:

```
./startfabric.Sh
```

您还可以选择针对编写的链代码运行本教程。 [`node.js`](#). 如果要继续执行此路由, 请改为发出以下命令:

```
./startfabric.sh 节点
```

注意

请注意, `node.js` 链代码方案大约需要90秒才能完成;也许更长的时间。脚本没有挂起, 而是增加的时间是在构建链码映像时安装结构填充的结果。

实验室 2: 编写您的第一个区块链应用程序

好吧, 现在你有了一个示例网络和一些代码, 让我们来看看不同的部分是如何组合在一起的。

应用程序如何与网络交互

要更深入地了解我们 fabcar 网络中的组件 (以及它们的部署方式), 以及应用程序如何在更多的粒度级别上与这些组件交互, 请参见[了解 fabcar 网络](#)。

开发人员更感兴趣的是查看哪些应用程序**做**--以及查看代码本身以查看应用程序是如何构造的--应该继续下去。目前, 最重要的是要知道的是, 应用程序使用软件开发工具包 (sdk) 来访问**Api**允许查询和更新分类帐。

实验室 2: 编写您的第一个区块链应用程序

注册管理员用户

注意

以下两部分涉及与证书颁发机构的通信。在运行即将推出的程序时,您可能会发现流式传输 ca 日志非常有用。

若要流式传输 ca 日志,请拆分终端或打开新外壳,然后发出以下问题:

码头工人日志-f 卡.例子.Com

现在跳回您的终端与跑车内容..。

实验室 2: 编写您的第一个区块链应用程序

当我们启动我们的网络时, 管理员用户 (管理员) 已在我们的证书颁发机构注册。现在, 我们需要向 ca 服务器发送注册呼叫, 并检索此用户的注册证书 (ecert)。我们不会在这里深入了解注册详细信息, 但只要说 sdk 和我们的应用程序需要这个证书就足够了, 以便为管理员形成一个用户对象。然后, 我们将使用此管理对象随后注册和注册新用户。向 ca 服务器发送管理员注册呼叫:

节点注册管理员.js

此程序将调用证书签名请求 (csr), 并最终将 emert 和密钥材料输出到此项目的根目录下的新创建的文件夹 (hfc-key)。然后, 当我们的应用程序需要为我们的不同用户创建或加载标识对象时, 他们将查看此位置。

实验室 2: 编写您的第一个区块链应用程序

注册和注册用户1

使用我们新生成的 `adecert`, 我们现在将再次与 `ca` 服务器通信, 以注册和注册新用户。这个用户--`user1`--将是我们在查询和更新分类帐时使用的标识。在此需要注意的是, 是管理员身份为我们的新用户发出注册和注册呼叫 (即该用户是在注册商的角色)。发送注册并注册用户的呼叫 1:

节点注册器用户.Js

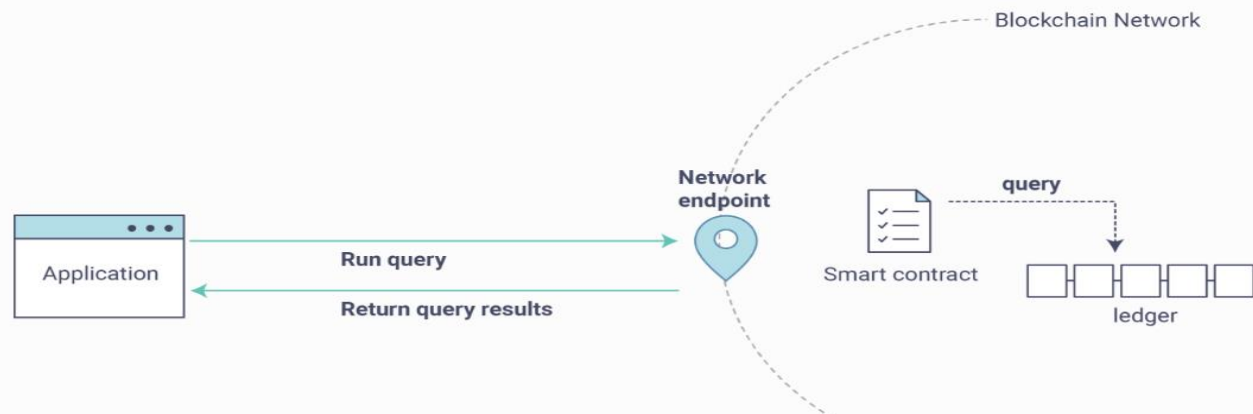
与管理员注册类似, 此程序调用 `csr`, 并将密钥和 `ecert` 输出到 `hfc-key-store` 子目录中。所以现在有了两个独立用户的身份资料--`admin` & `user1`。是时候和账簿交互了..。

实验室 2: 编写您的第一个区块链应用程序

查询分类帐

查询是您从分类帐中读取数据的方式。此数据存储为一系列键值对, 您可以查询单个键、多个键的值, 或者--如果分类帐是以 json 这样丰富的数据存储格式编写的--对其执行复杂搜索 (查找包含某些键的所有资产) 例如, ds)。

以下是查询工作方式的表示形式:



在运行来自区块链应用程序的查询时, 确实需要调用分类帐的智能协定。

实验室 2: 编写您的第一个区块链应用程序

首先, 让我们运行我们的查询.js 程序返回分类帐上所有汽车的列表。我们将使用我们的第二个标识--用户 1--作为此应用程序的签名实体。我们程序中的以下行指定用户1为签名者:

```
结构 _ 客户端.getUserContext('user1 ', 真实);
```

回想一下, 用户1注册材料已经放在我们的 hfc-keide-storesubococonbsib 中, 所以我们只需要告诉我们的应用程序获取该标识。定义了用户对象后, 我们现在可以继续从分类帐中读取。一个将查询所有汽车的函数, 查询 allcars, 在应用程序中预装, 因此我们可以简单地运行该程序, 如下所示:

节点查询.Js

实验室 2: 编写您的第一个区块链应用程序

它应该返回这样的东西:

已成功加载用户1从坚持
查询已完成, 检查结果

响应是 [{"Key":"CAR0", "记录": {"色": "蓝色", "制作": "丰田", "型号": "普锐斯", "所有者": "tomoko"}, {"Key":"CAR1", "记录": {"色": "红色", "制造": "福特", "模型": "野马", "所有者": "tomoko"}, {"Key":"CAR2", "记录": {"色": "绿色", "制造": "现代", "模型": "图森", "所有者": "金如此"}, {"Key":"CAR3", "记录": {"色": "黄色", "制造": "大众", "模型": "passat", "所有者": "max"}, {"Key":"CAR4", "记录": {"色": "白色", "制造": "tesla", "模型": "s", "所有者": "adriana"}, {"Key":"CAR5", "记录": {"色": "purple", "制造": "Peugeot", "模型": "205", "所有者": "Michel"}, {"Key":"CAR6", "记录": {"色": "white", "制造": "Chery", "模型": "S22L", "所有者": "Aarav"}, {"Key":"CAR7", "记录": {"色": "紫色", "制造": "菲亚特", "模型": "punto", "所有者": "pari"}, {"Key":"CAR8", "记录": {"色": "蓝色", "制造": "塔塔", "模型": "纳米", "所有者": "valeria"}, {"Key":"CAR9", "记录": {"色": "做": "holden", "模型": "bina", "所有者": "昭太郎"}}]

实验室 2: 编写您的第一个区块链应用程序

这是10辆车。阿德里安娜拥有的黑色特斯拉 s 型汽车、布拉德拥有的红色福特野马、帕里拥有的紫罗兰色菲亚特庞托等。分类帐是基于键值的, 在我们的实现中, 关键是 car0 通过 car9。这一点在某一时刻将变得特别重要。让我们仔细看看这个程序。使用编辑器 (例如原子或视觉工作室) 并打开**克雷**。

应用程序的初始部分定义了某些变量, 如通道名称、证书存储位置和网络终结点。在我们的示例应用程序中, 这些变量已被烘焙, 但在实际应用中, 这些变量必须由应用开发指定。

```
瓦尔通道=结构 _ 客户端.纽海峡 ("我的渠道");  
瓦尔同行=结构 _ 客户端.newPeer('grpc://localhost:7051');  
通道.点对 (对等);  
var 成员 _ user=空的; 空的  
var store _ path=路径.加入 (_ dirname, ' hfc-key-store ');  
安慰.日志 ("存储路径:"+存储路径);  
var tx _ id=空的; 空的
```

实验室 2: 编写您的第一个区块链应用程序

这是我们构造查询的块:

```
//查询汽车链码功能-需要1个参数, 例如: args: [' car4 '],  
//查询 allcars 链代码函数-不需要参数, 例如: args:[' '],  
康斯特请求={  
  //目标: ---让这个默认分配给被分配到通道链代码: ' fabcar ',  
  fcn: "查询所有的汽车",  
  args: [' ']  
};
```

当应用程序运行时, 它调用对等方上的 fabcar 链代码, 在其中运行 queryAllCarsfunction, 并且没有向其传递任何参数。

要查看我们的智能合同中的可用功能, 请导航到链码/fabc/to真目录在块样品和开放 fabck 的根目录。

实验室 2: 编写您的第一个区块链应用程序

注意

这些相同的函数是在场图链码的 `node.js` 版本中定义的。

您将看到我们有以下可供调用的功能: 内部分类器、查询车、查询汽车、创建车和变更车。

让我们仔细看看查询 `allcars` 函数, 看看它是如何与分类帐交互的。

```
func (s)*智能合同) 查询 allcars (apistab 垫片.链码-stubub接口)
```

```
Sc.响应 {启动键:="car0"
```

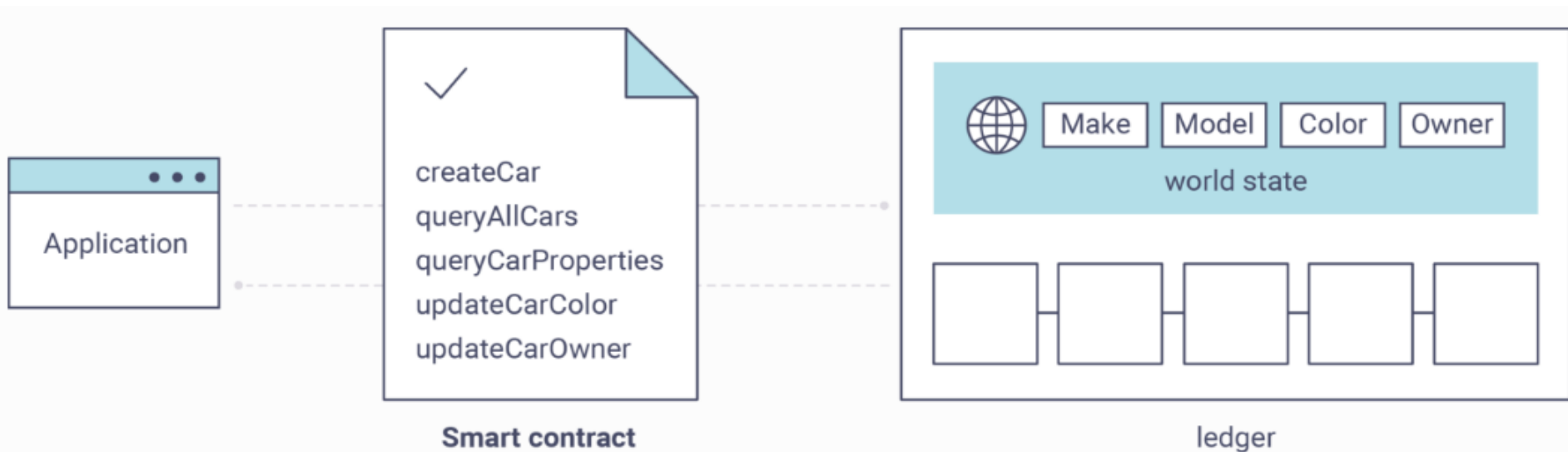
```
端键:="car999"
```

```
结果迭代器, 错误:=apistb.(开始键, 结束键)
```

这定义了查询 `allcars` 的范围。每个汽车之间的 `car0` 和 `car999`—总共 1,000 辆车, 假设每个关键已正确标记-将返回查询。

实验室 2: 编写您的第一个区块链应用程序

下面是应用如何调用链代码中的不同函数的表示形式。每个函数都必须根据链码填充接口中的可用 api 进行编码，这反过来又允许智能协定容器与对等分类帐正确接口。



访问区块链 api 的唯一方法是使用软件开发人员工具包 (sdk)。

实验室 2: 编写您的第一个区块链应用程序

我们可以看到我们的查询 allcars 功能, 以及一个称为 createcar 的函数, 它将允许我们更新分类帐, 并最终在瞬间将一个新的块追加到链中。

但首先, 返回到 query.js 程序并编辑构造函数请求以查询 car4。我们通过将查询中的函数从查询 allcar 更改为查询 car, 并将 car4 作为特定的键传递。

查询.js 程序现在应该如下所示:

```
康斯特请求={  
  //目标: ---让此默认值分配给分配给通道的对等方  
  链码: "法巴",  
  fcn: "查询车",  
  args: ["car4"]  
};
```

保存程序并导航回您的 fabcar 目录。现在再次运行该程序:

节点查询.Js

实验室 2: 编写您的第一个区块链应用程序

您应该看到以下内容:

```
{"颜色": "黑色", "制造": "特斯拉", "模型": "s", "所有者": "adriana"}
```

如果你回去看看之前我们询问每辆车的结果, 你可以看到 car4 是阿德里安娜的黑色特斯拉车型 s, 这是在这里返回的结果。

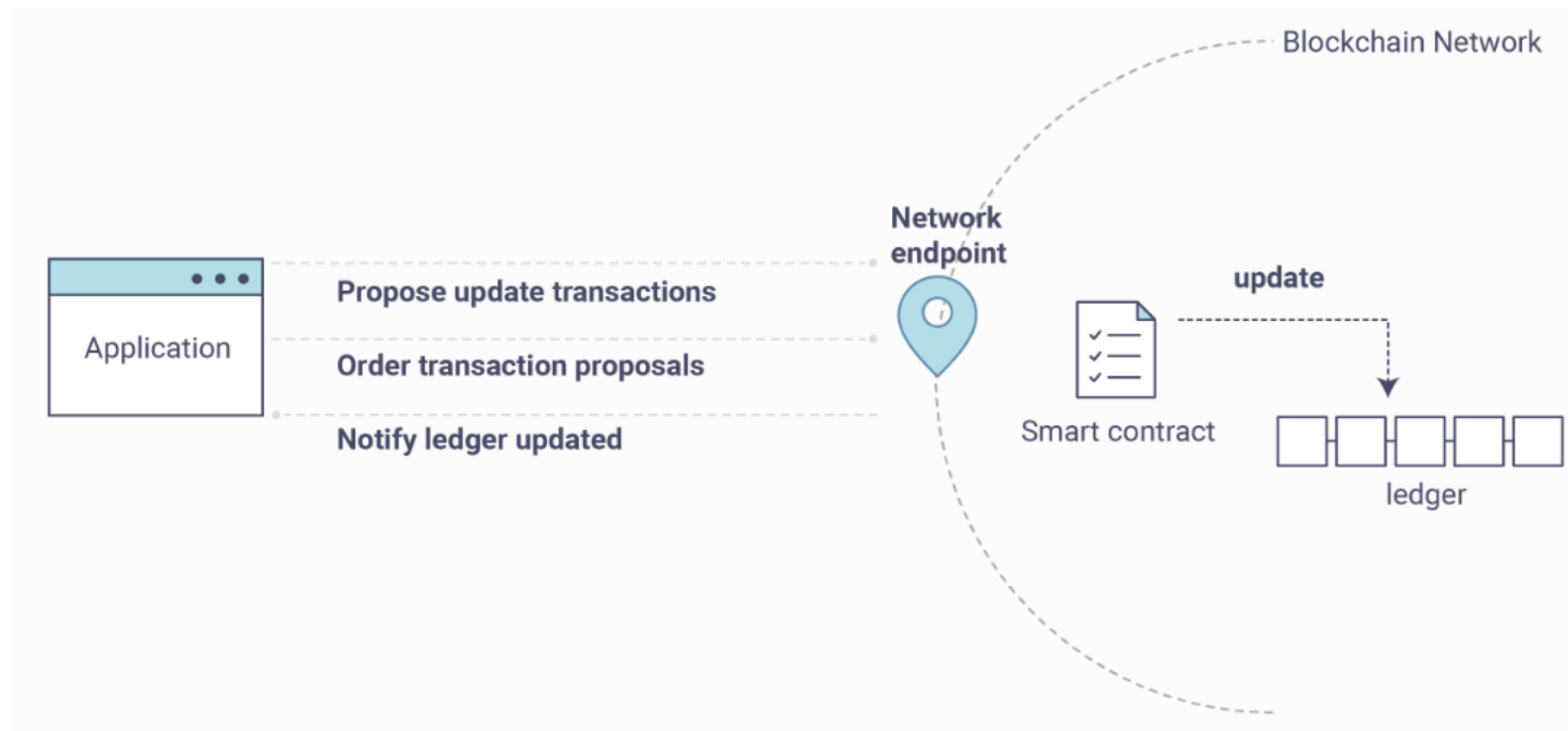
使用 querycar 功能, 我们可以对任何密钥 (例如 car0) 进行查询, 并获得与该车对应的任何品牌、型号、颜色和车主。

伟大。此时, 您应该对智能协定中的基本查询功能和查询程序中的少数参数感到满意。更新分类帐的时间..。

实验室 2: 编写您的第一个区块链应用程序

更新分类帐

现在, 我们已经完成了一些分类帐查询, 并添加了一些代码, 我们已经准备好更新分类帐了。我们可以进行很多潜在的更新, 但让我们从制造汽车开始。下面我们可以看到这个过程是如何工作的。建议进行更新, 对其进行批注, 然后返回到应用程序, 然后应用程序将其发送到订单并写入每个对等方的分类帐:



实验室 2: 编写您的第一个区块链应用程序

我们对分类帐的第一次更新将是创建一辆新车。我们有一个单独的 javascript 程序--vomk.js--我们将使用它进行更新。与查询一样, 使用编辑器打开程序并导航到代码块, 在其中我们构造调用:

```
//createcar 链码功能-需要 5 args, 前: [' car12 ', "本田", "雅阁", "黑色", "汤姆"],  
/变更车链代码功能-需要 2个 args, 前: args: [' car10 ', ' 巴里 '],  
//必须将该建议发送给认可同行  
    瓦尔请求={  
    //目标: 让默认分配给客户端的对等方  
    链码: "法巴",  
    fcn: "",  
    args: [' '],  
    链: "我的渠道",  
    txid:tx _ id  
    };
```

实验室 2: 编写您的第一个区块链应用程序

你会看到, 我们可以调用两个功能之一-创建汽车或变更车。首先, 让我们创建一个红色雪佛兰伏特, 并把它给一个老板叫尼克。我们的账簿上有 car9, 所以我们会在这里用 car10 作为识别钥匙。编辑此代码块, 使其如下所示:

```
瓦尔请求={  
  //目标: 让默认分配给客户端的对等方  
  链码: "法巴",  
  "创造汽车",  
  args: [' car10 ', "雪佛兰", "伏特", "红色", "尼克"],  
  链: "我的渠道",  
  txid:tx _id  
};
```

保存并运行该程序:
节点调用.Js

实验室 2: 编写您的第一个区块链应用程序

终端将有一些关于建议响应和承诺的输出。然而, 我们所关心的只是这个信息:
该事务已在对等本地主机上提交: 7053

若要查看此事务已写入, 请返回到 query.js, 并将参数从 car4 更改为 car10。

换句话说, 更改此:

```
康斯特请求={  
  //目标: ---让此默认值分配给分配给通道的对等方  
  链码: "法巴",  
  fcn: "查询车",  
  args: ["car4"]  
};
```

为此:

```
康斯特请求={  
  //目标: ---让此默认值分配给分配给通道的对等方  
  链码: "法巴",  
  fcn: "查询车",  
  args: ["car10"]  
};
```

实验室 2: 编写您的第一个区块链应用程序

再次保存, 然后查询:

节点查询.js

应返回此:

响应是{"颜色": "红色", "制作": "雪佛兰", "模型": "伏特", "所有者": "尼克"}

祝贺。你创造了一辆汽车!

所以现在我们已经做到了, 假设尼克感觉很慷慨, 他想把他的雪佛兰伏特给一个叫戴夫的人。要做到这一点, 请返回到车队.js, 并将函数从 createcar 更改为 "更换 carowner", 然后输入如下参数:

```
瓦尔请求={  
  //目标: 让默认分配给客户端的对等方  
  链码: "法巴",  
  fcn: "变形金刚",  
  args: [' car10 ', "dave"],  
  链: "我的渠道",  
  txid:tx _ id  
};
```

实验室 2: 编写您的第一个区块链应用程序

第一个论点--car10--反映了将改变车主的汽车。第二个论点--戴夫--定义了汽车的新车主。

再次保存并执行该程序:

节点调用.Js

现在, 让我们再次查询分类帐, 并确保 dave 现在与 car10 密钥相关联:

节点查询.Js

它应该返回此结果:

响应是{"颜色": "红色", "制作": "雪佛兰", "模型": "伏特", "所有者": "dave"}
car10 的所有权已从尼克改为戴夫。

注意

在现实世界中的应用程序中, 链代码可能会有一些访问控制逻辑。例如, 只有某些授权用户可以创建新车, 只有车主可以将汽车转移给其他人。

实验室 2: 编写您的第一个区块链应用程序

总结

现在我们已经完成了一些查询和一些更新,您应该对应用程序如何与网络交互有了相当好的认识。您已经了解了智能协定、**api** 和 **sdk** 在查询和更新中扮演的角色的基础知识,您应该了解如何使用不同类型的应用程序来执行其他业务任务和操作。

在随后的文档中,我们将学习如何实际**写**智能合同以及如何利用这些更低级别的应用程序函数 (特别是与身份和成员服务有关的功能)。

其他资源

中。[超分类结构节点 sdk 存储库](#)是一个很好的资源,用于更深入的文档和示例代码。您还可以咨询 "结构" 社区和组件专家[超帐火箭聊天](#)。

实验室 3: 建立自己的网络

在本练习中, 您将使用已设置的示例超分类分类分类分类帐结构网络, 其中包括两个组织 (每个组织维护两个对等节点) 和一个单独订购服务。该示例还使用一个容器来运行脚本执行, 该执行将连接到通道的对等方, 部署和实例化链代码, 并针对已部署的链代码推动事务的执行。

关于 "建立自己的网络" 模块的通知: 版权超收分类帐2017。修订 b29c9354。
根据知识共享归因4.0 国际许可获得许可

<https://creativecommons.org/licenses/by/4.0/legalcode>.

模块可在http://hyperledger-fabric.readthedocs.io/en/latest/build_network.html.

构建您的第一个网络

http://hyperledger-fabric.readthedocs.io/en/latest/build_network.html#building-your-first-network

向频道添加组织

http://hyperledger-fabric.readthedocs.io/en/latest/channel_update_tutorial.html

升级您的网络组件

http://hyperledger-fabric.readthedocs.io/en/latest/upgrading_your_network_tutorial.html

在结构中使用私有数据

http://hyperledger-fabric.readthedocs.io/en/latest/upgrading_your_network_tutorial.html

链码教程

<http://hyperledger-fabric.readthedocs.io/en/latest/chaincode.html>

面向企业的区块链平台

<http://hyperledger-fabric.readthedocs.io/en/latest/index.html>

构建您的第一个网络。

查看说明

在以下文件上:

- 构建您的第一个网络。
- 将组织添加到通道中. doc
- 升级网络组件. doc
- 在法理学中使用私有数据
- 链码教程. doc
- 使用 couchdb. doc
- <http://hyperledger-fabric.readthedocs.io/en/latest/videos.html>
- 操作指南. doc
- 命令引用. doc
- 体系结构参考. doc
- 常问的问题

<http://hyperledger-fabric.readthedocs.io/en/latest/>

s乌姆玛丽

现在,您已经完成了会话,您已经了解了可用于建模、构建和维护区块链业务网络的概念、方案 and 应用程序。更具体地说,您应该了解:

- 如何使用超级分类编辑器建模和构建一个简单的业务网络以及编辑器的结构
- 超分类结构的组件和结构, 以及如何与它进行交互
- 区块链解决方案组件, 如钱包、分类帐、参与者、共识、安全性和智能合同
- 帮助构建、建模、运行和维护区块链业务网络的人员的主要考虑因素和责任:
 - 开发人员
 - 管理员
- 达成共识的方法
- 可用于与网络交互的工具和应用程序

结束!

धन्यवाद

Hindi 印地
语

多謝

繁体中文

ขอบพระคุณ

泰语

Спасибо

俄语

谢谢

西班牙语

谢谢

英语

شكراً

阿拉伯语

奥布里加
多

葡萄牙语

格拉齐

意大利
语

多谢

简体中文

丹克

德语

谢谢

法语

நன்றி

Tamil

泰米尔
语

ありがとうございました

日语

감사합니다

朝鲜语

实验室: 设置高度分类编辑器游乐场

请使用此 url:<https://composer-playground-unstable.mybluemix.net/login>

点击**部署新的业务网络**.

在下一页上, 为网络命名, 向下滚动并选择汽车拍卖网络, 然后单击**部署**在页面的右侧。

在显示您的网络的主页上, 单击**立即连接**以与网络合作。

实验室: 超级分类编辑器游乐场

Welcome to Hyperledger Composer Playground!



In this web sandbox, you can deploy, edit and test business network definitions. Have a play and learn what Hyperledger Composer Playground is all about.

Let's Blockchain!



Not sure where to start? View our Playground tutorial.

实验室: 超级分类编辑器游乐场

Basic Sample Business Network

This is the "Hello World" of Hyperledger Composer samples, which demonstrates the core functionality of Hyperledger Composer by changing the value of an asset.

This business network defines:

Participant `SampleParticipant`

Asset `SampleAsset`

Transaction `SampleTransaction`

Event `SampleEvent`

SampleAssets are owned by a SampleParticipant, and the value property on a SampleAsset can be modified by submitting a SampleTransaction. The SampleTransaction emits a SampleEvent that notifies applications of the old and new values for each modified SampleAsset.

To test this Business Network Definition in the **Test** tab:

Create a `SampleParticipant` participant:

```
{
  "$class": "org.example.basic.SampleParticipant",
  "participantId": "Toby",
  "firstName": "Tobias",
  "lastName": "Hunter"
}
```

实验室: 超级分类编辑器游乐场

Create a `SampleAsset` asset:

```
{
  "$class": "org.example.basic.SampleAsset",
  "assetId": "assetId:1",
  "owner": "resource:org.example.basic.SampleParticipant#Toby",
  "value": "original value"
}
```

Submit a `SampleTransaction` transaction:

```
{
  "$class": "org.example.basic.SampleTransaction",
  "asset": "resource:org.example.basic.SampleAsset#assetId:1",
  "newValue": "new value"
}
```

After submitting this transaction, you should now see the transaction in the Transaction Registry and that a `SampleEvent` has been emitted. As a result, the value of the `assetId:1` should now be `new value` in the Asset Registry.

Congratulations!

实验室: 1 设置超级分类帐编辑游乐场

实验室: 2 在区块链网络中转移资产

实验室: 3 浏览编辑器视图和存档数据

区块链-卡 Auction_part1.pdf

区块链-卡 Auction_part2.pdf

区块链-卡 Auction_part3.pdf