記事
Toshihiko Minamoto · 2021年5月11日 9m read

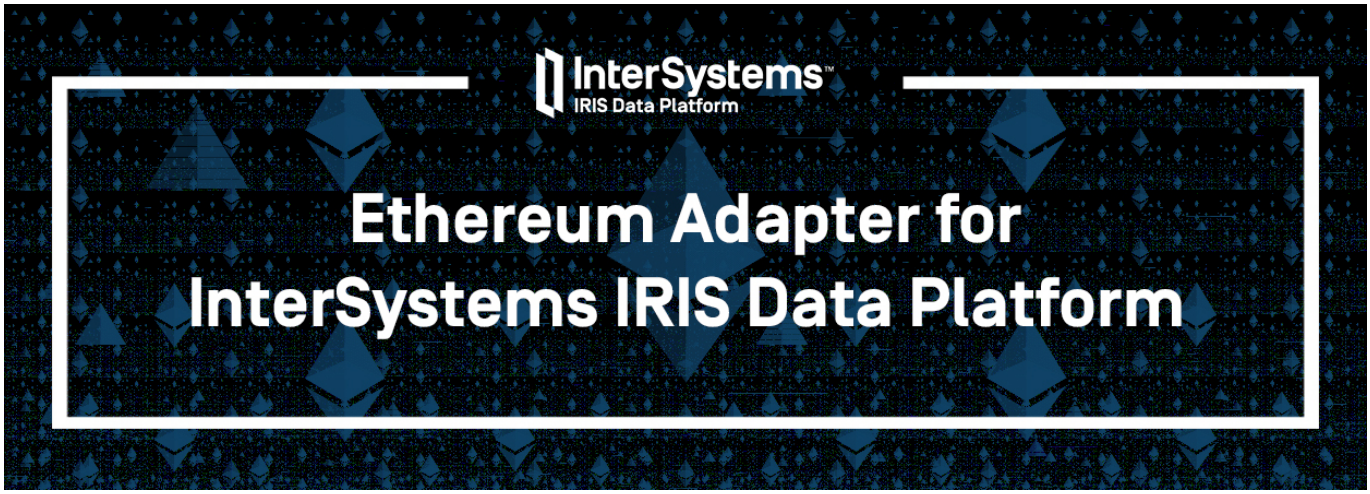
Open Exchange

InterSystems IRIS Data Platform の Ethereum アダプター

1. ブロックチェーン

この記事を書いている時点(2019年2月)で、ビットコインの価値はそれが絶頂期だったころの 5 分の 1 未満に下落しています。 そのため、ブロックチェーンの私の体験について誰かに話すときに最初に耳にするのは、「今頃ブロックチェーンを欲しがる人がいるのか」という偽りなく懐疑的な言葉です。

その通り。ブロックチェーンの盛り上がりは衰退しています。 ところが、それが基づいているテクノロジーはここにとどまり、特定の分野で使用され続けるでしょう。一般的にインターネットではこのテクノロジーの使用方法が記述された資料でいっぱいです。



（たとえばmedium と forbes に掲載されています）。

ご存知のように、ブロックチェーンは分散レジストリ、すなわち、レジストリの完全なコピーを格納している複数のノードに分散されたデータベースです。 レコード（トランザクション）がブロックを形成し、ブロックがブロックのチェーンを形成するところに、ブロックチェーンの鍵となる機能があります。
ブロックチェーンはアペンド演算のみをサポートしているため、 ブロックチェーンにすでに保存されたトランザクションに変更を加えることは、ほぼ不可能ということになります。

ブロックチェーンに関するチュートリアルはインターネット上にたくさんあります（ブロックチェーンについて聞いたことがない方は、こちらの簡単な動画をまずご覧ください）。

ブロックチェーンが順調な成果を見せていた時、文字通りどこにでも、そのテクノロジーを使用するための呼び出しが複数ありました。
ただし、ブロックチェーンが必要となる可能性のあるプロジェクトやタスクには、明確な特性があります。

まず第一に、一貫性と信頼性のある大量のデータを書き込むプレーヤー/ユーザーがたくさん存在する必要がある点です。

そうであれば、誰もが信頼するサードパーティは存在しないはずです。

公開データ検証の仕組みが必要となります。
こういった条件のすべてを満たす場合、ブロックチェーンの使用を検討するのも良いかもしれません。

こういったタスクは、どの業界にも見られます。 www.101blockchains.com プロジェクトでは、潜在的および既存のブロックチェーンプロジェクトに関する情報と、様々な業界でブロックチェーンテクノロジーを使用するニュアンスを集約しています。

たとえば、ブロックチェーンは、ヘルスケアの分野で次のタスクに使用することができます。

患者記録の安全なリモート管理

サプライチェーン全体で変更不可能なトランザクションによる、偽造医薬品への対策

詐欺やデータ改ざんの可能性の排除による、臨床試験の監視と有効性の改善

企業部門は通常、プライベート許可型ブロックチェーンと呼ばれる特殊なブロックチェーンを使用しています。
このようなネットワークには、トランザクションを検証するための特別なノードセットがあります。

しかし、最初の InterSystems IRIS
ブロックチェーンアダプター
を開発する際、非許可型ブロックチェーンに区分されるイーサリアム
という種類のブロックチェーンを選択しました。イーサリアムは、単一のコントロールセンターを持たないオープンプラットフォームです。 これは、このブロックチェーンエンジンの人気と、多くのツールやライブラリを備え、十分に成熟したインフラストラクチャであることを基に採用しました。
イーサリアムのツールを使えば、誰でもプライベートブロックチェーンを作成することができます。

2. アダプター

では、アダプターの話に戻りましょう。

InterSystems IRIS のアダプターは（Ensemble でも同様に）、外部システムと対話可能にする InterSystems IRIS クラスのクラスまたはパッケージです。 InterSystems IRIS アダプターは、インバウンド（外部システムが対話を開始した場合に外部システムからデータを受信）とアウトバウンド（InterSystems IRIS
が対話を開始した場合に外部システムと連携）に分けられます。

IRIS Ethereum アダプターはアウトバウンドアダプターであり、ほかのほとんどの InterSystems IRIS
アダプターとはわずかに異なります。 このアダプターには、小さな NodeJS モジュールも含まれています。 図 1 はこのアーキテクチャを示しています。

図 1.

アダプターの NodeJS モジュールではイーサリアムと連携するための NodeJS ライブラリを使用します。

アダプターは次のことを行います。

スマートコントラクトをイーサリアムにデプロイする（スマートコントラクト、開発ツール、および例を網羅した記事を別途掲載する予定です）。

スマートコントラクトのメソッド（ブロックチェーンの状態を変更するメソッドと変更しないメソッド）を呼び出す。

トランザクションを保存する（ウォレット間で資金を送金する）。

ブロックチェーンの状態を取得するほかのメソッドを呼び出す。

すべてのリクエストをログに記録する（NodeJS モジュールによって行われ、デバッグに有効）。

OpenExchange上のアダプターのソースコード。

3. 簡単な例

アダプターには「Hello World」サンプルが付属しています。

イーサリアムを使用するには（このサンプルを実行するには）以下の作業を行います。

使用するネットワークを選択します。 Ropsten などのテストネットワークは通常、開発目的で使用できます。

このネットワークにウォレットを作成して、資金を追加します。

ローカルイーサリアムクライアント（ Geth など）をインストールするか、クラウドプロバイダー（ Infura など）と連携するための鍵を取得します。

ビジネスオペレーションを構成する際は、次の項目を設定する必要があります（図 2）。

NodeJS モジュールが動作するサーバーとポート（デフォルトポートは 3000 ）

プロバイダーの設定（この場合、Infura へのアクセス）

ログイン情報（ユーザー名にウォレット番号、パスワードに秘密鍵を指定します。 InterSystems IRIS はログイン情報を別のデータベースに保存します。このデータベースは、暗号化を有効にする必要があります。）

図 2.

スマートコントラクトを使用するには、ファイルシステム内に（使用するスマートコントラクトごとに）フォルダを作成し、次の 2 つのファイルを配置する必要があります。
*abi.txt
*bytecode.txt

これらのファイルには、スマートコントラクトの ABI とそのバイトコードが含まれている必要があります。
スマートコントラクトの ABI は、JSON 形式によるインターフェースの正式な記述です。 ABI
とバイトコードは、スマートコントラクトのコンパイル時に作成されます。

バイトコードは、コントラクトのデプロイに必要です。

InterSystems IRIS 相互運用性テストサービスを使用して、ビジネスオペレーションをテストできます。

図 3 では、テストサービスを使用してどのようにスマートコントラクトがデプロイされるのかを示しています。
このビジネスオペレーションを呼び出すと、トランザクションのハッシュを含むメッセージが結果として返されます。

図 3.

ropsten.etherscan.io（https://etherscan.io/
）ブラウザを使用してこのトランザクションを見つけ、デプロイされたスマートコントラクトのアドレスを取得できます。

アダプターを使用してスマートコントラクトのメソッドを呼び出すには、運用構成の ContractFolder と
ContractAddress フィールドに入力する必要があります。

スマートコントラクトの実行コードは非常に単純です。

```
set ..ContractABI = [].%FromJSON(..ContractFolder_"abi.txt")
set contract = ..Adapter.GetContract(##class(Ethereum.Address).%New(..ContractAddress
),..ContractABI)
set result = contract.hello()
set pResponse = ##class(Ens.StringContainer).%New(result.args)
```

スマートコントラクトのアドレスと ABI をアダプターの GetContract
メソッドに渡し、メソッドを呼び出すために使用するスマートコントラクトオブジェクトを作成します。
この場合、文字列を返す hello() メソッドはスマートコントラクトで定義されている必要があります。

この例では、hello()
メソッドはブロックチェーンの状態を変更しないため、同期的に呼び出すことができますが、 ブロックチェーンの状態を変更するメソッドの実行時間は非常に長くなることがあります（トランザクションの検証を待つ必要があるため）。

このようなメソッドを呼び出すには、InterSystems IRIS が提供する遅延応答の仕組みを使用します。
アダプターは遅延応答トークンを送信する必要があり、トランザクションが承認されると、NodeJS
モジュールはその実行結果を InterSystems IRIS に渡します。 これを行うには、Web
アプリケーションを構成し、受信した応答を処理する追加のビジネスサービスを運用に追加する必要があります。

以下は、ブロックチェーンの状態を変更するメソッドを呼び出すためのコードです。

```
// EthereumAccount???????? privateKey ???
do ##class(Ens.Config.Credentials).GetCredentialsObj(.cred, "Ethereum.Demo.EthereumOp
eration", "Ens.Config.Credentials", ..Credentials)
set account = ##class(Ethereum.Address).%New(cred.Username)
set privateKey = cred.Password
```

```
//?????? ABI ?????
set ..ContractABI = [].%FromJSON(..ContractFolder_"abi.txt")

// ??????????????
set contract = ..Adapter.GetContract(##class(Ethereum.Address).%New(..ContractAddress
),..ContractABI)
$$$ThrowOnError(..DeferResponse(.deferred))

// gas ???
do contract.SetOptions(account)
set gasJSON = contract.EstimateGas("setName",pRequest.Name)
$$$TRACE(gasJSON.gas)

do contract.SetOptions(account, privateKey,  ##class(Ethereum.Wei).%New(1000000000),
##class(Ethereum.Wei).%New(100*gasJSON.gas),,deferred)

set result = contract.setName(pRequest.Name)
```

この場合、スマートコントラクトの setName()
メソッドを呼び出す前に、遅延応答トークンを含む多数のパラメーターを指定する必要があります。

次の記事では、スマートコントラクトについて詳しく説明し、InterSystems IRIS Ethereum
アダプターを使って実際の問題を解決する例を示します。

#相互運用性 #Ensemble #InterSystems IRIS
InterSystems Open Exchangeで関連アプリケーションを確認してください

ソースURL:https://jp.community.intersystems.com/post/intersystems-iris-data-platform-%E3%81%AE-ethereum-%E3%82%A2%E3%83%80%E3%83%97%E3%82%BF%E3%83%BC