ThemisStruct
テミストラクト

テミストラクト

アイデンティティプラットフォーム

「ThemisStruct Identity Platform」 クックブック 2018

～ 共通ID基盤、API連携認証システムを題材に、デモを交えて～

株式会社オージス総研
事業開発本部 テミストラクトソリューション部

鈴木 竜司

統合認証パッケージ

Themistruct Identity Platform のご紹介

統合認証パッケージ Themistruct Identity Platform

Themistruct Identity Platform は、ITシステム利用者のアイデンティティ管理と認証の機能を提供します。顧客向けにサービスを展開したり、従業員向けにアプリケーションを展開する際に必要となる、共通ID基盤の構築に活用いただけます。

本ご紹介ではこちら
を中心に説明

顧客向けサイト領域に活用

サービスサイトの
ID、認証の共通機能として利用

エンタープライズ領域に活用

エンタープライズアプリ群の
SSOやアクセス制御の基盤として利用

『顧客向けサイト』に向けた3大特長

□ 短期間での導入が可能

- AWSマネージドサービスのコンポーネントを活用し、導入時におけるキャパシティやアベイラビリティプランニングから解放します。スモールスタート、突発的アクセス集中に対応できる伸縮可能かつ事業継続性を考慮した認証基盤が、2時間でセットアップできます。

✓ High-Availability

✓ Scalability

□ IDとユーザー認証、アクセス認可の機能をアプリに組み込み

- 利用者のID情報の管理や認証など利用者IDに関連する共通機能の実装を支援する『フレームワーク』と『Web API』を提供します。これらを活用し、貴社のサービスサイトに認証機能やパスワード変更機能、アプリケーションへのID連携機能などを組み込むことができます。

□ Web APIの認証・アクセス認可に対応

- OAuth 2.0 の技術仕様に基づいた認証・アクセス認可機能を提供します。OAuth 2.0 を使ったAPI連携認証システムが実現でき、モバイルアプリ、オープンAPIを安全に公開できます。

本講演でのご紹介内容

□ 短期間での導入が可能

- AWSマネージドサービスのコンポーネントを活用し、導入時におけるキャ
ご紹介1：セットアップしてみる します。スモールスター
の事業継続性を考慮した
認証基盤が、2時間でセットアップできます。

- ✓ High-Availability
- ✓ Scalability

□ IDとユーザー認証、アクセス認可の機能をアプリに組み込み

- 利用者のID情報の管理や認証など利用者IDに関連する共通機能の実装を支
ご紹介2：共通ID基盤として構成してみる ます。スモールスター
の事業継続性を考慮した
認証基盤が、2時間でセットアップできます。

□ Web APIの認証・アクセス認可に対応

- OAuth 2.0 の技術仕様に基づいた認証・アクセス認可機能を提供しま
ご紹介3：API連携認証システムとして構成してみる ます。スモールスター
の事業継続性を考慮した
認証基盤が、2時間でセットアップできます。

セットアップしてみる

セットアップの概要：セットアップは4ステップで実現

ThemiStruct Identity Platformのシステム構成

【セットアップ手順】

ステップ1：VPCの作成

ステップ2：SNS Topicの作成

ステップ3：ThemiStruct Identity Platformのインストーラー配備

ステップ4：インストーラーで残るコンポーネントを構築

ステップ1. VPCを作成

■ CloudFormationとテンプレートを用いてVPCを作成

ThemiStruct Identity Platformより提供

identityplatform-vpc.json

VPCのサブネット情報入力
Identity Platform専用のネットワ
ークのため、基本は入力値のまま

ステップ 1. VPCを作成

スタックの作成 | アクション | テンプレートのデザイン

フィルター: アクティブ | スタックの名前でフィルター | 24 個のスタックを表示中

スタックの名前	作成時間	状況	説明
☒ ThemisStruct-IP-TEST-vpc	2017-08-15 10:29:38 UTC+0900	CREATE_IN_PROGRESS	AWS CloudFormation Template : Create VPC settings and initialize Network settings of Identity Platform.

スタックの作成 | アクション | テンプレートのデザイン

フィルター: アクティブ | スタックの名前でフィルター | 24 個のスタックを表示中

スタックの名前	作成時間	状況	説明
☒ ThemisStruct-IP-TEST-vpc	2017-08-15 10:29:38 UTC+0900	CREATE_COMPLETE	AWS CloudFormation Template : Create VPC settings and initialize Network settings of Identity Platform.

ThemisStruct-IP-TEST-vpc

その他のアクション | スタックの更新

スタックの名前: ThemisStruct-IP-TEST-vpc

スタック ID: arn:aws:cloudformation:ap-northeast-1:██████████:stack/ThemisStruct-IP-TEST-vpc/3408e4c0-8159-11e7-b42d-50d5ca9ff4ae

状況: CREATE_COMPLETE

状況の理由:

IAM ロール:

説明 AWS CloudFormation Template : Create VPC settings and initialize Network settings of Identity Platform.

▼ 出力

キー	値	説明	エクスポート名
Segment1cPublic	subnet-984d79c0	Segment1c public id	
Segment1cPrivate	subnet-2540747d	Segment1c private id	
VPC	vpc-a94460cd	VPC id	
Segment1aPrivate	subnet-3fe21276	Segment1a private id	
Segment1aPublic	subnet-16e4145f	Segment1a public id	

約10分で作成完了

作成結果の出力は後工程の入力で利用

2. SNS Topicを作成

■ CloudFormationとテンプレートを用いてSNS Topicを作成

テンプレートの選択 テンプレートは、スタックのリソースとそのプロパティを説明する、JSON/YAML形式のデータ

◎ サンプルテンプレートの選択

□ テンプレートを Amazon S3 にアップロード

ファイルを選択

◎ Amazon S3 テンプレート URL の指定

identityplatform-sns.json

スタックの作成

テンプレートの選択 詳細の指定

詳細の指定

スタック名とパラメータ値を指定します。AWS CloudFormation テンプレートに定義づけられるデフォルトのパラメータ値を使用、または変更することができます。詳細はこちら。

スタックの名前 ThemStruct-IP-TEST-sns

パラメータ

MailAddress email@example.com Please specify the E-mail Address

TopicName samplesnstopicstest Please specify the SNS Topics name

MailAddress (監視アラート通知先)

TopicName (トピックの識別子)

ThemStruct-IP-TEST-sns

その他のアクション ▼ スタックの更新

スタックの名前: ThemStruct-IP-TEST-sns

スタック ID: arn:aws:cloudformation:ap-northeast-1:123456789012:stack/ThemStruct-IP-TEST-sns/45451f00-8164-11e7-8dde-50fa13f1d0ad

状況: CREATE_COMPLETE

状況の理由:

IAM ロール:

説明 CloudFormation create CloudWatch Alarm

▼ 出力

キー	値	説明
MySNSTopicTopicARN	arn:aws:sns:ap-northeast-1:123456789012:samplesnstopicstest	SNS Topics ARN.

約10分で作成完了

作成結果の出力は後工程の入力で利用

3. インストーラの配備

CloudFormationとテンプレートを用いてThemiStruct Identity Platformインストーラを配備

詳細の指定

スタック名とパラメータを指定します。AWS CloudFormation テンプレートに定義づけられるデフォルトのパラメータを使用、または変更することができます。詳細はこちら。

スタックの名前

パラメータ

Network Configuration

VPC または Name タグ値で検索
Specifies the VPC ID to launch the EC2 of the Management Console.

PublicSubnet または Name タグ値で検索
Specifies the Subnet ID to launch the EC2 of the Management Console.

AllowedIPAddress Specifies the IP address(CIDR) to allow access to EC2 of the Management Console. (e.g. 192.0.2.0/24)

EC2 instance of Management Console Configuration

HostName Specifies the host name of Management Console.

DomainName Specifies the domain name managed on Route 53. It's set automatically as the domain name of the Identity Platform. (e.g. example.com)

SSHUserName Specifies the ssh user name of Management Console. (e.g. operatioruser)

SSHPublicKey Specifies the public key connecting to EC2 of the Management Console via SSH.

SetupUserName User Name for Setup/Update Signin.

SetupPassword User Password for Setup/Update Signin.

CodeCommit Configuration

RepositoryAccessKey Specifies the key of access to the software repository.

RepositorySecretKey Specifies the secret of access to the software repository.

Others

NotificationTopic Specifies the Notification Topic ARN.

Identity Platformのソフトウェアダウンロードキー

インストーラのホスト名/ドメイン名

インストーラへログインするためのユーザ名/パスワード

インストーラ用サーバ証明書 (HTTPS実現用)

インストーラのOSのユーザ名

インストーラへアクセス (SSH) 用の公開鍵

(前工程完了時の出力結果)

4. インストーラを実行

ThemisStruct-IP-TEST-setup

スタックの名前: ThemisStruct-IP-TEST-setup
スタック ID: arn:aws:cloudformation:ap-northeast-1:111111111111:stack/ThemisStruct-IP-TEST-setup/631da160-8181-11e7-bb71-50d5ca9ff4c6
状況: CREATE_COMPLETE
状況の理由:
IAM ロール:
説明 AWS CloudFormation Template : Create Management Console of Identity Platform.

▼ 出力

キー	値	説明
SetupWizardURL	インストーラURL	A URL that access to Management Console of Identity Platform.

約20分で作成完了

その他のアクション ▼

スタックの更新

Identity Platform Setup Signin

セットアップ、システムアップデートを利用するためにサインインしてください。
システム構築時に入力したユーザー名、パスワードを入力してください。

ユーザー名

パスワード

サインイン

4. インストーラを実行

Identity Platformのセットアップ

Step 1 共通設定

Step 2 データベース設定

Step 3 APIサービス設定

Step 4 管理コンソール設定

Step 5 セットアップ実施確認

共通設定

APIサービスのバージョン v1.4.0

データベーススキーマのバージョン v1.4.0

管理コンソールのバージョン v1.4.0

ユーザーインターフェイスのバージョン v1.4.0

プライベートサブネットのID (1a) *

プライベートサブネットのID (1c) *

APIサービスのホスト名 .dev.thematrix.net

APIサービスの構築環境の種類

管理者のパスワード

管理者のパスワード(確認)

Previous Next

共通の設定

認証/APIサービスのホスト名

構築環境の種別

システム管理者のパスワード

(前工程完了時の出力結果)

DBの設定

データストアタイプの選択

冗長性：Amazon Aurora

低冗長性：RDS for MySQL

データベースのMulti-AZ配置 (有効or 無効)

データベースのインスタンスクラス

データベースのストレージサイズ

データベースのマイナーバージョン自動更新 (有効or無効)

週次メンテナンス期間：上記の実施曜日/時間

バックアップの保存期間

データベースリソース/通信の暗号化 (有効or 無効)

Identity Platformのセットアップ

Step 1 共通設定

Step 2 データベース設定

Step 3 APIサービス設定

Step 4 管理コンソール設定

Step 5 セットアップ実施確認

データベース設定

データストアタイプの選択 冗長性 低冗長性

データベースのMulti-AZ配置 無効 有効

データベースのインスタンスクラス

バックアップの保存期間

データベースリソースの暗号化 利用する 利用しない

データベースインスタンスへの通信の暗号化 利用する 利用しない

Previous Next

4. インストーラを実行

Identity Platformのセットアップ

1 Step 1 共通設定
2 Step 2 データベース設定
3 Step 3 APIサービス設定
4 Step 4 管理コンソール設定
5 Step 5 セットアップ実施確認

APIサービス設定

APIサービスのサーバ証明書* C:\fakepath\cert.pem

APIサービスのサーバ証明書の秘密鍵* C:\fakepath\private.pem

APIサービスの中間証明書* C:\fakepath\chain.pem

Previous Next

Identity Platformのセットアップ

1 Step 1 共通設定
2 Step 2 データベース設定
3 Step 3 APIサービス設定
4 Step 4 管理コンソール設定
5 Step 5 セットアップ実施確認

管理コンソール設定

管理コンソールのサーバ証明書 C:\fakepath\cert.crt

管理コンソールのサーバ証明書の秘密鍵 C:\fakepath\private.pem

管理コンソールの中間証明書 C:\fakepath\chain.pem

Previous Next

認証画面やAPIのHTTPS設定

APIサービスのサーバ証明書秘密鍵

APIサービスのサーバ中間証明書

APIサービスのサーバ証明書

管理コンソールのHTTPS設定

管理コンソールのサーバ証明書秘密鍵

管理コンソールのサーバ中間証明書

管理コンソールのサーバ証明書

4. インストーラを実行

セットアップ実施中

23% 完了

データベース構築中

セットアップ処理一覧

作業完了	共通設定確認
作業完了	踏み台構築
作業中	データベース構築
未実施	APIデプロイ準備
未実施	データベース初期設定
未実施	APIデプロイ設定
未実施	APIデプロイ実施
未実施	APIサービス証明書適用
未実施	コンテンツデプロイ準備
未実施	コンテンツデプロイ
未実施	コンテンツデリバリネットワーク構築
未実施	マネージメントコンソール証明書適用
未実施	マネージメントコンソール設定

約80分で作成完了

Identity Platformのセットアップ

セットアップが完了しました

システムを利用する際は、以下のリンクにアクセスしてください。

Identity PlatformのサインインURL : サインインURL

管理コンソールのURL : 管理コンソールURL

Themistruct Identity Platform

管理者ID

サインイン

Copyright © 2016-2017 OGIS-RI Co., Ltd. All Right Reserved.

合計約120分でセットアップ完了

共通ID基盤として構成してみる

共通ID基盤の構成イメージ

構築 STEP 1

- ✓ サインイン画面にThemiStruct Identity Platformの認証機能を組み込む。

構築 STEP 2

- ✓ 会員管理機能で登録・更新される情報をThemiStruct Identity Platformに連携する。

構築 STEP 3

- ✓ サービスサイトと認証サーバでID連携を行い、SSOやユーザーの共通属性利用を実現する。

STEP1 : サインイン画面に認証機能を組み込む

ThemiStruct Identity Platform が提供するSDKを利用して認証機能を組み込み

1. サインイン画面のHTMLに認証機能（認証フォーム）を配置したい箇所に以下のタグを追加する

```
<div id="signin-container"></div>
```

2. SDKの読み込みタグを追加する

```
<script  
  src="https://${Identity PlatformのFQDN}/public/javascripts/signin.bundle.min.js">  
</script>
```

2つのタグを画面に埋め込むことで独自画面に
認証機能を組み込むことが可能！

STEP1 : サインイン画面に認証機能を組み込む

■ 認証機能を実際に組み込んだ設定と動作画面

```
<h1 id="page_title">会員ログイン (ThemiStruct Identity Platform組み込みあり) </h1>
<div class="typel noMargin_b">
  <div class="panel">
    <div class="login overlap">
      <!--tsIP Login form start -->
      <div id="signin-container">
        <script src="public/javascripts/lang.js"></script>
        <script src="https://x.x.x.x/public/javascripts/signin.bundle.min.js"></script>
      </div>
    </div>
  </div>
</div>
```

SDKが認証フォームを自動生成

```
<div class="signin-input">
  <input id="userName" class="form-control" data-id="18" type="text" />
  <input id="password" class="form-control" name="password" type="password" />
  <button id="signin-button" class="btn btn-info" data-id="19" type="button" />
</div>
```

STEP2：会員管理機能で登録・更新される情報を連携する

会員登録の実現イメージ

STEP2 : 会員管理機能で登録・更新される情報を連携する

ThemiStruct Identity Platform のアカウント管理APIを利用

■ SCIM (System for Cross-domain Identity Management) に準拠したアカウント管理APIを提供

エンドポイント	メソッド	概要
/account/v2/Users	POST	リクエストパラメータに指定されたユーザ情報でユーザを登録する
/account/v2/Users/{id}	GET	「{id}」で指定されたユーザのユーザ情報を取得する
/account/v2/Users/{id}	PUT	「{id}」で指定されたユーザのユーザ情報を更新する
/account/v2/Users/{id}	DELETE	「{id}」で指定されたユーザを削除する

STEP2 : 会員管理機能で登録・更新される情報を連携する

アカウント登録 (POST /account/v2/Users) のリクエストデータ

```
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:custom:scim:schemas:1.0:User",
    "urn:identityplatform:scim:schemas:1.0:User"
  ],
  "userName": "emp0001",
  "nickName": "Tanaka-san",
  "active": true,
  "password": "I0darAVFGV0I5Emzclj",
```

【続き】

```
"urn:identityplatform:scim:schemas:1.0:User": {
  "jpName": {
    "familyName": "田中",
    "familyNameKana": "タナカ",
    "givenName": "哲也",
    "givenNameKana": "テツヤ"
  }
},
"urn:custom:scim:schemas:1.0:User": {
  "employeeType": "A100",
  "organaizationCode": "B2000",
  "positionCode": "C30"
}
}
```

STEP3 : サービスサイトと認証サーバでID連携を行う

サービスサイトにOpenID ConnectでのID連携組み込み

Apacheで構築されたサービスサイトにApacheのモジュール「mod_auth_openidc」を組み込み、ID連携を行う仕組みであるOpenID Connectを利用したID連携を構築してみます

■ インストール

```
# yum install -y mod_auth_openidc
```

■ インストール後の構築手順

1. ThemiStruct Identity PlatformへID連携を行うクライアントとして登録する
2. mod_auth_openidcの設定を実施する
3. サービスサイト側でmod_auth_openidcから渡されるユーザ情報を受け取る処理を組み込む

STEP3 : サービスサイトと認証サーバでID連携を行う

■ 認証サーバへID連携を行うクライアント（サービスサイト）を登録

クライアント

作成

アプリケーション名* DemoServiceSite

Clientの種類 Confidential Public

リダイレクトURI*
Redirect URI + Add URI
https://www.example.com:443/callback

登録

クライアント

編集 削除

登録後払い出されたIDとシークレットキー
mod_auth_openidc側の設定で利用

編集

クライアントID LA8ZyY8leczumkN6sFB9OUPtj3OK6ckg

クライアントシークレット vAmMEUxsM1wFyO993EEWKU6fyFUj088u

アプリケーション名* DemoServiceSite

Clientの種類 Confidential Public

〜〜中略〜〜

グループID defaultGroup

スコープ* openid * profile *

ID連携用にこのクライアントに以下のスコープ（権限）を設定
openid : クライアントがOpenID ConnectによるID連携を可能 ※必須
profile, email, address, phone ※アクセスを許可する範囲に応じて設定（独自定義の追加も可）

STEP3 : サービスサイトと認証サーバでID連携を行う

■ mod_auth_openidcの設定

```
# cat /etc/httpd/conf.d/openidc.conf
OIDCProviderMetadataURL https://<認証サーバURL>/oauth/v2/.well-known/openid-configuration
OIDCClientID <払い出されたクライアントID>
OIDCClientSecret <払い出されたクライアントシークレット>
OIDCRedirectURI https://www.example.com:443/callback
OIDCScope "openid profile"
<Location />
  AuthType openid-connect
  Require valid-user
</Location>
```

■ サービスサイトコンテンツでユーザ情報連携を組み込み

```
# vi /var/www/html/member/top.php
<TITLE>Demo</TITLE>
</HEAD>
<BODY>
<h3>ようこそ、
    <?php print $_SERVER['OIDC_CLAIM_family_name']; ?> さん
</h3>
<h4>【会員メニュー】 </h4>
. . .
```

STEP3 : サービスサイトと認証サーバでID連携を行う

mod_auth_openidcが
認証済みかチェック

未認証でサービスサイトに
アクセスすると

mod_auth_openidcが
ユーザ情報を連携

認証画面が表示され
認証を行うと

サービスサイトに遷移し、
サイト側でヘッダーに格納され
たユーザ情報を受け取るこ
とができます

API連携認証システム として構成してみる！

API連携認証システムの構築イメージ

実現するために必要な対応

■ ThemiStruct Identity Platform側の設定

- ① APIをリソースとして登録
 - ② スcope (※) の定義
 - ③ API利用アプリをクライアントとして登録
- ※ トークンに付与される権限表す

■ オープンAPIでの実装

- ① トークンを受け取る
- ② トークン情報を取得する
- ③ 取得したトークン情報を検証する
- ④ APIの処理/応答を定義する

■ API利用アプリでの実装

- ① 認可要求を行う
- ② トークンを取得する
- ③ トークンを付与してAPIを呼び出す

許可されたアクセス条件の確認の実現

オープンAPIの公開に必要な設定 & ThemisStruct Identity Platform が提供するAPI

■ APIをリソースとして登録

リソース

作成

リソース名* demoResource

グループID defaultGroup

説明 Description

キャンセル 作成

リソース

編集 削除

登録後払い出されたIDとシークレットキー

編集

リソースID FijiS5f9p3WpNyzGE9zNoFnmKb5p0R9I

リソースシークレット InrMi79WYX7YxNhZWsyLhXtRvRkU3vk9

リソース名* demoResource

■ ThemisStruct Identity Platform

① APIをリソースとして登録

■ オープンAPI

① トークンを受け取る

② トークン情報を取得する

■ トークンの詳細情報取得のエンドポイントを提供

種別	エンドポイント	説明
トークン情報取得	/oauth/v2/token/introspect	提示したトークンの詳細情報を取得します

許可されたアクセス条件の確認の実現

■ スコープ（アクセス権限）の登録

スコープ

作成

スコープID* point_read

グループID defaultGroup

アクセストークンの有効期限(秒)* 3600

リフレッシュトークンの有効期限(秒)* 1209600

リフレッシュトークンの発行有無* 利用する 利用しない

認証状態との紐付け有無* 利用する 利用しない

連携するクレーム At least select 1 Attribute.

説明 Description

独自のスコープを定義することが可能。

■ ThemiStruct Identity Platform側の設定

- ① APIをリソースとして登録
 - ② **スコープ (※) の定義**
 - ③ API利用アプリをクライアントとして登録
- ※ トークンに付与される権限表す

許可されたアクセス条件の確認の実現

オープンAPI側の実装

```
/* API(1) ↓
 * ポイント照会↓
 */↓
router.get('/point', function(req, res, next) {↓
↓
  var authzHeader = req.get('Authorization');↓ アクセストークン受け取り
  if(!authzHeader) {↓ (Authorizationヘッダー)
    . . . ↓
  }↓
  intro.tokenintro(req, clientId, clientSecret).then((ret) => {↓
↓
    var info = ret;↓
    if(info.code == 200) {↓
      if(checkActive(info.active) &&↓
        checkScope(info.scope, GETPOINT_SCOPE) &&↓
        checkIss(info.iss, GETPOINT_ISS) ) {↓
        resource.getpoint(info.sub).then((rows) => {↓
          res.header('Content-Type', 'application/json; charset=utf-8');↓
          res.send(rows[0]);↓
        }).catch((err) => {↓
          badRequest(res);↓
          return;↓
        });↓
      } else {↓
        invalidRequest(res);↓
        return;↓
      }
    }
  }
}
```

アクセストークンの情報取得

アクセストークンの検証

■オープンAPIでの実装

- ① トークンを受け取る
- ② トークン情報を取得する
- ③ 取得したトークン情報を検証する
- ④ APIの処理/応答を定義する

■アクセストークン情報

項目名	格納値
active	トークンの有効性
client_id	トークンを要求したクライアント
scope	トークンが許可するアクセス範囲を表すScopeのリスト
sub	ユーザの識別子
iss	トークン発行元
exp	トークンの有効期限
iat	トークンの発行日時

※上記はJSON形式で返却される

APIアクセスのための認証要求の実現

ThemiStruct Identity Platformで提供されるエンドポイント & 導入に必要な設定

■ 認証要求を許可 (トークンを発行) するクライアントを登録

クライアント

作成

アプリケーション名* DemoAPP

Clientの種類 Confidential Public

リダイレクトURI*

Redirect URI

https://tsdev.themistrustruct.com/cb

リフレッシュトークン機能の利用* 利用する 利用しない

利用許可するグラントタイプ Authorization Code

設定項目例

スコープ (許容するアクセス範囲)

リフレッシュトークン有無

発行されたトークンの有効期限

■ ThemiStruct Identity Platform

③ API利用アプリをクライアントとして登録

■ API利用アプリでの実装

- ① 認可要求を行う
- ② トークンを取得する

■ 認証要求及びトークン取得のエンドポイントを提供

No	エンドポイント	説明
APIアクセスのための認証要求	/oauth/v2/authz	リクエストパラメータを基に必要な認証の要求を行います。未認証の場合は認証画面を表示します。認証成功すると登録されたコールバック先に認可コードやトークンを送信する。
	/oauth/v2/token	認証成功後に受け取った認可コード等からアクセストークンを取得します。

APIアクセスのための認証要求の実現

API利用アプリ側の実装

■ 認可要求の実装例

```
@RequestMapping("/xxxxx")  
public String xxxxx(Model model, Principal principal,  
    HttpServletRequest request) throws ServletException {  
  
    ^ AuthorizationCodeRequestUrl codeUrl  
    ^     = new AuthorizationCodeRequestUrl(AUTHORIZATION_SERVER_URL, CLIENT_ID);  
    ^ codeUrl.setScopes(Arrays.asList(SCOPE_REF));  
    ^ codeUrl.setResponseType(Arrays.asList("code"));  
    ^ codeUrl.setRedirectUri(REDIRECT_URL);  
    ^ codeUrl.setState(createState(request));  
    ^ codeUrl.set(NONCE, createNonce());  
  
    ^ return "redirect:" + codeUrl.build();  
}  
}
```

■ 認可コードからアクセストークンを取得する実装例

```
private static Credential requestAccessToken(String code) throws IOException {  
    ^ TokenResponse response = null;  
    ^ try {  
    ^     response = new AuthorizationCodeTokenRequest(new NetHttpTransport(), new JacksonFactory(),  
    ^         ^ new GenericUrl(TOKEN_SERVER_URL), code)  
    ^         ^ .setRedirectUri(REDIRECT_URL)  
    ^         ^ .setClientAuthentication(new BasicAuthentication(CLIENT_ID, CLIENT_SECRET))  
    ^         ^ .execute();  
    ^     }  
    ^ }  
}
```

Google OAuth2 API Client Library for Java利用時の実装例

■ 認証成功後のコールバック実装例

```
public static Credential oauth2Callback(HttpServletRequest request) {  
    ^     StringBuffer fullUrlBuf = request.getRequestURL();  
    ^     . . .  
    ^     AuthorizationCodeResponseUrl authResponse  
    ^     ^     = new AuthorizationCodeResponseUrl(fullUrlBuf.toString());  
    ^     . . .  
    ^     // 認可コードを取得  
    ^     code = authResponse.getCode();  
    ^ }  
    ^ . . .  
    ^ try {  
    ^     // 認可コードからアクセストークン取得  
    ^     cre = requestAccessToken(code);  
    ^     . . .  
    ^ }
```

■ API利用アプリでの実装

- ① 認可要求を行う
- ② トークンを取得する
- ③ トークンを付与してAPIを呼び出す

本ご紹介のまとめ

本ご紹介のまとめ

□ ThemisStruct Identity Platformのセットアップ

- AWSマネージドサービスを利用したシステムを4つのステップで構築
- セットアップはCloudFormationやインストーラーを利用して自動化
- 上記により高可用・スケーラブルな環境を約2時間で構築可能

□ 共通ID基盤として構成する

- サービスサイトへの認証機能の組み込みを簡略化するためのSDKを提供
- アカウント情報の連携を可能とするアカウント管理APIを提供
- ID情報連携をOpenID Connectや対応ミドルウェア等を利用して実現することが可能

□ API連携認証システムとして構成する

- OAuth2.0に準拠した認証要求やアクセス条件確認を行うためのエンドポイントを提供
- 各種エンドポイントを利用するためのクライアントやスコープ設定を管理GUIから設定可能
- オープンAPIの公開でAPI側はトークンの受け取り／情報取得／検証の3ステップを組み込み

ご清聴ありがとうございました

ThemisStruct
テミストラクト

【お問い合わせ先】

株式会社オージス総研

TEL: 03-6712-1201 / 06-6871-8054

mail: info@ogis-ri.co.jp