Agenda

【パイメゾン 真弓様 二回目レッスンの実施内容】

・目標, 現状再確認

・C# アプリケーションデモ（JWT認証あり）

・JWT認証説明

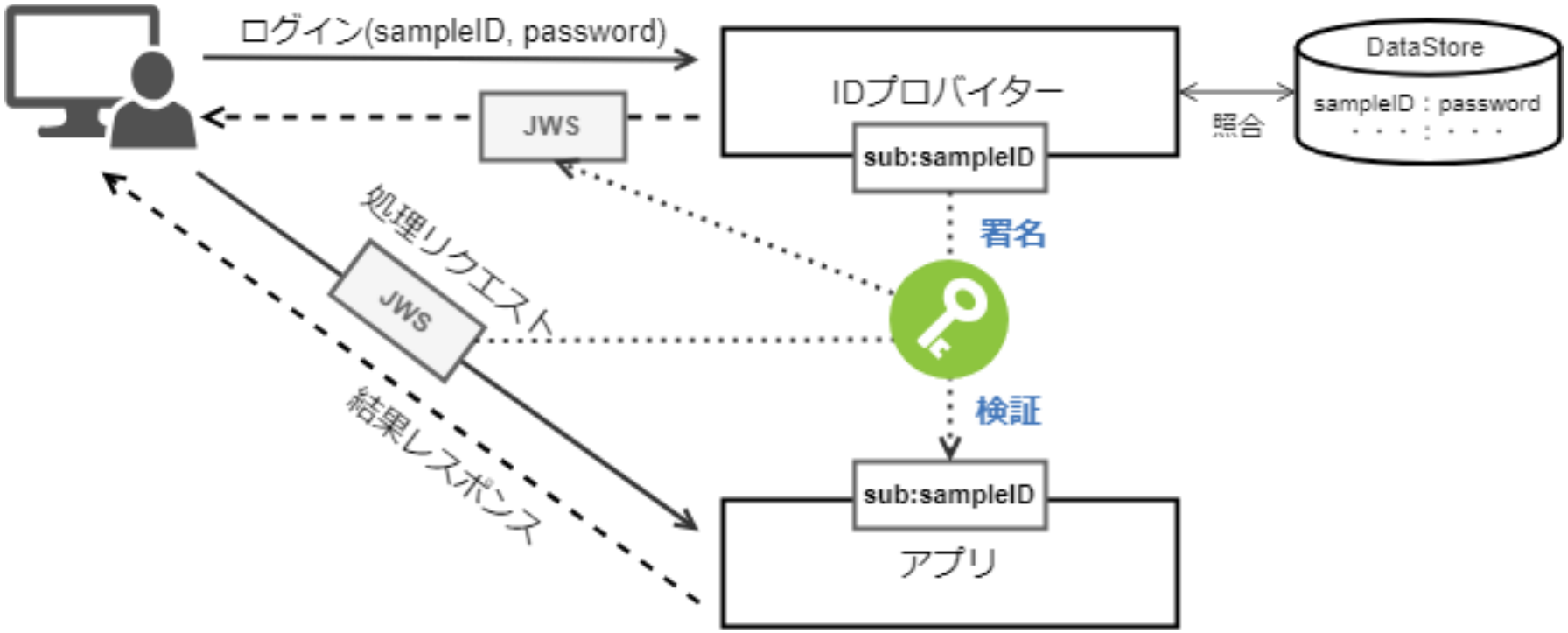
・C# コード説明

・テスト環境構築

・次回レッスン予定

12/5(火) 20:00-

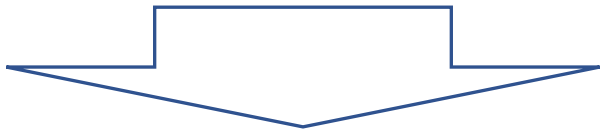
認証フロー


ログイン(sampleID, password)
JWS
IDプロバイター
sub:sampleID
照合
DataStore
sampleID : password
署名
処理リクエスト
JWS
結果レスポンス
検証
sub:sampleID
アプリ

JWT 認証

JWT(JSON WEB Token)[ジョット]

JWTの定義(RFC) ：”HTTPヘッダーやクエリパラメータ等サイズに制約がある環境で使うことを前提に、
JSON形式のデータをURLセーフでコンパクトな型式にしたもの”



・JSONデータをURLセーフにする方法を規定　→ JSONデータをBASE64URLエンコードする
・JSONデータをコンパクトにする方法を規定　→ よく使われるデータ項目の名称を省略形にする

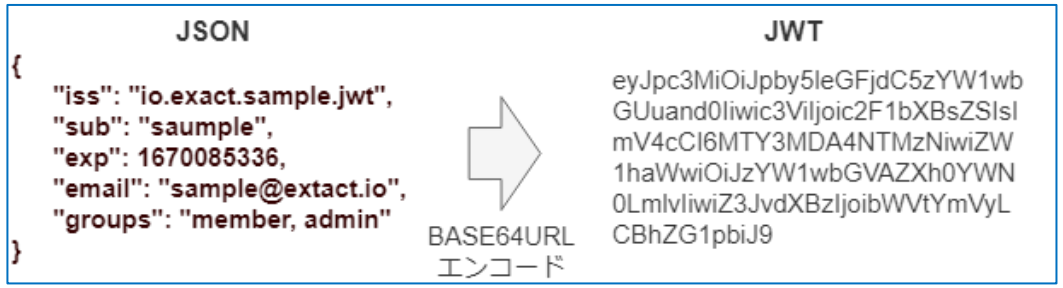
省略名	項目名	説明
sub	subject	ユーザー識別子などJWTの主体
iat	Issued At	JWTの発行日時
nbf	Not Before	JWTの有効開始日時
jti	JWT ID	JWTの一意な識別子
exp	expiration time	JWTの有効期限

JWTデコーダ
https://jwt.io/#debugger-io

Unixタイムスタンプ → 日付
https://www.unixtimestamp.com/

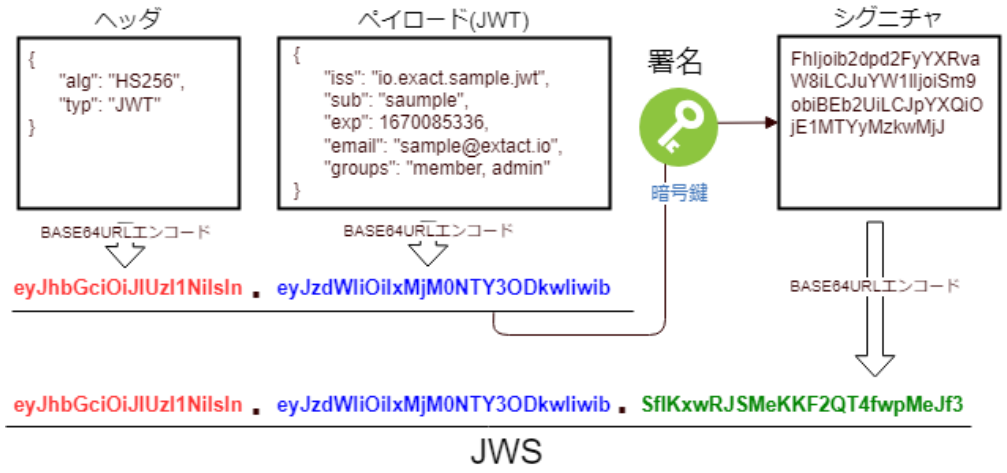
JWT 認証

JWTとは
・JSONをBASE64URLエンコードしたもの
・よく利用されるキー名を省略名で予約登録しているもの
（JWTの本来の定義には認証や暗号化に関して規定していない）



JWS

JWS(JSON Web Signature)
・改竄検知のためのシグニチャを追加したもの：HTTPのヘッダに入れて送信される。



REST API

HTTPを用いたアプリケーションインターフェース

【特徴】

・クライアント／サーバー

・ステートレス：前後の状態は無関係

・リソースの識別：URIでデータを識別する

・データ：現在ではほぼJSONが用いられている

HTTPプロトコル

・TCPパケットのデータ部にHTTPプロトコルパケットが入る

・デフォルトのポート番号は 80番

・HTTPを暗号化したプロトコルがHTTPS(SSL):443番
→ 認証局の証明書が必要(Verisign等)

HTTPプロトコル

・HTTPは原則的に『ステートレス』なプロトコル

『ステートレス』とは**”今回のパケット”**が**”前回のパケット”**の影響を受けていないし、**“次回のパケット”**にも影響を与えない。
→ 1つ１つのパケットで完結している。

・そのため、WEBサイトでステート管理（状態管理）を
実現するには クッキーやセッション等の追加の仕組みが
必要となる
（ex. ユーザー認証情報等）

ステート：状態

HTTPプロトコル：コマンド

要求

応答

コマンド, URL, プロトコル, バージョン, データ
(GET/ www.yahoo.co.jp / HTTP / 1.0)

・肯定応答　　：200番台
・リダイレクト　　：300番台
・リクエスト内容エラー：400番台 404 Not Found
・サーバーエラー　　：500番台 500 Intenal Server Error

・GET
・POST
・PUT
・DELETE

REST API

データ取得，操作：CRUD

	HTTP コマンド	HTTP 送信データ	DB操作
Create	POST	JSON	insert
Read	GET	ー	select
Update	PUT	JSON	update
Delete	DELETE	ー	delete

環境構築

GitHubからC#サンプルコードとDocker環境ダウンロード

C# サンプルコード

```
git clone https://github.com/TakSakamoto-Osaka/REST_CSharp
```

→ Visual Studio でビルドできるか確認

Docker環境

```
git clone https://github.com/TakSakamoto-Osaka/REST_API
```

Docker環境構築 1

・Dockerデスクトップを起動する

・PowerShell で ダウンロードした REST_APIフォルダに移動する

・WEBサーバーイメージ生成するため下記コマンド実行

```
docker build –t api-demo .
```

・dockerコンテナ起動するため下記コマンド実行

```
docker-compose up -d
```

Docker環境構築 2

・データベース生成するため下記コマンド実行

```
docker exec –it api-db bash
```

```
mysql –u root -psamurai
```

```
create database api_demo
```

```
exit
```

```
exit
```

Docker環境構築 2

・データベース展開

```
docker cp api_demo.sql:api-db:/root/
```

```
docker exec –it api-db bash
```

```
cd
```

```
mysql –u root –psamurai api_demo < api_demo.sql
```

```
exit
```

Docker環境構築 3

・docker コンテナ再起動

```
docker-compose down
```

```
docker-compose up -d
```