



OAuth2 and REST Services

Mike Peat
Unicorn InterGlobal



Summary

The OAuth2 mechanism

REST APIs

HTTP Verbs

The OAuth Dance

Building an application: Google Drive

- Create a new Web App

- Sign up - Register application

- Get Credentials & use to sign in

Test our access: make a call to the API

JSON

Using RESTGen to build & use structs

Using data from the API in our Web App

What is OAuth2?

A mechanism for allowing applications access to user resources held in third-party services such as:

- Google
- Microsoft Office 365
- Facebook
- And many, many more

Without the user having to share their credentials with the applications, while limiting the “**scope**” and duration of that access

REST APIs

- OAuth2 is generally used to access REST APIs
- **REST: RE**presentational **S**tate **T**ransfer
- An HTTP mechanism where each operation is represented by a specific combination of **URI** and **HTTP verb**
- Data interchange is usually via **JSON** (Java**S**cript **O**bject **N**otation)

HTTP Verbs

REST APIs generally use a wider set of **verbs** than the familiar GET and POST, providing the equivalent of database CRUD operations:

- **POST** **C**reates (or **U**pdates) data
- **GET** **R**eads data
- **PATCH** **U**pdates data
- **DELETE** **D**eletes data (Well... Doh!)
- **PUT** also **C**reates data

(PUT is less commonly used than the others - POST is the usual "create" verb)

Getting Started

- The first step is to register your application with the service provider in question
- Usually involves first signing up as a developer for that provider – free in most cases (not MS)
- You provide a **name** for the app and a “**redirect URL**” (and usually more)
- On registering, your app you will be given a “**client ID**” and a “**client secret**” (or “**key**”) for it to use to identify itself to the provider

The DataFlex OAuth2 Component

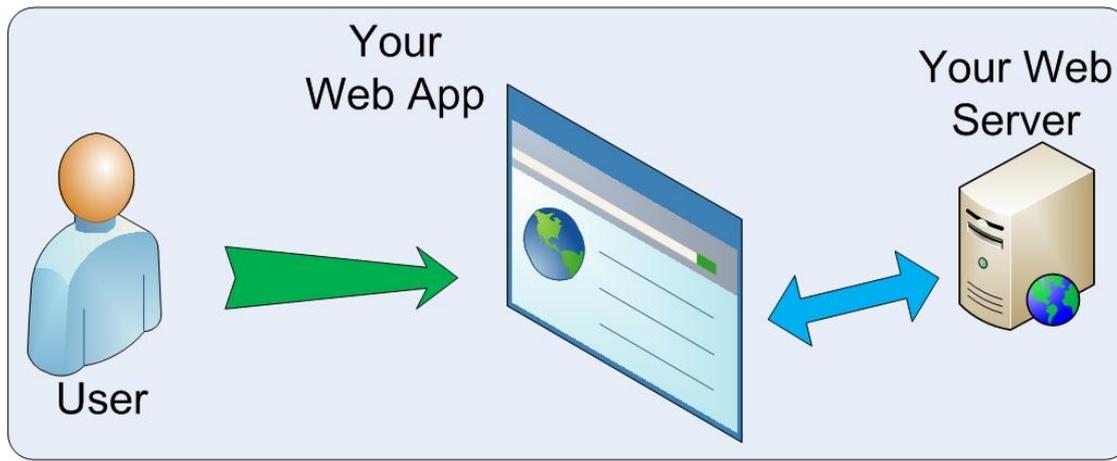
- Your application should use the OAuth2 DataFlex library
- Copy the OAuth2.js and use it in your index.html
- Drag an OAuth2 component into a WebView
- Set its properties (there are many, but only a few need be set in most cases)
- Define pre- and post-login behaviours, as well as those for login failure
- Have a mechanism for triggering the login process - Send Login of the OAuth2 object

Using OAuth2

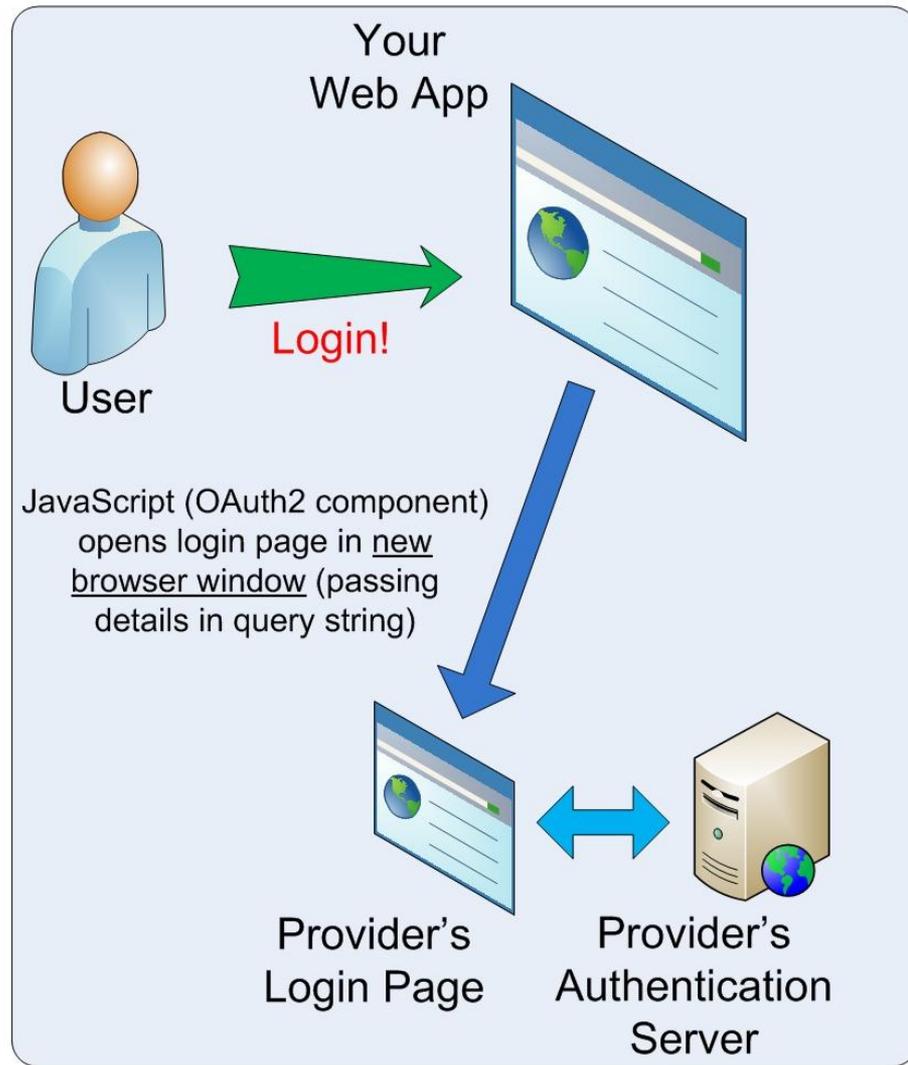
- The OAuth2 authentication and authorisation process is somewhat complicated (although less so than its predecessor, OAuth1)
- It involves multiple actors and interactions in what is sometimes called...

The OAuth Dance

The OAuth Dance Step 1



The OAuth Dance Step 2

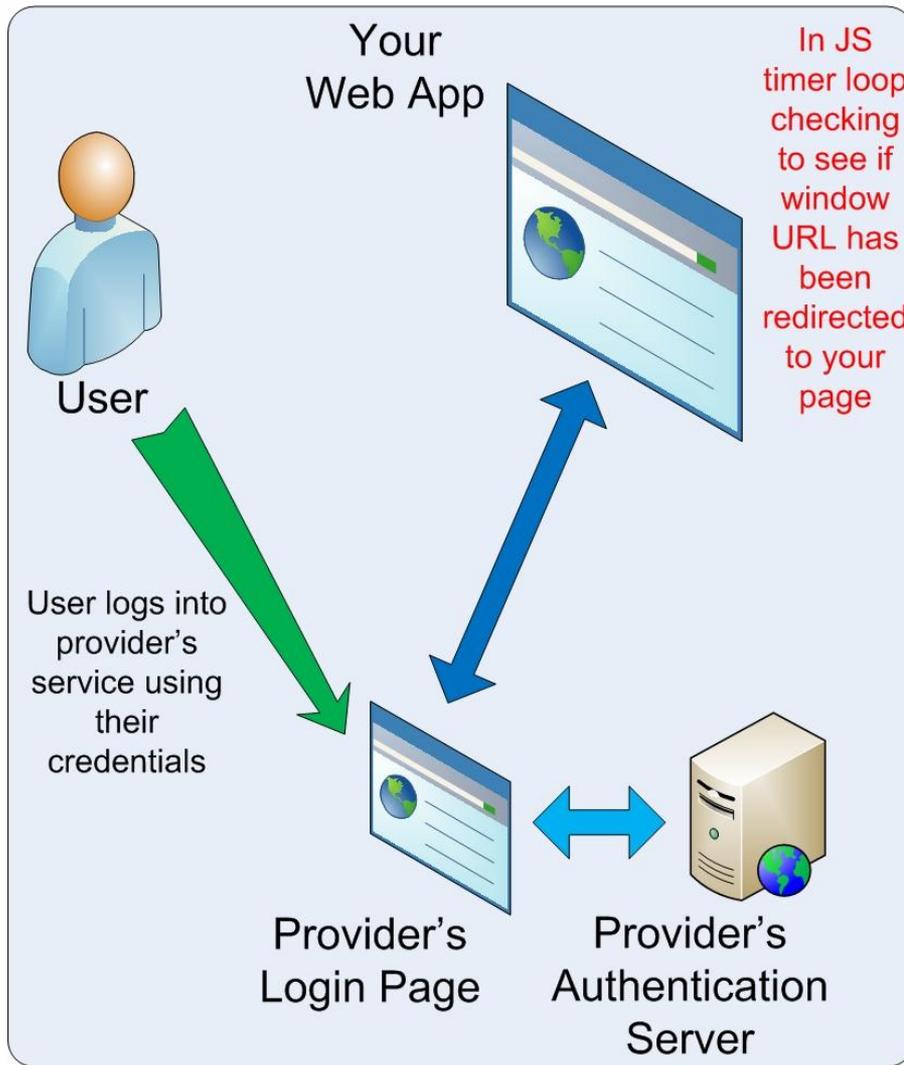


Your Web Server



Provider's REST API Server

The OAuth Dance Step 3

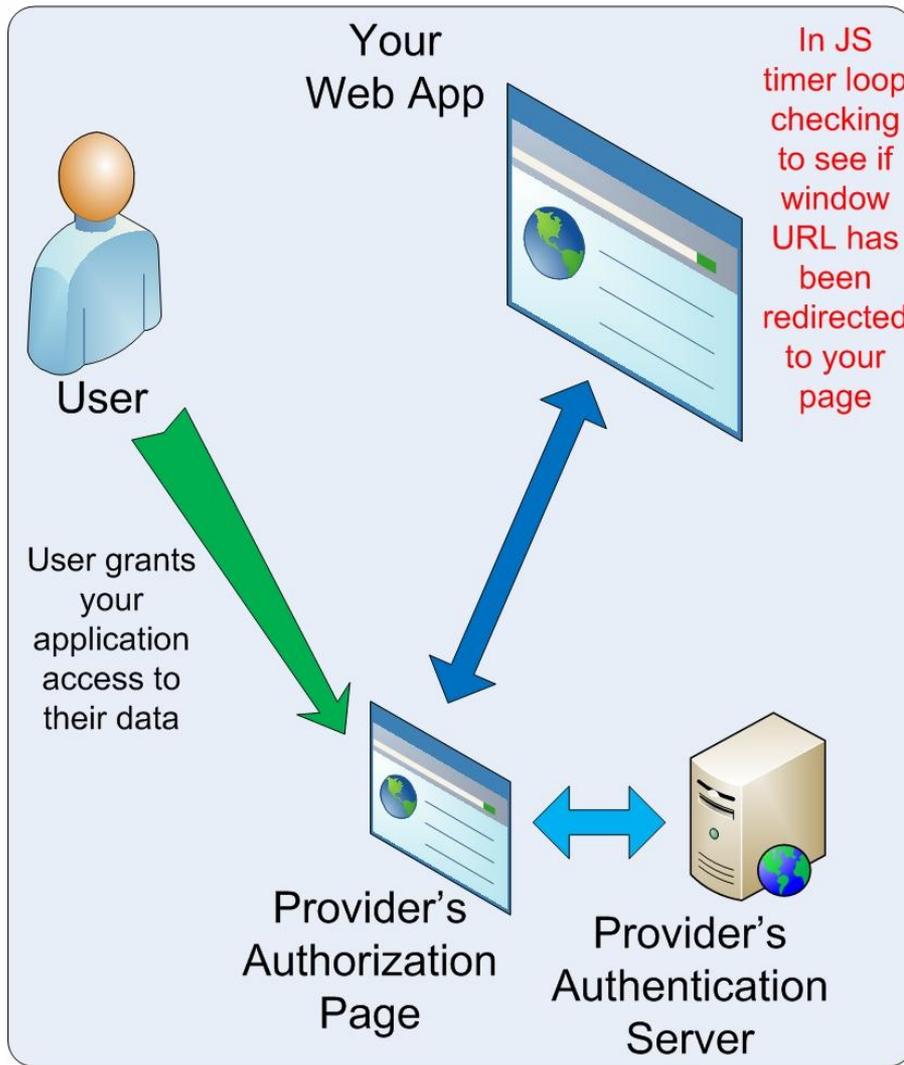


Your Web Server



Provider's REST API Server

The OAuth Dance Step 4

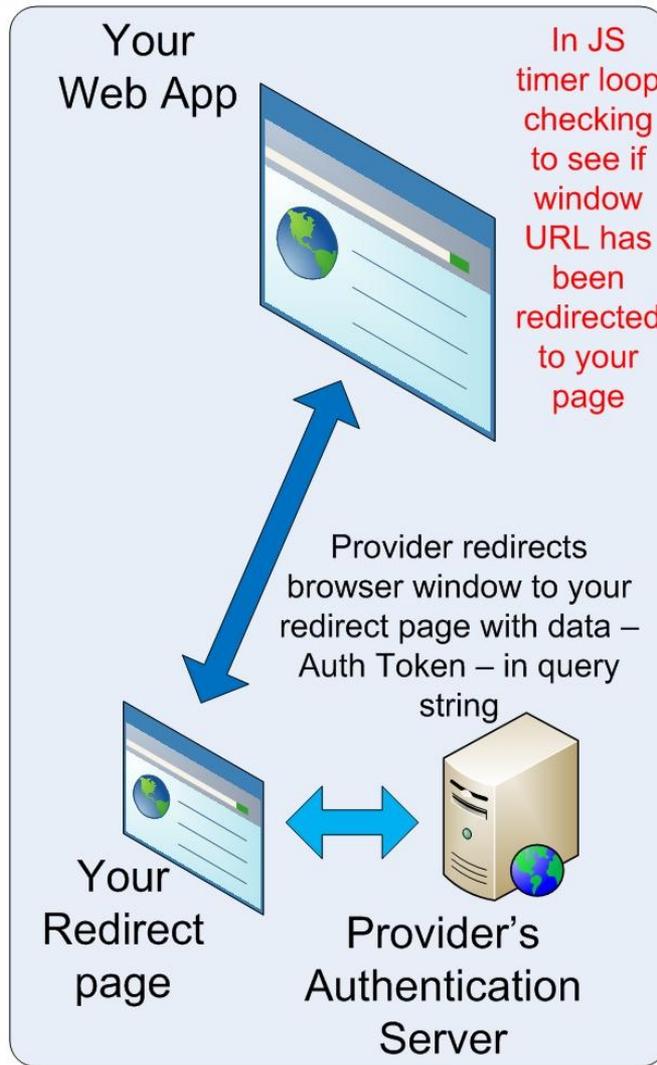


Your Web Server

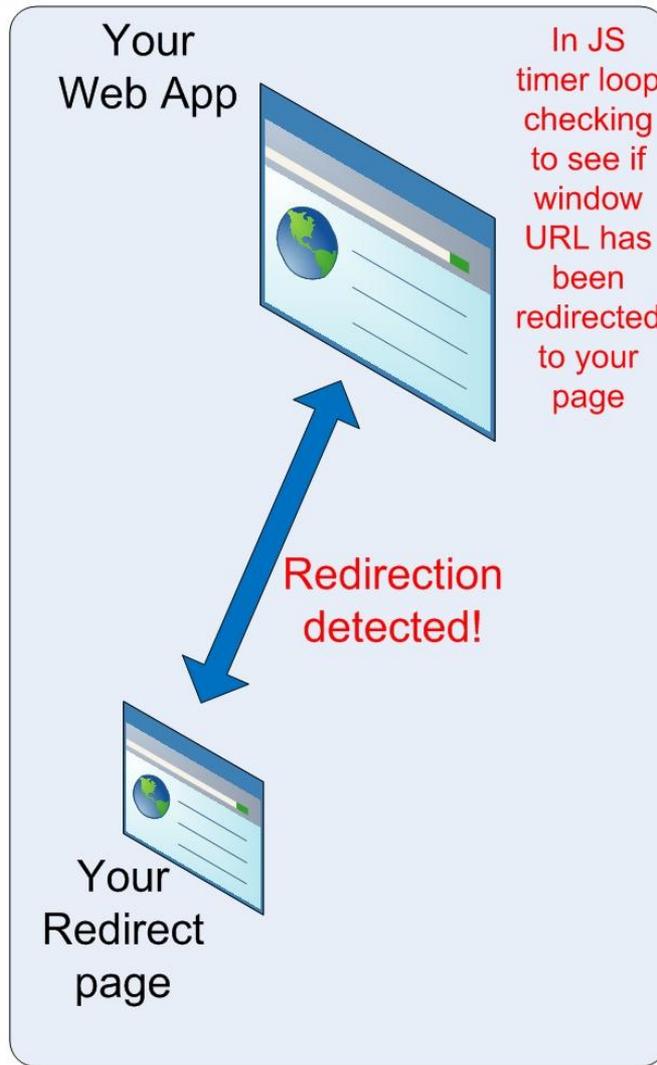


Provider's REST API Server

The OAuth Dance Step 5



The OAuth Dance Step 6

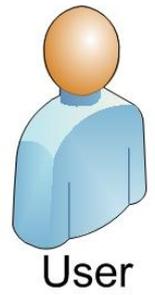


Your Web Server



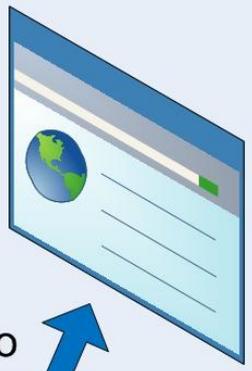
Provider's REST API Server

The OAuth Dance Step 7



User

Your Web App



Capture URL info (including Auth Token) then close browser window



Your Redirect page

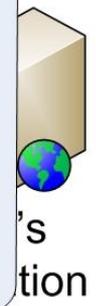


Server

Your Web Server

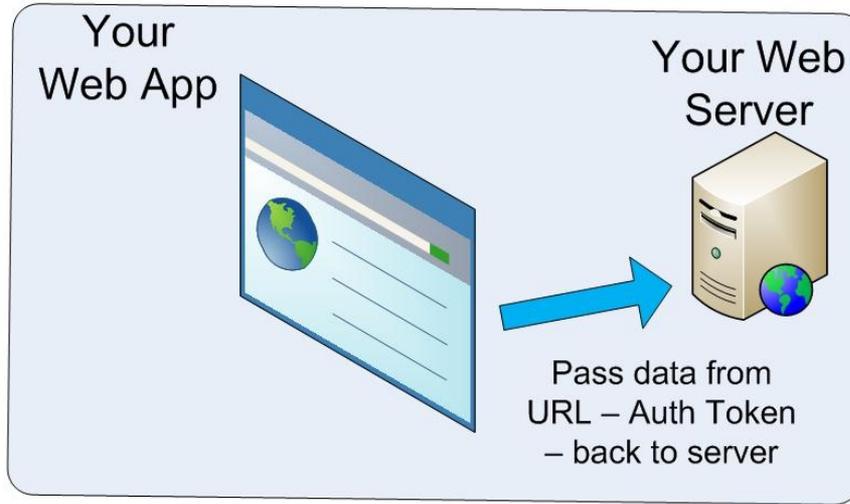


Provider's REST API Server



's tion

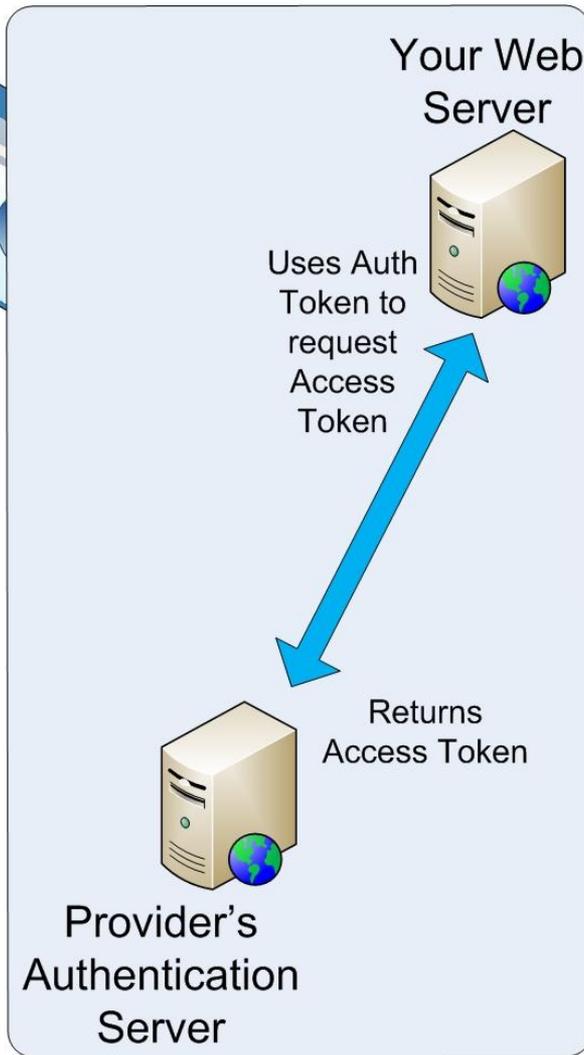
The OAuth Dance Step 8



The OAuth Dance Step 9



Your Web App



Your Web Server



Uses Auth Token to request Access Token



Returns Access Token

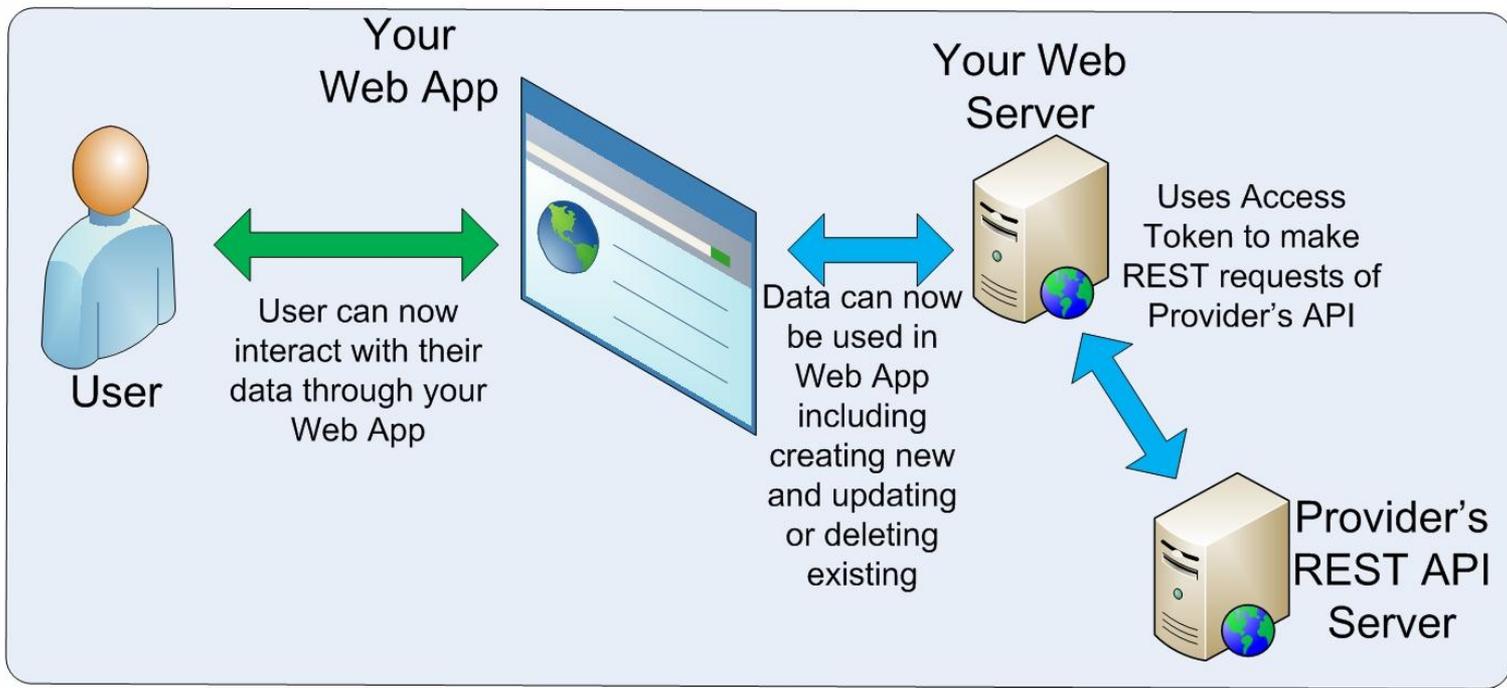


Provider's Authentication Server

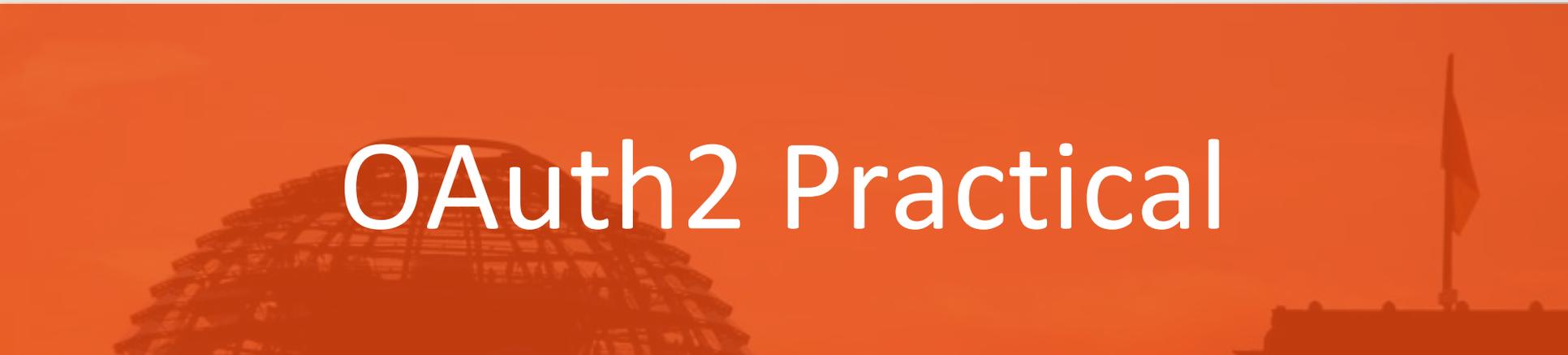


Provider's REST API Server

The OAuth Dance Step 10



Provider's
Authentication
Server

The background of the slide features a solid orange horizontal band. Within this band, there is a faint, semi-transparent image of a large, geodesic dome structure on the left and a flag on a tall pole on the right. The text 'OAuth2 Practical' is centered in white, bold font across the middle of the orange band.

OAuth2 Practical

Next Steps

So now we have a big pile of returned JSON

What are we going to do with it?

JSON

Lightweight data exchange format (see [JSON.org](https://www.json.org/))

Used by DataFlex web framework

Also used by most REST APIs

Only six data types:

- **String:** "Abcd"
- **Number:** 22 or 123.456 or 3.1415e2
- **Boolean:** true or false
- **Null:** Null
- **Object:** {"name": val, "name": val, "name": val}
- **Array:** [val, val, val]

JSON

Most service providers give sample JSON

If not, you can always just make a call and examine the result

How do we get this mass of data into a form we can work with?

JSON

When I started, I manually created the required structs

Then I used Sture Andersen's VDFXray program to generate packages which had functions for translating JSON data into the structs and from the structs into JSON data

These rest on top of Sture's JsonFunctions package - part of his StureApsPublicLib library

JSON

Manually creating those structs is tedious, time consuming and error-prone

So I wrote a program to analyse the JSON (again, making use of Sture's `JsonFunctions.pkg`) and write packages which contained both the structs and the handler code for them

RESTGen

The background of the slide features a solid orange horizontal band. Within this band, there is a faint, semi-transparent image of a large, geodesic dome structure on the left and a building with a flag on the right. The text 'RESTGen Practical' is centered in white, bold, sans-serif font over the orange band.

RESTGen Practical

The background of the slide features a horizontal orange band. Within this band, on the left, is a semi-transparent image of a large, dome-shaped architectural structure with a grid-like pattern. On the right side of the band, there is a silhouette of a flagpole with a flag flying, set against a lighter background.

Practical - Finish

A horizontal orange band spans the width of the page. In the background, there is a faint, semi-transparent image of a building with a curved, grid-like facade on the left and a flag on a tall pole on the right. The word "Questions?" is written in white, sans-serif font across the center of the orange band.

Questions?