This minimal app uses GitHub for authentication,

benefitting from the autoconfig features in Spring Boot.

Configure your app to use GitHub as the authentication provider.

\* <<github-register-application,Add a New GitHub app>>

\* <<github-application-config,Configure application.yml>>

\* <<github-boot-application,Boot up the application>>

[[github-register-application]]

=== Add a New GitHub App

To use GitHub's OAuth 2.0 authentication system for login,

you must first https://github.com/settings/developers[Add a new GitHub app].

Select "New OAuth App" and then the "Register a new OAuth application" page

is presented.

Enter an app name and description.

Then, enter your app's home page, which in this case is: http://localhost:8080

Indicate the Authorization callback URL as

`http://localhost:8080/login/oauth2/code/github`

Click \_Register Application\_.

The OAuth redirect URI is the path in the application that the end-user's

user-agent is redirected back to after they have authenticated with GitHub

and have granted access to the application on the \_Authorize application\_ page.

TIP: The default redirect URI template is `{baseUrl}/login/oauth2/code/{registrationId}`.

The \*\_registrationId\_\* is a unique identifier for the `ClientRegistration`.

[[github-application-config]]

=== Configure `application.yml`

To make the link to GitHub, add the following to your `application.yml`:

.application.yml

[source,yaml]

----

spring:

security:

oauth2:

client:

registration:

github:

clientId: github-client-id

clientSecret: github-client-secret

# ...

----

Simply use the OAuth 2.0 credentials you just created with GitHub, replacing

`github-client-id` with the client id and `github-client-secret` with the client

secret.

[[github-boot-application]]

==== Boot Up the Application

With that change, you can run your app again and visit the home page at

http://localhost:8080.

Now, instead of the home page, you should be redirected to login with GitHub.

If you do that, and accept any authorizations you are asked to make,

you will be redirected back to the local app, and the home page will be visible.

If you stay logged in to GitHub, you won't have to re-authenticate with this

local app, even if you open it in a fresh browser with no cookies and no cached data.

(That's what Single Sign-On means.)

TIP: If you are working through this section with the sample application, be

sure to clear your browser cache of cookies and HTTP Basic credentials.

For a single server, open a new private window.

\*\*\*\*

It's safe to grant access to this sample since only the app running locally

can use the tokens and the scope it asks for is limited.

Be aware of what you are approving when you log into apps like this though:

They might ask for permission to do more than you are comfortable with

(e.g. they might ask for permission to change your personal data, which

is unlikely to be in your interest).

\*\*\*\*

== What Just Happened?

The app you just wrote, in OAuth 2.0 terms, is a \_Client Application\_, and it

uses the

https://tools.ietf.org/html/rfc6749#section-4[authorization code grant]

to obtain an access token from GitHub (the Authorization Server).

It then uses the access token to ask GitHub for some personal details

(only what you permitted it to do), including your login ID and your name.

In this phase, GitHub is acting as a Resource Server, decoding the token

that you send and checking if it gives the app permission to access the user's

details.

If that process is successful, the app inserts the user details into the

Spring Security context so that you are authenticated.

If you look in the browser tools (F12 on Chrome or Firefox) and follow the

network traffic for all the hops, you will see the redirects back and forth

with GitHub, and finally you'll land back on the home page with a new

`Set-Cookie` header.

This cookie (`JSESSIONID` by default) is a token for your authentication

details for Spring (or any servlet-based) applications.

So we have a secure application, in the sense that to see any content a

user has to authenticate with an external provider (GitHub).

We wouldn't want to use that for an internet banking website.

But for basic identification purposes, and to segregate content between

different users of your site, it's an excellent starting point.

That's why this kind of authentication is very popular these days.

In the next section, we are going to add some basic features to the application.

We'll also make it a bit more obvious to users what is going on when they get

that initial redirect to GitHub.