MAKE SCHOOL

More Authentication & Lab

BEW 1.2

Agenda

- Learning Outcomes
- Warm-Up
- Custom Validators
- **BREAK**
- Work Time: Authentication & Homework 4

Learning Outcomes

By the end of today, you should be able to...

1. **Define** Single Sign On & the OAuth protocol.
2. **Explain** why using Single Sign On can be beneficial to a web application.
3. **Use** the Requests-OAuthlib library to implement Single Sign On.

Warm-Up

Warm-Up (10 minutes)

In a group of 3, go over the **Books Lab (Auth)** and **Homework 4 (Auth)** assignments.

- How far did you get?
- What questions do you still have about authentication?

Custom Form Validators

Custom Validators

We can use custom validators to decide whether the data in a form is valid:

```
class MyForm(Form):
    name = StringField('Name', [InputRequired()])

    def validate_name(form, field):
        if len(field.data) > 50:
            raise ValidationError('Name must be less than 50 characters')
```

Activity - 10 min

With a group of 3, use the Validators Documentation to write code in the **Books (Forms)** or **Books (Auth)** Lab to:

1. Add custom validator to raise an error when creating a new book if the book title contains the word "banana".
2. Display the custom error message to the user when they fill out the form.

Stretch Challenge: If you finish early, try adding a `does_not_contain_word` validator that is reusable and takes in a custom word.

Then we'll go over the answer together.

Password Verification

Password Verification

In the Books (Auth) tutorial, we verified the user's password in the `/login` route. However, it's more correct to add validator methods. Let's try that together.

(Solution on next slide)

Password Verification

```
class LoginForm(FlaskForm):
    username = StringField('User Name',
        validators=[DataRequired(), Length(min=3, max=50)])
    password = PasswordField('Password', validators=[DataRequired()])
    submit = SubmitField('Log In')

    def validate_username(self, username):
        user = User.query.filter_by(username=self.username.data).first()
        if not user:
            raise ValidationError('No such user. Please try again.')

    def validate_password(self, password):
        user = User.query.filter_by(username=self.username.data).first()
        if user and not bcrypt.check_password_hash(user.password, password.data):
            raise ValidationError('Passwords didn\'t match. Please try again.')
```

Break - 10 min

Work Time

Use this time to work on Homework 4 (Auth).

Do not leave class until I check in with you 1-on-1!

Single Sign On & OAuth

What is OAuth?

MAKE SCHOOL

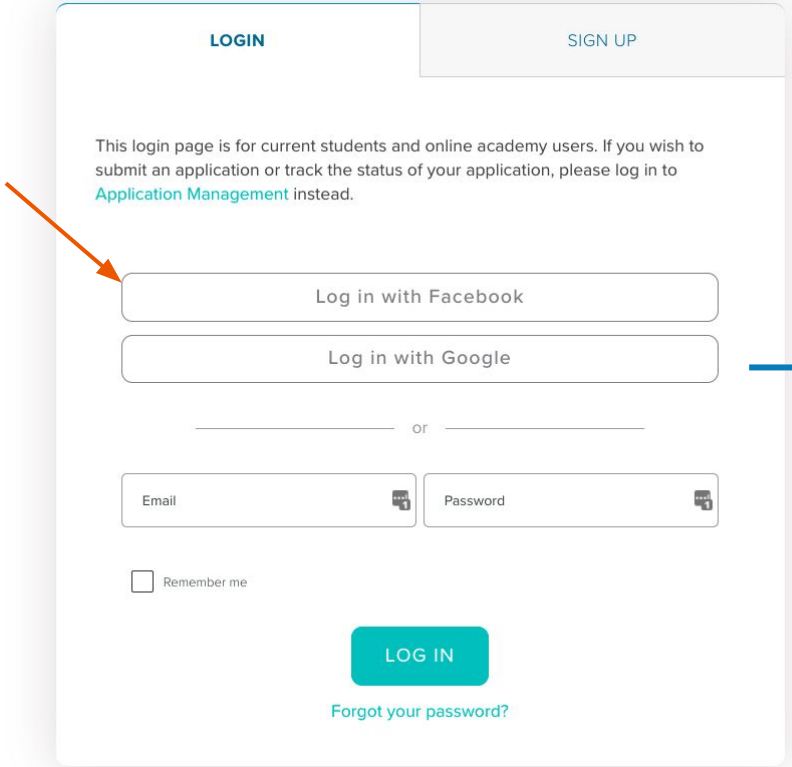
OK, so now we know how to let users provide a username and password to log into your site.

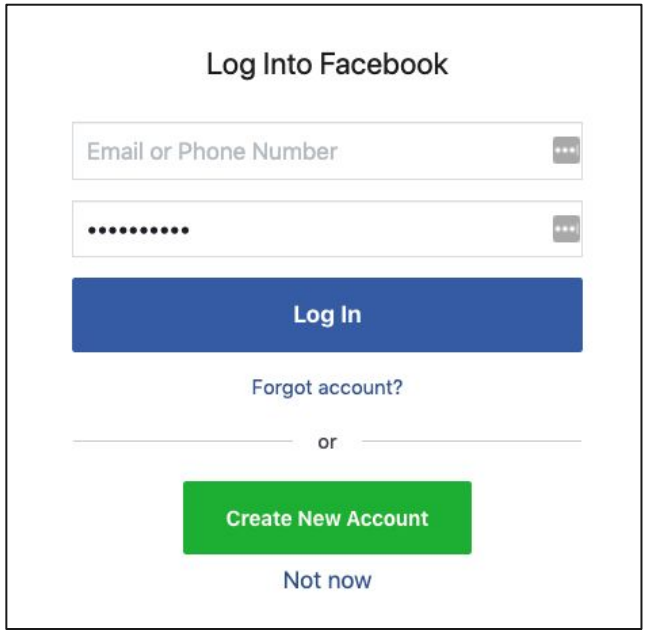
But, what if users don't want to remember a password??

We can let users sign in using another account that they already own, such as Facebook or Google, using a process called **Single Sign On**. We can implement this using the **OAuth** protocol.

What is OAuth?

Here is how **Single Sign On** looks on Make School's website.





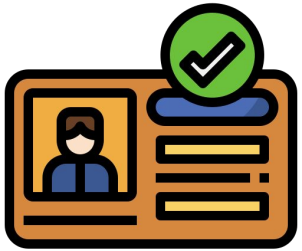
What is OAuth?

Single sign-on (SSO) is an authentication method that enables users to securely authenticate with **multiple applications** and websites by using just *one set of credentials*. Source

OAuth is an **authorization protocol** that allows one website (the **application**) to safely access a user's information on another website (the **API**, e.g. Facebook or Google) without ever accessing the user's *secret* information such as their password.

An Analogy

It's kind of like showing your driver's license in order to buy a drink at a bar.



API

Application

- The bar didn't issue your driver's license.
- They also don't need to know your private information (e.g. social security number).
- But because it comes from a trusted source (e.g. the state of California), they trust it to verify your identity & age.

Why use OAuth? (5 minutes)

Let's brainstorm some reasons why a website might want to use Single Sign On as an authentication method. In a group of 3, add some reasons below:

-
-
- Users are more likely to want to use your site
- Website doesn't have to store passwords or usernames
- If using OAuth, it does the authentication for you
- Convenience for the user
- Expose service to others via user's social graph
- Higher probability of logging in and using service for first time

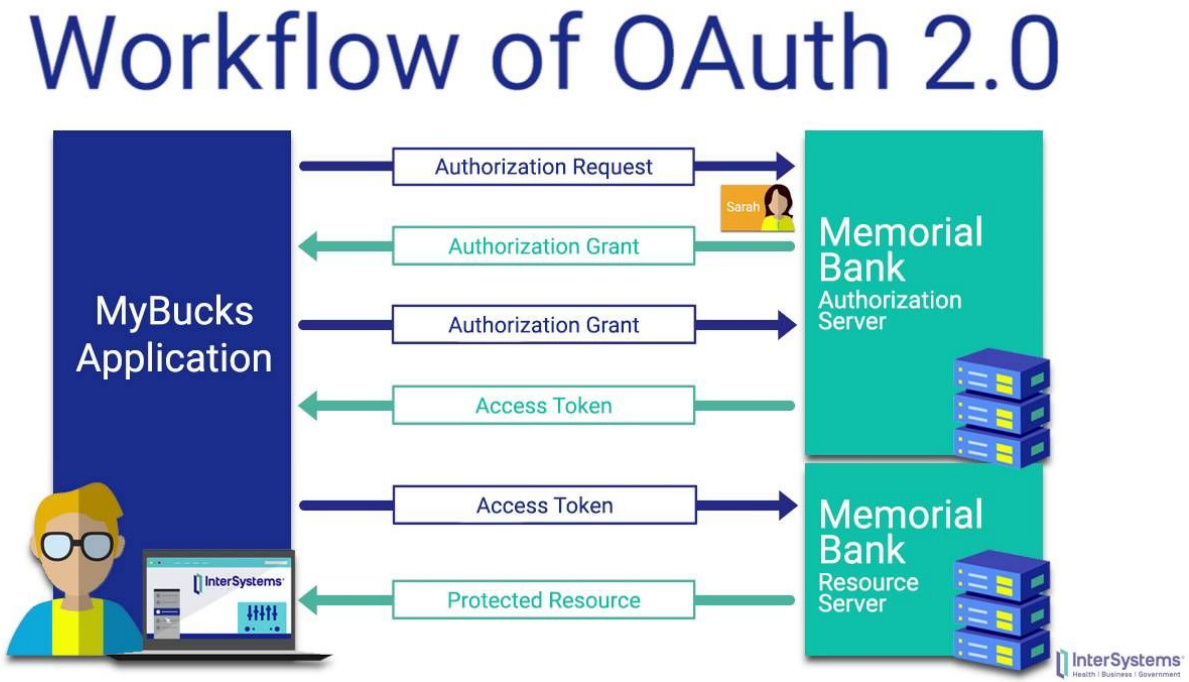
- It is easier
-
- One account for everything
-
- Convenience for user
- Use one account for everything

OAuth Tutorial (25 minutes)

With a partner, complete this OAuth tutorial to create a simple web application that utilizes Single Sign On with SimpleLogin and Facebook.

What is OAuth?

So, how does OAuth actually work? This video explains the process.



Why use OAuth? (5 minutes)

Let's brainstorm some reasons why a website might want to use Single Sign On as an authentication method. In a group of 3, add some reasons below:

- Hi!
- Removes the responsibility of encrypting passwords and keeping them safe-outsources that
-
- Speed up process of signing up
- One universal account to log in with immediately
- Removes barriers to desired call to action(ie;one less step to complete the purchase)

- Aren't responsible for storing usernames/pws in db
- Don't have to have a 'forgot my password' process
- Trust of the user (people often reuse passwords)
- Gives users multiple options for logging in
- You don't have to verify email
- Gives easy access to information about the user. So you don't have to store as much.