MAKE SCHOOL

More Forms

BEW 1.2

Agenda

- Learning Outcomes (5 minutes)
- Code Review (20 minutes)
- Form Errors (20 minutes)
- **BREAK**
- Lab time - work on homework 3!

Learning Outcomes

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

1. Use form errors to show the user any mistakes so that they can re-submit a form.
2. Review other form concepts.

Warm-Up: Code Review

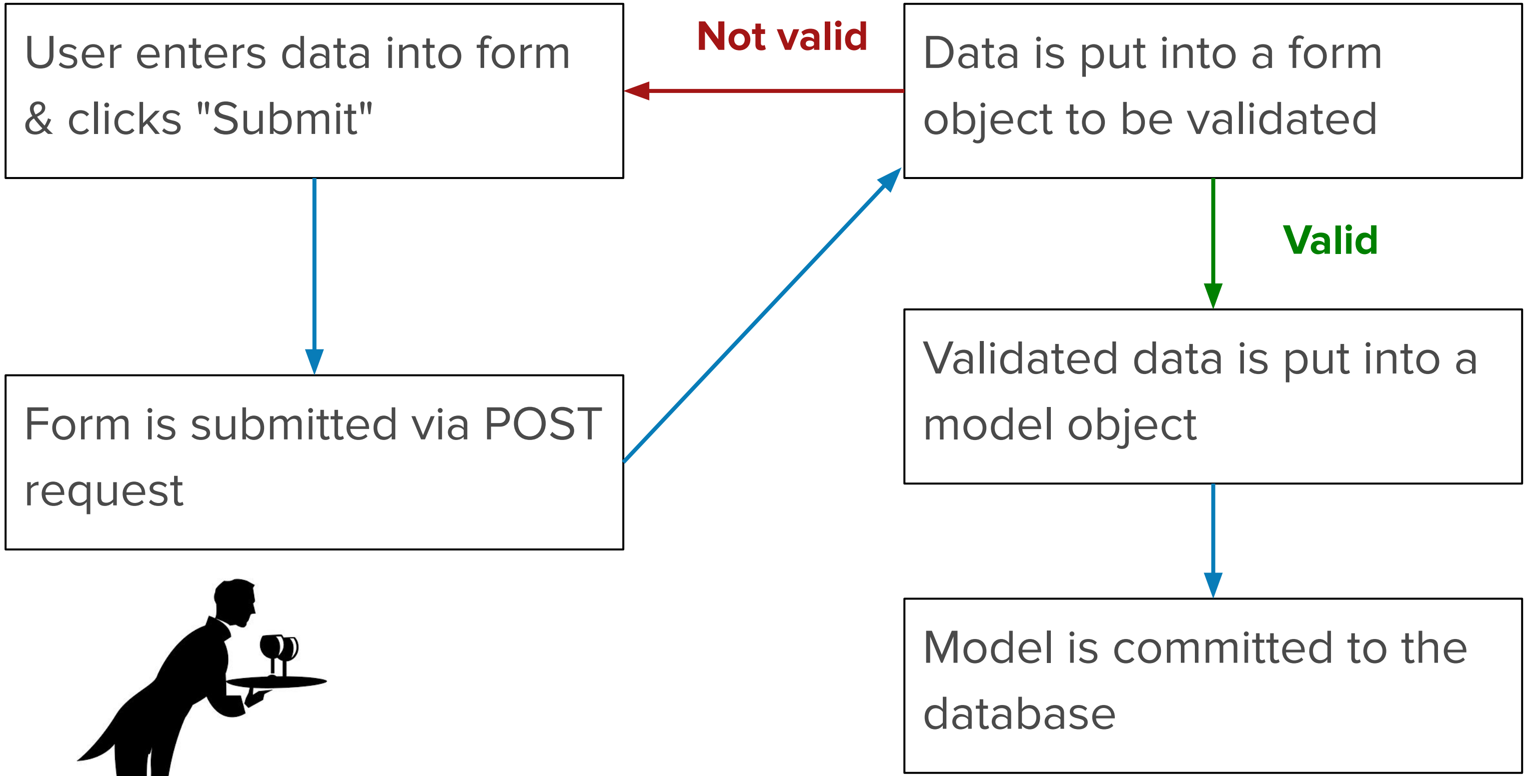
Code Review (6 minutes)

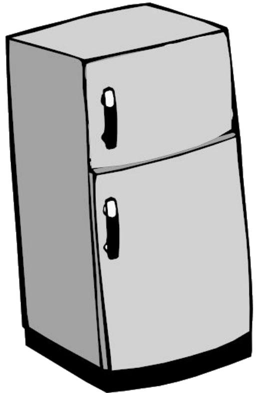
Form breakout rooms of 3.

With your group, discuss how far you got on the lab and any questions that came up.

Review: Forms Concepts

Flow of data


User enters data into form & clicks "Submit"
Form is submitted via POST request
Data is put into a form object to be validated
Not valid
Valid
Validated data is put into a model object
Model is committed to the database



Form Fields

Typically, a form class has one **field** per form input. There are different types of fields for different input types.

If the form is used to create a model, the fields will typically correspond to the fields in the model.

```
class Book(db.Model):
   id = db.Column(db.Integer, primary_key=True)
   title = db.Column(db.String(80),
nullable=False)
   publish_date = db.Column(db.Date)
   # ...
```

```
class BookForm(FlaskForm):
   """Form to create a book."""
   title = StringField('Book Title',
       validators=[DataRequired()])
   publish_date = DateField('Date Published')
```

Validators

A validator is used to impose some constraints on the form input values.

Some examples of validators are:

- DataRequired - cannot be empty
- Email - must be an email address
- Length
- URL
- RegExp

validate_on_submit

We can create a form object within a route, and it will automatically be populated with the user's responses (if any).

If the form was submitted, **and** all values were valid, then `validate_on_submit()` will return `True`.

```
@main.route('/create_book', methods=['GET', 'POST'])
def create_book():
    form = BookForm()
    if form.validate_on_submit():
        # ... do form processing here ...
```

csrf_token

The `csrf_token` is used to prevent **Cross-Site Request Forgery (CSRF)** attacks.

We can put it in the HTML form element and it will automatically be processed on submit.

```
<form method="POST" action="/submit">
   {{ form.csrf_token }}
   <!-- ... form inputs go here ... -->
</form>
```

Forms Lab (25 minutes)

With a partner, complete the Forms Lab activity by following the instructions for parts 1 and 2.

MAKE SCHOOL

Form Errors

Form Fields

Typically, a form class has one **field** per form input. There are different types of fields for different input types.

If the form is used to create a model, the fields will typically correspond to the fields in the model.

```
class Book(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80),
nullable=False)
    publish_date = db.Column(db.Date)
    # ...
```

```
class BookForm(FlaskForm):
    """Form to create a book."""
    title = StringField('Book Title',
        validators=[DataRequired()])
    publish_date = DateField('Date Published')
```

Validators

A validator is used to impose some constraints on the form input values.

Some examples of validators are:

- DataRequired - cannot be empty
- Email - must be an email address
- Length
- URL
- RegExp

validate_on_submit

We can create a form object within a route, and it will automatically be populated with the user's responses (if any).

If the form was submitted, **and** all values were valid, then `validate_on_submit()` will return `True`.

```
@main.route('/create_book', methods=['GET', 'POST'])
def create_book():
    form = BookForm()
    if form.validate_on_submit():
        # ... do form processing here ...
```

Validators

So, validators can catch any errors…

…but we also want to show these errors to the user!

We can access the errors associated with a given field via `form.field_name.errors`.

Errors

We can use code like this to show errors to the user:

```
{{ form.title.label }}
{{ form.title }}
<ul>
  {% for error in form.title.errors %}
    <li class="error">{{ error }}</li>
  {% endfor %}
</ul>
```

show the label

show the input element

show each of the errors in a list

Activity (10 minutes)

With a partner, add error checking to the "Books (Forms)" lab for each of the fields in the New Book form.

MAKE SCHOOL

Break - 10 min

MAKE SCHOOL

Final Project Proposal

Final Project Proposal

Go over the Final Project requirements and answer any questions.

The proposal will require you to:

- Summarize your project idea
- Create an Entity Relationship Diagram demonstrating the relationships between database models
- List the routes you will implement for the project

Lab Time

Wrap-Up

MAKE SCHOOL

- **Homework 2 (Models)** - please turn in your work by today (Tuesday)!
- **Homework 3 (Forms)** - Due on Tuesday, April 27
- **Quiz 1** - due today