Serialization

We've gotten the hang of using partials by now to send html back to our javascript, but this approach is only suitable if our app will only be used from the web. What if we want to use our API (server/routes/models) to request data on a mobile device? What if we want other people to get easy access to our data like Stripe or other popular services? In each of those cases, rendering a partial just won't cut it. We'll need to send a response back in JSON format.

What is JSON and why would we use it?

JSON stands for "JavaScript Object Notation", and it's essentially a way of formatting content in a recognizable structure, that's easily transmitted and parsed by various applications. Almost every language supports JSON, in fact, it's probably one of the most common formats you'll see when you're using an external API or even building your own. In a way, configuring our app to use JSON is a step toward making our API universally accessible.

How do we get started?

When using JSON there are a few things to know. Python has json support built in to its standard library. You can use it by importing the json module. Since python types are unique to the python language, you'll need to convert any python data into json before your server can send it back to the browser.

```
import json
# python types must be converted to json using the json module
my_list = [
    {
        "first_name": "Wes",
        "last_name": "Harper",
        "email": "wharper@codingdojo.com",
    }
]
my_jsonified_list = json.dumps(my_list)
```

Serialization

If you've been using SQLAlchemy to do your database queries we'll need to do some additional setup. Since SQLAlchemy uses custom classes to create new types, we can't use json.dumps on our queried data. Luckily, the python community has built a tool called Flask-Marshmallow.

First, install Flask-Marshmallow and Marshmallow-SQLAlchemy into your virtual environment.

```
pip install flask-marshmallow
```

```
pip install marshmallow-sqlalchemy
```

Next, you'll need to add the following lines to your config.py.

```
from flask_marshmallow import Marshmallow
# this was already here
db = SQLAlchemy(app)
migrate = Migrate(app, db)
# this is new
ma = Marshmallow(app)
```

Now setup your models to be serialized extending the ModelSchema class from Flask-Marshmallow. We can use this class to covert our Model objects into the json format we need for a proper API response.

```
from config import db, ma
class SomeModel(db.Model):
    __tablename__ = 'users'
    id = db.Column(db.Integer, primary_key=True)
    # the rest omitted for brevity
class SomeModelSchema(ma.ModelSchema):
    class Meta:
        model = SomeModel
```

And finally, in our controller, we can use our new model schema to serialize our information to json. Note the use of Flask's built-in jsonify function. Also, be careful to note the **difference** in how we deal with a **collection** of information.

```
# conveniently, Flask has a jsonify function
from flask import render_template, request, redirect, session, url_for, flash, jsonify
from server.models.some_model import SomeModel, SomeModelSchema
def one():
    one_item = SomeModel.query.first()
    some_schema = SomeModelSchema()
    output = some_schema.dump(one_item)
    return jsonify({"some_item": output})
def all():
    items = SomeModel.query.all()
    some_schema = SomeModelSchema(many=True) # this is different!
    output = some_schema.dump(items)
    return jsonify({"some_item": output})
```