

CS50 Beyond

Agenda

- ORMs
- API Design

ORMs

```
class Flight(db.Model):
    __tablename__ = "flights"
    id = db.Column(db.Integer, primary_key=True)
    origin = db.Column(db.String, nullable=False)
    destination = db.Column(db.String, nullable=False)
    duration = db.Column(db.Integer, nullable=False)
```

```
CREATE TABLE flights (  
  id SERIAL PRIMARY KEY,  
  origin VARCHAR NOT NULL,  
  destination VARCHAR NOT NULL,  
  duration INTEGER NOT NULL  
);
```

```
db.create_all()
```

```
INSERT INTO flights
  (origin, destination, duration)
VALUES ('New York', 'Paris', 540)
```

```
flight = Flight(origin="New York",
                 destination="Paris",
                 duration=540)
db.session.add(flight)
```

```
SELECT * FROM flights;
```

```
Flight.query.all()
```

```
SELECT * FROM flights
WHERE origin = 'Paris';
```

```
Flight.query.filter_by(origin="Paris").all()
```



```
SELECT * FROM flights  
WHERE origin = 'Paris' LIMIT 1;
```

```
Flight.query.filter_by(origin="Paris").first()
```

```
SELECT COUNT(*) FROM flights  
WHERE origin = 'Paris';
```

```
Flight.query.filter_by(origin="Paris").count()
```

```
SELECT * FROM flights WHERE id = 28;
```

```
Flight.query.filter_by(id=28).first()
```

```
Flight.query.get(28)
```

```
UPDATE flights SET duration = 280  
WHERE id = 6;
```

```
flight = Flight.query.get(6)  
flight.duration = 280
```

```
DELETE FROM flights WHERE id = 28;
```

```
flight = Flight.query.get(28)  
db.session.delete(flight)
```

```
COMMIT;
```

```
db.session.commit()
```

```
SELECT * FROM flights  
ORDER BY origin;
```

```
Flight.query.order_by(Flight.origin).all()
```

```
SELECT * FROM flights  
ORDER BY origin DESC;
```

```
Flight.query.order_by(Flight.origin.desc()).all()
```



```
SELECT * FROM flights  
WHERE origin != 'Paris'
```

```
Flight.query.filter(  
    Flight.origin != "Paris").all()
```

```
SELECT * FROM flights  
WHERE origin LIKE '%a%'
```

```
Flight.query.filter(  
    Flight.origin.like("%a%")).all()
```

```
SELECT * FROM flights  
WHERE origin IN ('Tokyo', 'Paris');
```

```
Flight.query.filter(  
    Flight.origin.in_  
        ["Tokyo", "Paris"])).all()
```

```
SELECT * FROM flights
  WHERE origin = 'Paris'
  AND duration > 500;
```

```
Flight.query.filter(
  and_(Flight.origin == "Paris",
  Flight.duration > 500)).all()
```

```
SELECT * FROM flights
  WHERE origin = 'Paris'
 OR duration > 500;
```

```
Flight.query.filter(
  or_(Flight.origin == "Paris",
      Flight.duration > 500)).all()
```

```
SELECT * FROM flights JOIN passengers  
    ON flights.id = passengers.fought_id;
```

```
db.session.query(Flight, Passenger).filter(  
    Flight.id == Passenger.flight_id).all()
```

Relationships

```
SELECT * FROM passengers  
WHERE flight_id = 1;
```

```
Flight.query.get(1).passengers
```



```
SELECT * FROM flights JOIN passengers
  ON flights.id = passengers.flight_id
 WHERE passengers.name = 'Alice';
```

```
Passenger.query.filter_by(name="Alice") \
                .first().flight
```

APIs

JSON

```
{  
  "origin": "Tokyo",  
  "destination": "Shanghai",  
  "duration": 185  
}
```

```
{  
  "origin": "Tokyo",  
  "destination": "Shanghai",  
  "duration": 185,  
  "passengers": ["Alice", "Bob"]  
}
```

```
{
  "origin": {
    "city": "Tokyo",
    "code": "HND"
  },
  "destination": {
    "city": "Shanghai",
    "code": "PVG"
  },
  "duration": 185,
  "passengers": ["Alice", "Bob"]
}
```

/flights/

/flights/28

/flights/28/passengers/

/flights/28/passengers/6

HTTP Methods

- GET: retrieve resource
- POST: create a new resource
- PUT: replace a resource
- PATCH: update a resource
- DELETE: delete a resource

requests

```
requests.get(url)
```

```
requests.post(url)
```

```
requests.put(url)
```

```
requests.patch(url)
```

```
requests.delete(url)
```

Status Codes

- 200 OK
- 201 Created
- 400 Bad Request
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 422 Unprocessable Entity
- ...

API Keys

Afternoon Project

- Continue with importing books database into PostgreSQL.
- Build a web application that:
 - Lets users search for books by (partial) title and/or author.
 - Links to pages for individual books.
 - Links to pages for individual authors.

CS50 Beyond