File I/O and Databases

Working with files

- Ruby is quite good at working with files
- It allows you to analyse files, parse files etc.

The Process

- Open up a new instance of the File class
    - Using a specified mode
- Work with the file (add stuff to it, get stuff from it etc.)
- Close of the instance (Ruby can only hold a connection to 1000 files or databases at a time)

File Modes

r	Read-only, starts at beginning of file
r+	Read-write, starts at beginning of file
w	Write-only, replaces the file (and will create a file if necessary)
w+	Read-write, replaces the file (and will create a file if necessary)
a	Write-only, appends data at end of file (will create)
a+	Read-write, appends data at end of file and will create a file if necessary

Let's start by reading a file

```ruby
# Create a new instance of the File class in read mode.
f = File.open( "sample.txt", "r" )

# Read a line at a time
f.readline
f.readline
f.readline

# Go back to the start of the file
f.rewind
f.readline

# Get all of the lines in an array
all_lines = f.readlines

# Close the connection to the file
f.close
```

Let's start by reading a file

```ruby
# Create a new instance of the File class in read mode.
f = File.open( "sample.txt", "r" )

# Iterate through the file, line by line
all_lines = f.readlines

all_lines.each do |line|
    puts line
end
```

Now, let's write!

```
# Create a new instance of the File class in append mode.
f = File.open( "people.txt", "a+" )

# Add new line
f.puts "Groucho"
f.puts "Chico"
f.puts "Gummo"

# Work on the same line
f.print "Zeppo, "
f.print "and Harpo."

# Close the connection
f.close
```

Now, let's write!

```ruby
# Create a new instance of the File class in append mode.
f = File.open( "people.txt", "a+" )

# Add new line
f.puts "Groucho"
f.puts "Chico"
f.puts "Gummo"

# Work on the same line
f.print "Zeppo, "
f.print "and Harpo."

# Close the connection
f.close
```

Some File Links

- Tutorial Point
- Tutsplus
- Ruby Monk
- The Bastards Book of Ruby
- Udemy
- Ruby Docs

SQL

Our first database

What is it?

- **S**tructured **Q**uery **L**anguage
- Was created by Donald D. Chamberlin and Raymond F. Boyce at IBM in the early 1970s
- Made for relational databases

Let's install it

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.co
```

This is installing something called Homebrew

- This is a package manager
- You can install lots of cool things through it

```
brew install sqlite3
brew install tree
brew install fortune
brew install cowsay
brew install cmatrix
brew install sl
```

Some important links

- Code Academy
- Khan Academy
- Learn SQL the Hard Way

CRUD!

Is absolutely everything in Web Development, it's the foundation of every application

- ***C****reate*
- ***R****ead*
- ***U****pdate*
- ***D****elete*

In every CRUD-based app...

1. The database itself
2. The tables within that database
3. Individual records on a particular table

We need to create those things!

Let's create the database

Create the database file

- touch database.db

Open up the database

- sqlite3 database.db

Let's add a table

```sql
-- This is in a file called people.sql

CREATE TABLE table_name (
    id INTEGER PRIMARY KEY,
    column_name COLUMNTYPE
);

CREATE TABLE person (
    id INTEGER PRIMARY KEY,
    first_name TEXT,
    last_name TEXT
);
```

We need to add it to the DB

```
sqlite3 database.db < people.sql
```

Let's create some records

```sql
-- This is in a file called add_people.sql

INSERT INTO table_name ( comma, seperated, columns )
       VALUES ( commas_value, seperated_value, column_value);

INSERT INTO person (id, first_name, last_name, age)
       VALUES ( 1, "Groucho", "Marx", 37 );
```

We need to add it to the DB

```
sqlite3 database.db < add_people.sql
```

Let's see what was in the DB

```sql
-- This is in a file called read_people.sql
-- SELECT what FROM what_table WHERE some_options;

SELECT * FROM people;

SELECT name FROM people;

SELECT * FROM people WHERE first_name == "Groucho";
```

We need to add it to the DB

```
sqlite3 database.db < read_people.sql
```

Let's update a person

```sql
-- This is in a file called update_people.sql
-- UPDATE table SET updates WHERE some_options;

UPDATE people SET first_name = "Harpo"
              WHERE first_name = "Groucho";

UPDATE people SET first_name = "Zeppo"
              WHERE id = 1;
```

We need to add it to the DB

```
sqlite3 database.db < update_people.sql
```

Let's delete a person

```sql
-- This is in a file called update_people.sql
-- DELETE FROM table WHERE some_options;

DELETE FROM people WHERE first_name = "Groucho";

DELETE FROM people WHERE id = 1;
```

We need to add it to the DB

```
sqlite3 database.db < update_people.sql
```

How to map these things out

Operation	Method	URL	Query	Def
CREATE	POST	/animals	INSERT	#create
	GET	/animals/new		#new
READ	GET	/animals	SELECT	#index
	GET	/animals/:id	SELECT	#show
UPDATE	POST	/animals/:id	UPDATE	#update
	GET	/animals/:id/edit		#edit
DELETE	(delete)	/animals/:id/delete	DELETE	#delete

But how do we use it with the web?

```ruby
# Require the gems

require 'sinatra'
require 'sinatra/reloader'
require 'sqlite3'
```

But how do we use it with the web?

```ruby
# Create a connection to the database
db = SQLite3::Database.new 'database.db'

# Ask for the information in a nicer format
db.results_as_hash = true

# Show the SQL that was generated in the logs
puts sql

# Execute a line of SQL and store the result
result = db.execute sql
```

Here is the homework