Website Design

CSc 47300

Week 4

Overview

- MVC Architecture
- Datastores
- Data Modeling
- ORM's / SQLAlchemy

Structuring Your Project

The Model-View-Controller (MVC) Architecture

What is MVC?

(M)odels, (V)iews, and (C)ontrollers

MVC is a software pattern that defines how we should structure and layout our application code.

Many modern web frameworks have adopted the use of the MVC pattern.

What is MVC?

What web frameworks if any have adopted the use of the MVC pattern?

What is MVC?

What web frameworks have adopted the use of the MVC pattern?

- Django (Python)
- Rails (Ruby)
- Spring (Java)

What is MVC?

What are some other design patterns that you have used or heard of?

- Singleton: Limits the creation of a class to only one object.
  - Applicable scenarios: Cache
- Strategy: Allows grouping related algorithms under an abstraction, which allows switching out one algorithm or policy for another without modifying the client/caller.
  - Applicable scenarios: Notifications (Email, SMS, Push)

Models, Views, and Controllers

Models

Models represent our data entities, and provide a mechanism to store the data to persistent storage (databases, filesystem, services, etc), and to retrieve the data.

Models, Views, and Controllers

Models

> *Models represent our data entities, and provide a mechanism to store the data to persistent storage (databases, filesystem, services, etc), and to retrieve the data.*

This is where we model the data that our web application is working with, such as: *Articles*, *Movies*, *Songs*, *Books*, *Cars*, *Users*, *Admins*, *etc*.

It is an interface to our database/persistence layer, (although there does not have to be one).

Models, Views, and Controllers

Views

> *Views generate our applications output. Primarily concerned with the presentation and display of our data.*

This is where the output presentation is handled. Minimal to no logic should exist here.

We should treat it as if it were a “fill in the blanks” template.

Models, Views, and Controllers

Controllers

Controllers contain action methods (route handlers) that receive HTTP requests, the action method processes any input (route, query, and body parameters), it calls models and services as needed, and finally uses a view to provide output for the HTTP response.

Models, Views, and Controllers

Controllers

> *Controllers contain action methods (route handlers) that receive HTTP requests, the action method processes any input (route, query, and body parameters), it calls models and services as needed, and finally uses a view to provide output for the HTTP response.*

This is where we map our desired URL route space to specific action functions in Flask.

The actions manage the lifecycle of http request and response.

Models, Views, and Controllers

MVC is only a pattern. It is up to programmer to use the pattern effectively to layer their application.

For example:

- Views should not include DB queries or business logic code
- Controllers should not directly talk to the database and it should not directly generate HTML
- Models should not be concerned with HTML output or business logic rules*

** These are the “minimum” application layers. Your business needs may require that you add additional layers between these 3 as your applications become more complex.

What is business logic?

- Authorization and Access rules
  - Who is allowed to access the data?
  - Who is allowed to modify the data?
- What input is needed?
- What actions are allowed?

Datastores

What are databases?

Databases are collections of information/data.

Types of Databases

- Relational Databases (SQL)
- Key-Value Stores (NoSQL)
- Document Stores (NoSQL)
- Graph Databases
- Many other types...

Relational Databases

Relational Databases → Structured collections of data with associations between collections.

Relational Databases

Relational Databases → Structured collections of data with associations between collections.

Collections of data are typically modeled as database tables.

Relational Databases

Relational Databases → Structured collections of data with associations between collections.

Collections of data are typically modeled as database tables.

Rows in a table represent a record (or entry) of a data point.

Relational Databases

Relational Databases → Structured collections of data with associations between collections.

Collections of data are typically modeled as database tables.

Rows in a table represent a record (or entry) of a data point.

Columns in a table represent the fields/attributes of each record. Each column in the table is typically constrained to a single data type.

Database Tables

table name:	**blogs**				
column datatype:	*int*	*string*	*text*	*datetime*	*datetime*
column name:	**id**	**title**	**body**	**created_at**	**modified_at**
4 Records	1	How to build a website	Lorem ipsum ...	10/1/15	10/1/15
	2	How to deploy with Heroku	Lorem ipsum ...	10/2/15	10/2/15
	3	Github or Bitbucket, which is best?	Lorem ipsum ...	10/3/15	10/3/15
	4	Express.js Tutorial	Lorem ipsum ...	10/4/15	10/4/15

Database Tables

table name:		**blogs**				
column datatype:		*int*	*string*	*text*	*datetime*	*datetime*
column name:		**id**	**title**	**body**	**created_at**	**modified_at**
4 Records		1	How to build a website	Lorem ipsum ...	10/1/15	10/1/15
		2	How to deploy with Heroku	Lorem ipsum ...	10/2/15	10/2/15
		3	Github or Bitbucket, which is best?	Lorem ipsum ...	10/3/15	10/3/15
		4	Express.js Tutorial	Lorem ipsum ...	10/4/15	10/4/15

- Id → Convention for every table to have an *id* column. Each record should have a unique id value.
- Created At and Modified At → Convention for each table to have these columns to track creation time and last edit time.
    - Admin/User information can be recorded as well for auditing purposes.

Movies Example

Let's take a look at another example:

id	movie_name	movie_synopsis	genre	year	actor_name	actor_dob	actor_bio	actor_salary
1	Independence Day	Blah blah blah...	sci-fi	1996	Will Smith	9/25/68	In west Philadelphia born and raised	$5M
2	Men in Black	Bleh bleh bleh...	comedy	1997	Will Smith	9/26/68	In west Philadelphia born and raised	$5M
3	I, Robot	Lorem ipsum...	sci-fi	2004	Will Smith	9/27/68	In west Philadelphia born and raised	$28M
4	I Am Legend	Hmm, blah blah...	sci-fi	2007	Will Smith	9/28/68	In west Philadelphia born and raised	$25M

Does this OK? Are there any problems?

Movies Example

id	movie_name	movie_synopsis	genre	year	actor_name	actor_dob	actor_bio	actor_salary
1	Independence Day	Blah blah blah...	sci-fi	1996	Will Smith	9/25/68	In west Philadelphia born and raised	$5M
2	Men in Black	Bleh bleh bleh...	comedy	1997	Will Smith	9/26/68	In west Philadelphia born and raised	$5M
3	I, Robot	Lorem ipsum...	sci-fi	2004	Will Smith	9/27/68	In west Philadelphia born and raised	$28M
4	I Am Legend	Hmm, blah blah...	sci-fi	2007	Will Smith	9/28/68	In west Philadelphia born and raised	$25M

Concerns with this table:

Data Redundancy → A lot of duplicate data taking up storage.

Data Integrity → What if “Will Smyth” is misspelled in 1+ entries?

What if the Bio has to be updated?

Data Modeling

Data Normalization

Normalization is the design of database tables and columns to reduce redundancy and maintain data integrity.

Data Normalization

Let's look at a better way to model the Movie data:

movies				
id	name	synopsis	year	genre_id
1	Independence Day	Blah blah blah...	1996	2
2	Men in Black	Bleh bleh bleh...	1997	4
3	I, Robot	Lorem ipsum...	2004	2
4	I Am Legend	Hmm, blah blah...	2007	2

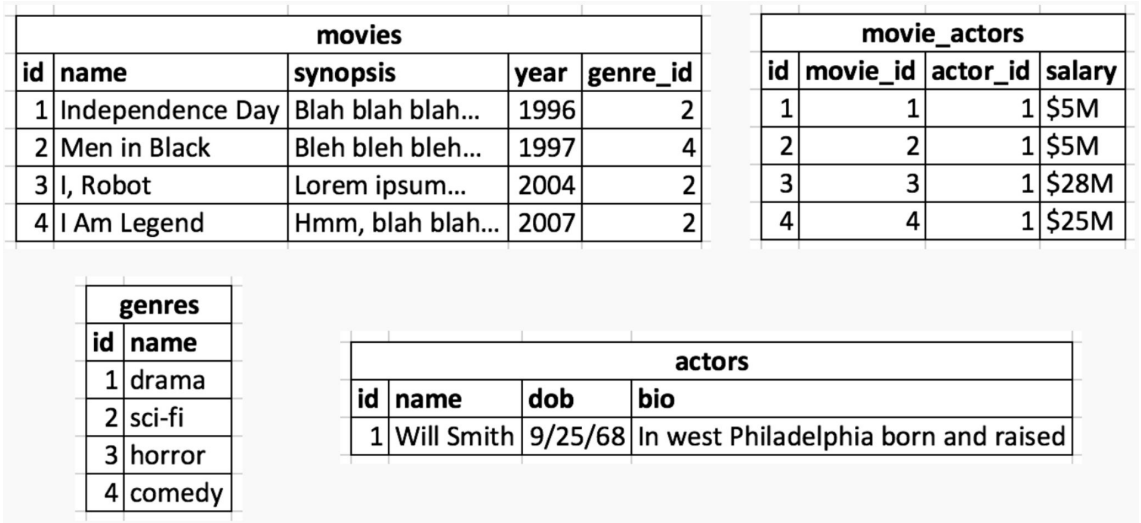
movie_actors			
id	movie_id	actor_id	salary
1	1	1	$5M
2	2	1	$5M
3	3	1	$28M
4	4	1	$25M

genres	
id	name
1	drama
2	sci-fi
3	horror
4	comedy

actors			
id	name	dob	bio
1	Will Smith	9/25/68	In west Philadelphia born and raised

Data Normalization

Why is this better?



movies

id	name	synopsis	year	genre_id
1	Independence Day	Blah blah blah...	1996	2
2	Men in Black	Bleh bleh bleh...	1997	4
3	I, Robot	Lorem ipsum...	2004	2
4	I Am Legend	Hmm, blah blah...	2007	2

movie_actors

id	movie_id	actor_id	salary
1	1	1	$5M
2	2	1	$5M
3	3	1	$28M
4	4	1	$25M

genres

id	name
1	drama
2	sci-fi
3	horror
4	comedy

actors

id	name	dob	bio
1	Will Smith	9/25/68	In west Philadelphia born and raised

Data Normalization

How is this better:

- I can add actors to a movie, without duplicating movie information.
- I can update an actor's bio once, without having to update multiple movies.
- When I add a movie, I don't have to enter data for actors that already in my database.

How is this more complicated:

- How do I determine the genre name for a given movie?
- How do I lookup all of the movies for a given actor?
- How do I find all of the actors for a given movie?
- How do I determine all the genres an actor has worked in?

SQL → Structured Query Language

Finding answers to my data questions will require looking at multiple tables.

SQL → Structured Query Language

Finding answers to my data questions will require looking at multiple tables.

SQL is a declarative programming language designed to facility querying normalized data. It helps us follow the relations in our database tables.

> *We will take a look at SQL later in the course.*

SQL → Structured Query Language

Finding answers to my data questions will require looking at multiple tables.

SQL is a declarative programming language designed to facility querying normalized data. It helps us follow the relations in our database tables.

> *We will take a look at SQL later in the course.*

Columns such as **movies.genre_id**, **movie_actors.movie_id**, and **movie_actors.actor_id** produce relations among our records.

Relations and Associations

There are three types of record associations:

- 1 to 1 (1:1)
- 1 to many (1:n)
- Many to many (n:m)

One to One Associations

If we look at our movie dataset, we can split the actor table into two tables, an **actor** and **actor_bio** table.

actors			
id	name	dob	actor_bios_id
1	Will Smith	9/25/68	6

actor_bios	
id	bio
6	In west Philadelphia born and raised

In our modeling we say: Each actor **has one** actor_bio.

This is useful when one table has many fields, but many are less frequently used than others. The split is done for query performance.

One to Many Associations

genres	
id	name
1	drama
2	sci-fi
3	horror
4	comedy

movies				
id	name	synopsis	year	genre_id
1	Independence Day	Blah blah blah...	1996	2
2	Men in Black	Bleh bleh bleh...	1997	4
3	I, Robot	Lorem ipsum...	2004	2
4	I Am Legend	Hmm, blah blah...	2007	2

In our modeling we say:

- Each genre has many movies.
- Each movie belongs to a single genre.

Many to Many Associations

movies				
id	name	synopsis	year	genre_id
1	Independence Day	Blah blah blah...	1996	2
2	Men in Black	Bleh bleh bleh...	1997	4
3	I, Robot	Lorem ipsum...	2004	2
4	I Am Legend	Hmm, blah blah...	2007	2

movie_actors			
id	movie_id	actor_id	salary
1	1	1	$5M
2	2	1	$5M
3	3	1	$28M
4	1	2	$4M
5	4	3	$15M
6	4	1	$25M

actors			
id	name	dob	bio
1	Will Smith	9/25/68	In west Philadelphia born and raised
2	Jane Doe	11/26/57	Lorem ip...
3	John Ramon	9/27/89	Lorem ipsum...

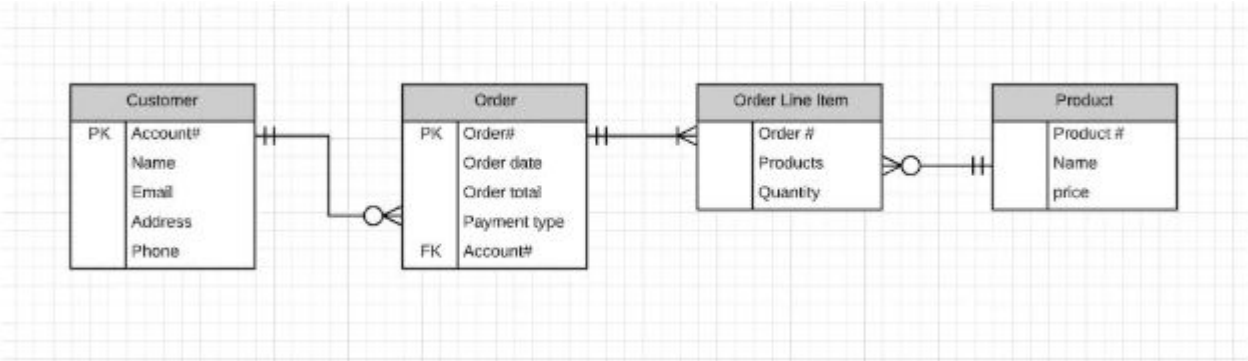
In our modeling we say:

- Each movie has many actors through movie_actors.
- Each actor has many movies through movie_actors.

Data Modeling

A visual way to model tables and relations is using ER-Diagrams (entity-relations). Each table is modeled as an object, and relations are arrows.

Entity Relationship Diagrams



[Cardinality Reference]

Further Reading

Terms to know

- RDBMS → Relational Database Management System
- Primary Keys
- Foreign Keys

About Databases

- Introduction to SQL → https://cs.lmu.edu/~ray/notes/introsql/
- [Extra] Codd's 12 Rules → https://www.tutorialspoint.com/dbms/dbms_codds_rules.htm
- [Advanced] Database Normalization: → https://en.wikipedia.org/wiki/Database_normalization

ORM's / SQLAlchemy

ORM → Object Relational Mapping

The majority of our database tables and associations fit into an object and association abstraction.

ORM's are software tools that help us create objects for each table, and provide us CRUD and association methods for each object.

SQLAlchemy → https://www.sqlalchemy.org/

SQLAlchemy is an ORM for Relational databases (Postgres, MySQL, Oracle, etc)

ORM → Object Relational Mapping

ORM's generate SQL code for interacting with our databases.

ORM's are not a magic tool for every case. Given very complicated relations and chainings, it may be more formant to write your own SQL command, then to rely on the ORM's generated SQL.

Questions? Concerns?