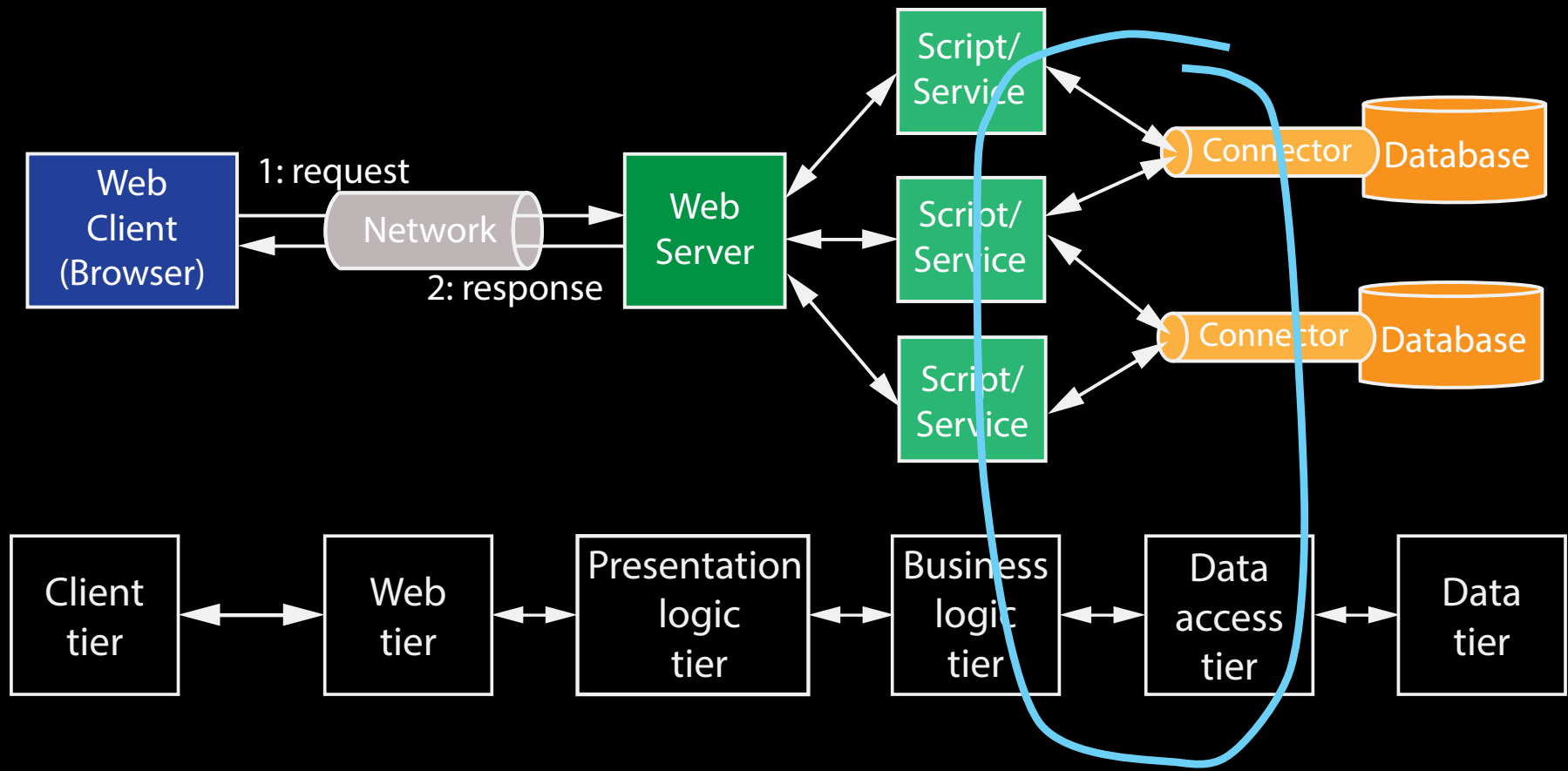
Web Application Architectures

Module 3: Database Interactions
Lecture 3: The Active Record Design Pattern

THE UNIVERSITY of NEW MEXICO

Focus of our Attention



Active Record Design Pattern

- The Active Record design pattern was named by Martin Fowler in 2003 in his book *Patterns of Enterprise Application Architecture*.
- Active Records are commonly used in Ruby as a means of persisting data, i.e., of saving data persistently, so that it can be recalled for later use, even after you've closed the program that created the data, and turned off your computer.
- More specifically, the Active Record pattern is used to access data stored in relational databases — it allows you to perform CRUD operations without worrying about the specific underlying database technology (e.g., SQLite, MySQL, PostgreSQL, SQL Server, Oracle, etc).

Active Record Design Pattern

- Most software applications today are written using some OO language, and often it is necessary to persist (i.e., save and later retrieve) the objects associated with these applications.
- There's a big problem: The classes and objects associated with an OO language are incompatible with the structure of relational databases.
- Active Records to the Rescue: This design pattern encapsulates that notion an object-relational mapping (ORM), i.e., a mapping between OO language constructs and relational databases constructs.
- The ORM provided by Active Records automatically converts object into constructs that can be stored in a database (and converts them back upon retrieval).
- This creates, in effect, a "virtual object database" that can be used from within an OO language.

ORM – How It's Done

Active Records use the following ORM:

OO Language Feature		Relations DB Item
Classes	$\longrightarrow$	Tables
Objects	$\longrightarrow$	Records (Rows in a Table)
Attributes	$\longrightarrow$	Record Values (Columns in a Table)

Active Records in Ruby

THE UNIVERSITY of NEW MEXICO

- The Active Record design pattern is provided in a Ruby module called `ActiveRecord`.

- Using the functionality provided by this module you can:
  - Establish a connection to a database.
  - Create database tables.
  - Specify associations between tables that correspond to associations between the Ruby classes.
  - Establish an ORM between Ruby classes/objects/attributes and the tables/rows/columns in the underlying database.
  - Peform CRUD operations on Ruby `ActiveRecord` objects.

- The `ActiveRecord` module is built into Rails – the functionalities above are utilized when you create a Rails app and run scaffold and model generators.

Active Records in Rails

THE UNIVERSITY of NEW MEXICO

- The `ActiveRecord::Base.establish_connection` method uses the information in ./conifg/database.yml in order to connect a Rails application to a database.
- The `ActiveRecord::Migration` object is used to incrementally evolve your database schema over time – migrations update the ./db/schema.rb file.
- The `ActiveRecord::Schema.define` method, in ./db/schema.rb, is created by inspecting the database and then expressing its structure programmatically using a portable (database-independent) DSL. This can be loaded into any database that `ActiveRecord` supports.

ActiveRecord Module

THE UNIVERSITY of NEW MEXICO

- If you create a new class by inheriting `ActiveRecord::Base`, and call it `Post`, it is assumed a database table will exist that is called `posts`. I.e., it pluaralizes the name of the class, and then looks for a table with that name.

- The `Base` class in the `ActiveRecord` module will inspect the `posts` database, and determine that it has `title` and `body` fields, and it will automatically add member variables (and accessors) with these same names in the `Post` class. I.e., it takes care of the ORM!

- Furthermore, a query interface is also provided – in most cases, `ActiveRecord` insulates you from the need to use SQL.

  Ex.
  ```
  Post.all
  Post.first
  Post.find_by(1)
  Post.find_by_title("My First Post")
  ```

The Interactive Ruby Shell

THE UNIVERSITY of NEW MEXICO

- The interactive Ruby shell (IRB) is an interpreter that is provided with Ruby, and allows for real-time experimentation through interactive sessions – great for debugging. To use it, in a terminal window type:

```
$ irb
```

- The Rails console allows you to interact with a Rails application through the IRB command line – i.e., it starts IRB and loads the Rails environment. To use it, from the root of a Rails application directory type:

```
$ rails console
```