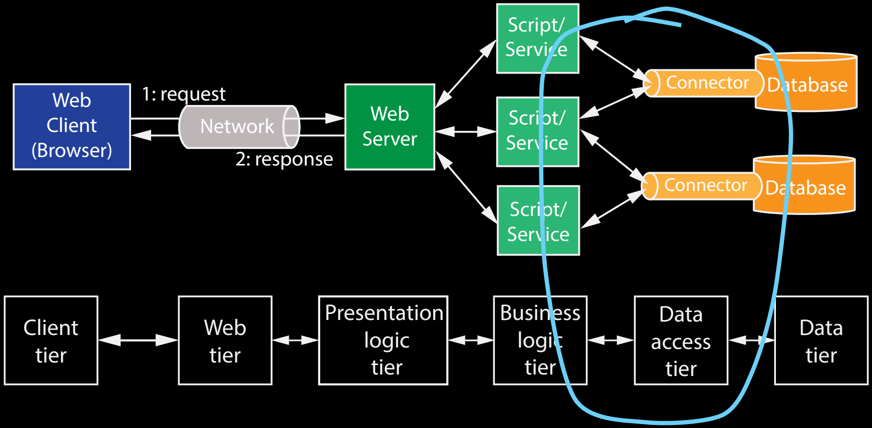
Web Application Architectures

Module 3: Database Interactions
Lecture 2: Databases in Rails

THE UNIVERSITY *of* NEW MEXICO

Focus of our Attention



Database Migrations

- Each time we ran the scaffold generator in our blog application, Rails created a database migration file and placed it in the `db/migrate` directory.
- Rails uses the `rake` command to run these migrations, which creates a schema, from which appropriate databases can then be created.
- Because migrations can also be used to undo the work of previous migrations (e.g., to make corrections), they are timestamped, and executed in the order they were created. The filename contains the timestamp.

Rails Environments

- Rails automatically sets up applications to run in one of three prebuilt environments (you can also add your own):

  Development – Used when you're developing the application.
  Test – Used when your run tests.
  Production – Used when you deploy your application.

- By default, when you run:

  ```
  $ rails server
  ```

  Rails runs in the development environment.

- To force rails to run in a different environment, use:

  ```
  $ rails server -e production
  ```

Rails Databases

One of the architectural elements most likely to differ between development and production environments is the database:

- During development, you (the developer) are the only one accessing the database.
- Rails automatically sets up the development environment to use SQLite, a simple easy-to-use file-based relational database that runs in memory. SQLite is not a production grade database.
- When an application goes into production, it may receive 100's or 1000's of "hits" in a matter of seconds, and the database needs be able to handle the data requests associated with these hits.
- Popular production database include: PostgreSQL and MySQL.
- The databases that Rails will use in different environments is specified in: `db/database.yml`.