DATABASES APP DEMO

ΕΜΠ ΗΜΜΥ Βάσεις Δεδομένων 2022

Section 1 – Introduction

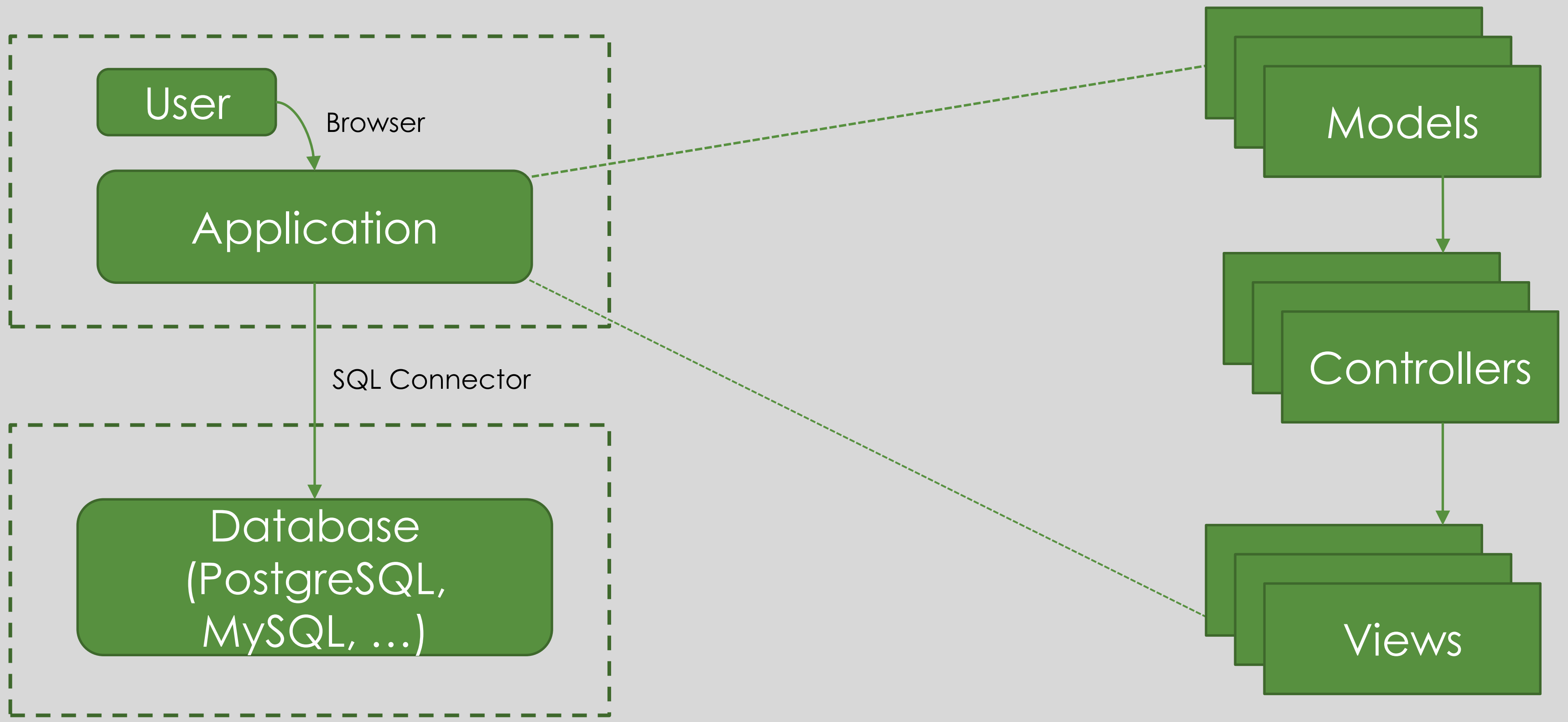
- Basic Concepts
- Client – Server Architecture
- Model – View – Controller (MVC)
- Show Schema used in Demos
- General Idea of Implementation
- Dummy data generation
- Do’s and Don’ts

Basic Concepts

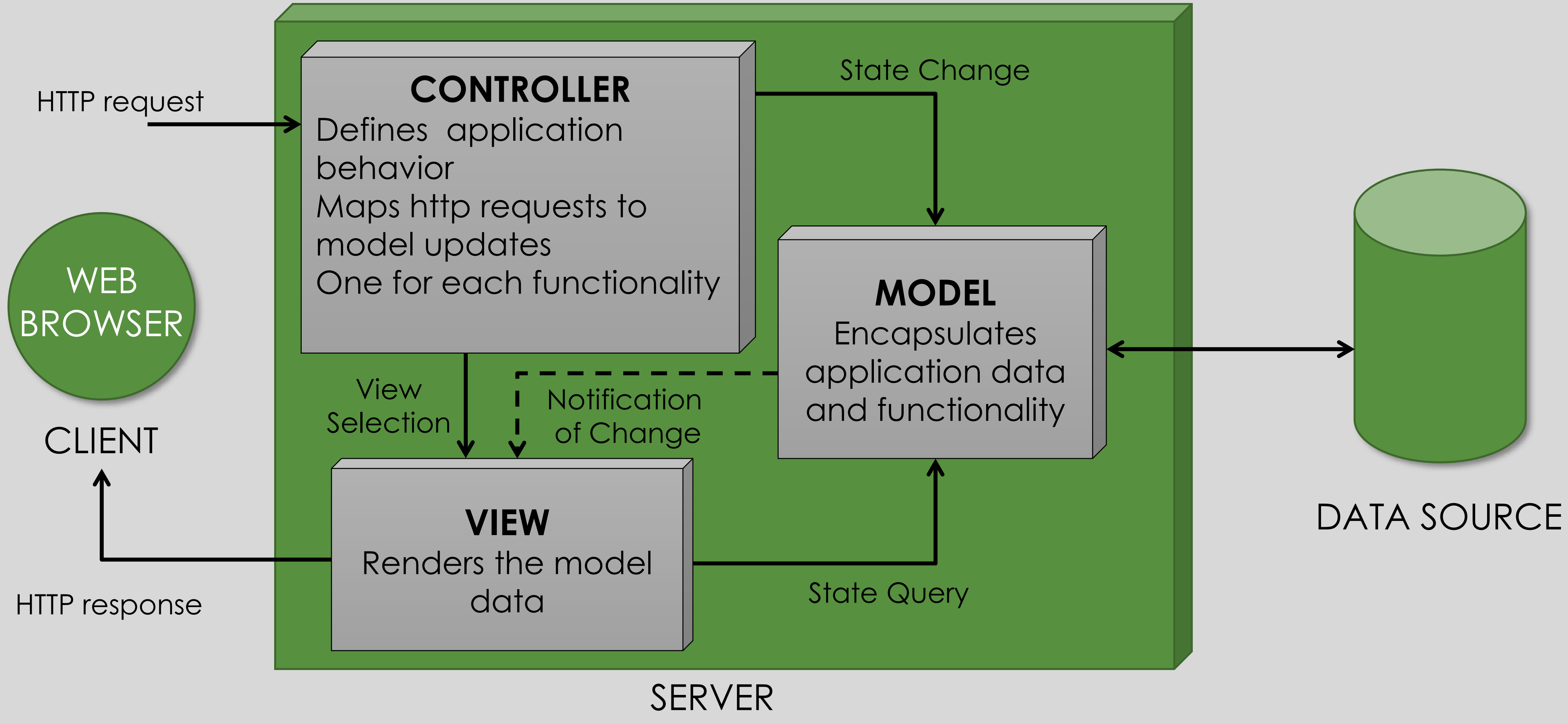
- *Endpoint*: A unique URI where a specific task is completed. For example, http://localhost:3000/signin is used for signing in to an application, http://localhost:3000/user/5 gets the user with id equal to 5 etc.
- *HTTP requests* (GET, POST, etc): Network requests sent to the server from the client via the HTTP protocol, with the purpose of accessing a resource (endpoint). GET requests retrieve data, while POST requests send data.
- *Route*: Consists of an endpoint and an HTTP request type.
- *Templating engine*: A *template engine* enables you to use static template files in your application. At runtime, the template engine replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client. This approach makes it easier to design an HTML page.

Operations we want to support:
Create
Read
Update
Delete

Client - Server Architecture

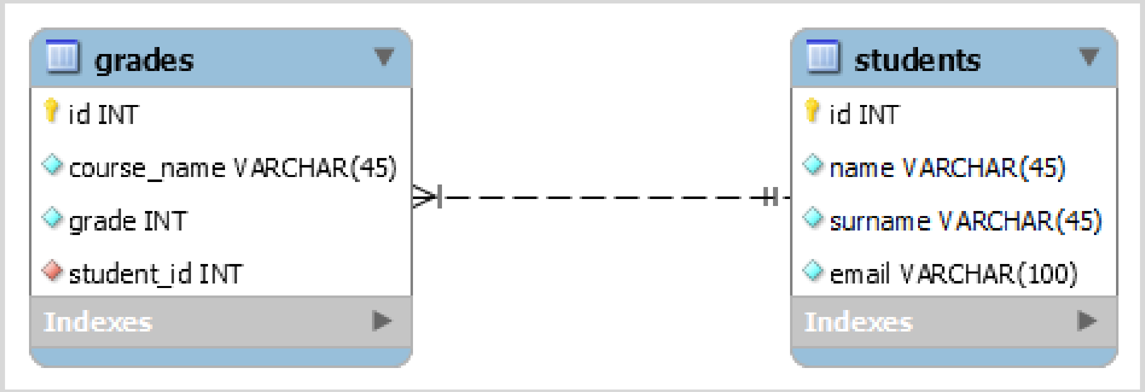


Model - View - Controller



Jacyntho, Mark, et al. "A Software Architecture for Structuring Complex Web Applications". *Journal of Web Engineering 1*, 2002, p.37-60.

Schema used in Demos



General Idea of Implementation

Assumptions:

- ER translated to SQL (SQL Database is active)
- Connection established
- (Dummy data exist in the Database → see next page)

1. SQL Query written and works (can check from Database Administration Tools like Dbeaver, Navicat, MySQL Workbench)
2. Decide Route (endpoint and HTTP request type)
3. Create the respective Controller (Where the query takes place and HTML is returned)
4. Create the respective View (that is called in Controller)
5. Repeat

Dummy data generation

- Dummy data can be generated with the help of libraries such as:
  - Faker JS
  - Faker Python
- Be careful when creating data for related tables
- Examples below

JavaScript

Python

Do’s and Don’ts

Don’t Use:

- Object–Relational Mapping libraries (ORMs) such as
  - SQLAlchemy (Python)
  - SQLObject (Python)
  - TypeORM (JS)
  - Sequelize (JS)

  etc etc

Do Use:

- Only raw SQL queries can be used for this project.
- Any front-end framework you are familiar with.

Section 2 – Demo Implementations

Section 2.1 – NodeJS Implementation

Required Dependencies

- express: Minimal and flexible Node.js web application framework that provides robust set of features to develop web and mobile apps
- mysql2: MySQL driver – providing connectivity for the client
- ejs: Templating engine

Optional Dependencies

- express-session: Create session middleware – keep user specific data
- connect-flash: The flash is a special area of the session used for storing messages
- nodemon: A cli utility that wraps the Node app, watches the file system and automatically restarts the process on changes (development)
- custom-env: configure different environment variables for the project, based on the environment (localhost, development)

Section 2.1 – NodeJS Implementation

Project Structure

- controllers: the logic of the routes (where the sql queries take place, and then the result can be either returned as a rendered page, or in other formats such as JSON, csv etc.)
- public: all publicly accessible files to which a user of the system can have access (usually css, icons, static pages etc)
- routes: the endpoints of the application.
- utils: can contain any utilities you want for your project, for example the connection of the database, so as you can just import and use it when needed (mostly in controllers)
- views: the pages of the project

Section 2.2 – Java Implementation

Initialize Project in IntelliJ

- File > New > Project > Java Enterprise
- Select Application Server the latest Tomcat Server from the dropdown menu
- Build Tool - Maven, Test runner - JUnit, Languages Java. Press Next
- Choose Web Profile. Press Next
- Name the project, Name the Group (package)

Project Structure

- src/main/java/<package-name>/ : contains the Java Classes (Controllers – Logic of the Web App)
- src/main/java/webapp: contains the assets (css, images), WEB-INF folder, JSP pages
- src/main/webapp/WEB-INF: contains lib folder and inside the lib folder MySQL connector jar file must be included

Section 2.2 – Java Implementation

Advice – Tips

- How to add the MySQL connector jar file
  - Create a new directory under WEB-INF named lib
  - Unzip the downloaded file
  - Insert the .jar file inside lib folder
- Remove <project_name>_war_exploded url base
  - Stop the server (if running)
  - From Menu Bar above press → Run
  - Debug
  - Edit Configurations
  - Select your server (Tomcat)
  - Deployment
  - From Application Context Bar delete (packageName)_war_exploded/
  - Press Apply and Debug

Section 2.3 – Python Implementation

Technologies used:

- MySQL
- Python
  - Flask (Jinja templating engine)
  - Flask-MySQLdb
  - WTForms

Connecting to the database:

- *Connection objects* represent a connection to the database (TCP/IP), and are the interface to commit operations and create cursor objects.
- *Cursor objects* represent SQL statements submitted as strings.
- *Query results* of SELECT statements, in the form of sequences of sequences (e.g. list of tuples).

Section 2.3 – Python Implementation

Project structure

- Single module: “dbdemo”
  - `__init__.py` is executed as soon as the module is imported. It configures the application, including the necessary information and credentials for the database.
  - `routes.py` contains all the endpoints and corresponding controllers.
  - folder `templates/` contains the views.
- `run.py` launches the server and runs the app on it.
- `requirements.txt` contains all dependencies in such a format that they can be batch-installed with pip.
- `db-project-demo.sql` is the SQL code (DDL and DML) needed to create the demo’s toy database.

Section 2.4 – PHP Implementation

Technologies used:

- Apache Server: Open-source HTTP Server ("httpd"), the most popular web server
- MySQL: MySQL Server
- PHP: Open-source general-purpose scripting language

Optional Dependencies

- phpMyAdmin: Software tool written in PHP, intended to handle the administration of MySQL over the web.

Otherwise: "AMP" Stacks

- A: Apache HTTP Server
- M: MySQL Database Server
- P: PHP / Perl / Python

Examples: XAMPP, LAMP, MAMP, WAMP

Section 2.4 – PHP Implementation

Project Configuration:

- Both Apache and MySQL servers up and running
- Configure Apache (`httpd.conf` file) **`Servername`** and **`Listening Port`** (e.g.: `localhost:80` or `localhost:8080`)
- Set Apache's **`DocumentRoot`** directory (usually called and preconfigured as `htdocs` in most "AMP" Stacks)

Project Structure:

- `db_connection.php`: Configure the connection to the database
- `index.php`: Home Page of the project (if `index.html` file exists rename it to `index.php` or change priority in Apache configuration file)
- Every `.php` file is in fact an html page with inline code snippets of php for the execution of queries.
- Go to a web browser and type `localhost:80/index.php` (or simply `localhost:80`) to access the home page of the project.
- Every webpage of the project can be accessed via an address such as `localhost:80/example.php`

References

- Git & Github lesson: https://www.youtube.com/watch?v=4A9upA8aDRA
- For Java (IntelliJ license): https://education.github.com/pack
- Discord

THE END