ECE 448/528
Application Software Design

Lecture 20. The MVC Pattern

Spring 2025

Won-Jae Yi, Ph.D.

Department of Electrical and Computer Engineering
Illinois Institute of Technology

The MVC Pattern

More UI Features for Groups

- User stories
  - As an end-user, I want to list groups a member belongs to and does not belong to, so I can see group/member relations for a specific member.
  - As an end-user, I want to remove a member from the groups it belongs to, and to add it to a group it doesn't belong to, so that I can manage group/member relations for a specific member.
- Can we support such user stories without changing the backend?
  - In other words, can we achieve this on the front end?
  - As a quick way to address end users' requirements.

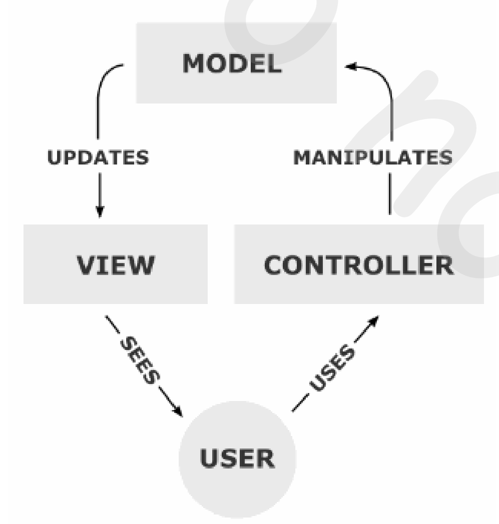
Implementation Ideas

- Yes, we can implement those user stories just at the frontend.
- A data structure to search for groups given member name.
  - Previously, our JSON data structure supported searching for members given group name.
- UI elements to display information.
  - e.g., a table with rows as members and columns as groups.
- Event handlers
  - Translate UI inputs into RESTful requests.
  - Process RESTful responses to update the data structure.

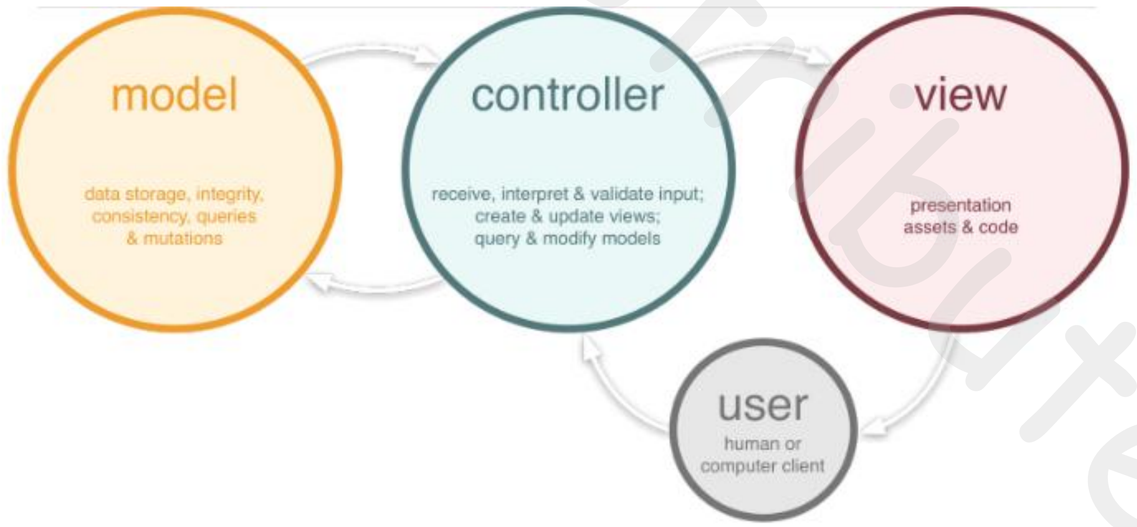
The MVC (Model-View-Controller) Pattern

- A design pattern for applications with complex graphical user interface (GUI).
  - Dates back to the 70s when GUI was introduced.
  - A lot of variants and implementations have been created since then.
  - Still widely used today for applications that interact with users via GUI, e.g., web and mobile applications.
- Consisting of three components with distinct responsibilities.
  - Model: the data structure.
  - View: the UI elements.
  - Controller: the event handlers.
- Please note that we discuss only the general concepts but not specific implementations.

The MVC (Model-View-Controller) Pattern



- User uses the Controller to manipulate data
- The Controller manipulates the data through the Model
- The Model updates the View (data visualization) to show the data to User
- User sees the data through the View


model
data storage, integrity, consistency, queries & mutations
controller
receive, interpret & validate input; create & update views; query & modify models
view
presentation assets & code
user
human or computer client

MVC Responsibilities: Model

- Model: the data structure
  - Manage data with a data type, providing methods to support CRUD operations.
  - Similar to RESTful data models.
- Two kinds of models for web applications
  - Copies of RESTful data models obtained from the server backend.
    - Since such models are pretty much caches for the RESTful data models, we will discuss very similar issues later.
  - Models related to the current UI behavior, e.g., to sort rows according to a certain column.
  - While in most cases such models are not sent to the server backend, there are cases such models are stored either on the server backend or locally as user preferences.

MVC Responsibilities: View

- View: the UI elements
  - Render UI elements according to the models.
  - Associate event handlers with UI elements.
- Views are not supposed to…: define event handlers.
  - e.g., to modify models or to send RESTful requests when the user clicks a button.
- In other words, models are read only to views.
  - So that when working with views, you just need to reason with the models as they are currently, without the need to reason with their changes.

MVC Responsibilities: Controller

- Controller: the event handlers.
  - Send RESTful requests to update data models on server backends.
  - Modify models.
  - Notify views that models have been modified so that views will render themselves.
- Two kinds of event handlers for web applications.
  - To be associated with UI elements to handle user inputs.
  - To handle RESTful responses.
- In other words, controllers are what drive the whole application.

The MVC Pattern

Model

- Application information, initial values of variables, constant values, etc..
- Model should have all data that the user wants to manipulate
  - e.g.) attributes of data size, location on the display, contents, formats, etc.

View

- UI components such as Input text, checkbox, …
- Should not save any information that the Model has…
  - View is only responsible for rendering the information on the display

Controller

- A bridge between the Model and the View
- Event Handler
- Should know about Model and View and should monitor any changes from Model or View

The MVC Pattern

- Why use MVC Pattern?
  - Model ‘team’ can focus on Model development only…
  - View ‘team’ can focus on View development only…
  - Controller ‘team’ can focus on Controller development only…
  - This delivers better maintenance, application extensibility and flexibility, eliminates duplicated coding.

MVC as (Finite) State Machine

- While MVC is apparently just a design pattern for software, it directly corresponds to a finite state machine (FSM) that hardware designers are familiar with.
    - Not a surprise as FSM is quite universal to reason with complex computations.
    - Software community usually refers FSM as state machines.
- Models represent state of the application.
- Views are the output function.
- Controllers as event handlers define state transitions.

Reasoning with FSM for MVC Designs

- Should we update models right after sending RESTful requests?
  - Yes, we should update the state, thus models, since we send RESTful requests in event handlers, and they are state transitions.
  - The application would go into “waiting for response” states and views should show users “waiting for response”.
  - Though in practice, we omit that to make code simple; we need to keep in mind that if there is a huge delay for a RESTful response, not notifying the users will confuse them.
- How to handle hierarchy in MVC?
  - We use hierarchy to address complexity in applications.
  - As an FSM, we will add hierarchy to models, views, and controllers, respectively, instead of having MVCs of MVCs.

UI Mockup

UI Mockup

- To give a visual demonstration of what the UI looks and feels like.
  - End users may give feedback on what they like or don’t like.
  - Developers may decide if things are feasible and may reason about implementation details.
- As “tests” in TDD for UI design.
  - Facilitate communication between users and developers.
- Focus on the View in the MVC pattern.
  - No, you don’t need to build the whole application in order to show users how it looks and feels like.

Our Approach to UI Mockup

- While professional tools are available for UI mockup, you can always start with the 'drawing board'.
  - Draw the UI using your intuition.
  - Treat yourself as the end user.
- As we are building web applications, we may build a concrete UI example with HTML/CSS.
  - Help to understand how HTML/CSS are integrated.
  - Help to reason with model and controller designs.
  - Provide templates to build views.
  - In `public/members_mockup.html` under branch `lec20-mockup`