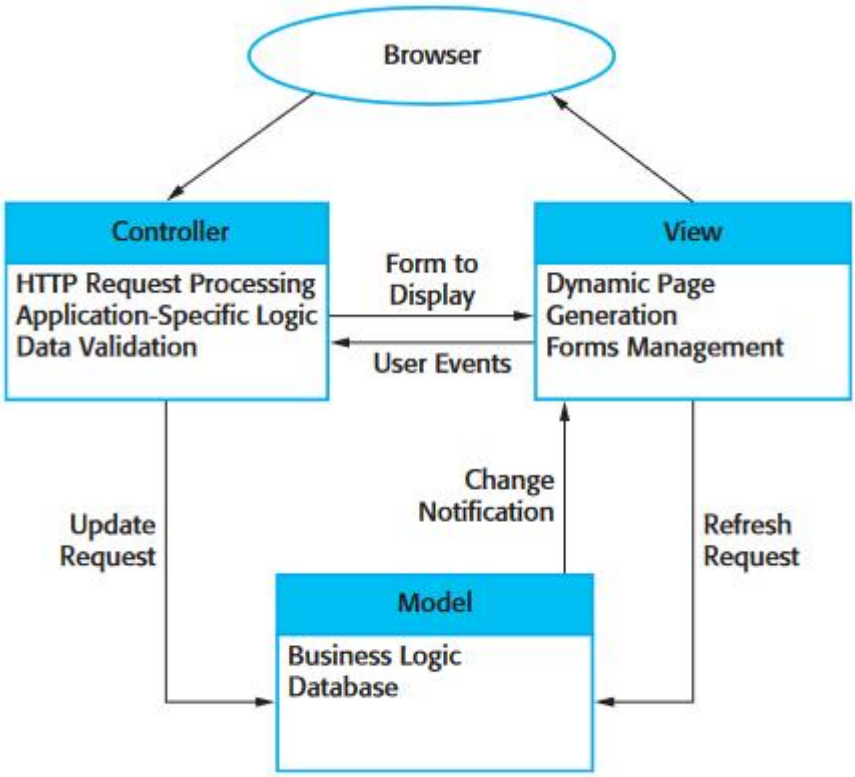
1. **Q1:** How we can understand the MVC workflow in a more complex scenario, regarding the extra step like User Event (View -> Controller), Change Notification (Model -> View) and Refresh Request (View -> Model).



A:

- With respect to User Event (View -> Controller), we should first understand the **user events** mean the actions taken directly by the user when they are interacting with the interface. They can include actions like clicking buttons, typing into input fields, submitting forms, or selecting items in a drop-down menu. In MVC, the View captures these user events because they happen directly on the interface elements. In contrast, **user requests** are a more formalized type of user event that usually results from certain user actions, often requiring the Controller to respond. E.g., Suppose a user clicks the "Login" button and submits their username and password. This is a user event captured by the View. Then, the View forwards this action as a user request to the Controller (often as an HTTP request in a web app). To sum up, the workflow is still the same, where the controller is responsible for handling the user request. The extra User Event (View -> Controller) information flow acts as a "pre-processing" step in the View component. The View captures a user action (like a physical interaction, e.g., a click or form submission) and "translates" it into a structured request (a system command) that the Controller can process.
- With respect to Change Notification (Model -> View), this typically occurs after the Model's data has been modified as a result of some user action or system process. So the process would be like:
  - The Controller initiates an update in the Model (e.g., updating a record, adding a new item).

- The Model completes the update and triggers a Change Notification to inform the View that the data has changed.
- The View, once receiving this notification, can refresh or re-render itself to display the updated information to the user.

- With respect to the Refresh Request (View -> Model), it is initiated by the View when it needs to retrieve the latest data from the Model to ensure it's displaying up-to-date information. A Refresh Request typically occurs when the user performs an action that require the View to get the latest data from the Model (like navigating back to a page and expects the data to be up-to-date, or a user simply clicking the refresh button). It works like:
  - The View detects that it needs updated data.
  - The View sends a Refresh Request to the Model, requesting the latest information.
  - The Model responds to the request by providing the up-to-date data.
  - The View then updates itself based on the new data provided by the Model.

Q2: We say the layered pattern is not good at performance (e.g., response time/speed) mainly because the request has to go through every each of the layers. However, the request in the MVC should also go through all the components, why its performance is not a significant issue?

A:

- In a layered architecture, requests typically MUST pass through each layer sequentially. MVC, on the other hand, allows the Controller to interact directly with the Model and View without unnecessary intermediaries, creating a more streamlined flow.
- Each layer in a layered pattern often performs its own set of checks or data transformations, which can lead to redundant processing. E.g.,, data might be validated or formatted at multiple layers (e.g., in both the business logic and data access layers), adding additional processing time. MVC reduces this redundancy by centralizing much of the control in the Controller, which coordinates data retrieval, updates, and transformations more efficiently.
- Overall, while both the layered pattern and MVC involve requests flowing through multiple components, the MVC pattern (IN GENERAL, BUT NOT ALWAYS, DEPENDING ON THE EXACT SITUATION) is more flexible in how these components interact.