Model-View-Controller (MVC)

- A compound pattern consisting of Observer, Strategy, and Composite patterns
- Widely used in GUI and web applications (Model 2)
- Decouples view from model
  - View is there for users to see and interact with
  - Model contains application logic and data
  - Controller passes user input from view to model as necessary – model changes its state and notifies view

MVC

- Model makes use of Observer so that it can keep observers (view) updated, yet stay decoupled from them
- Controller is the Strategy for the View. The View can use different implementations of the Controller to get different behavior
- The View may use the Composite pattern to implement the user interface, which usually consists of nested components like panels, frames, and buttons

MVC

- MVC allows the three entities to be decoupled allowing for designs that are clear and flexible
  - Model does all work independent from View and Controller – it does not depend on them at all
  - View only cares about data/information it gets from Model
- Permits independent development, maintenance, and testing of the three entities

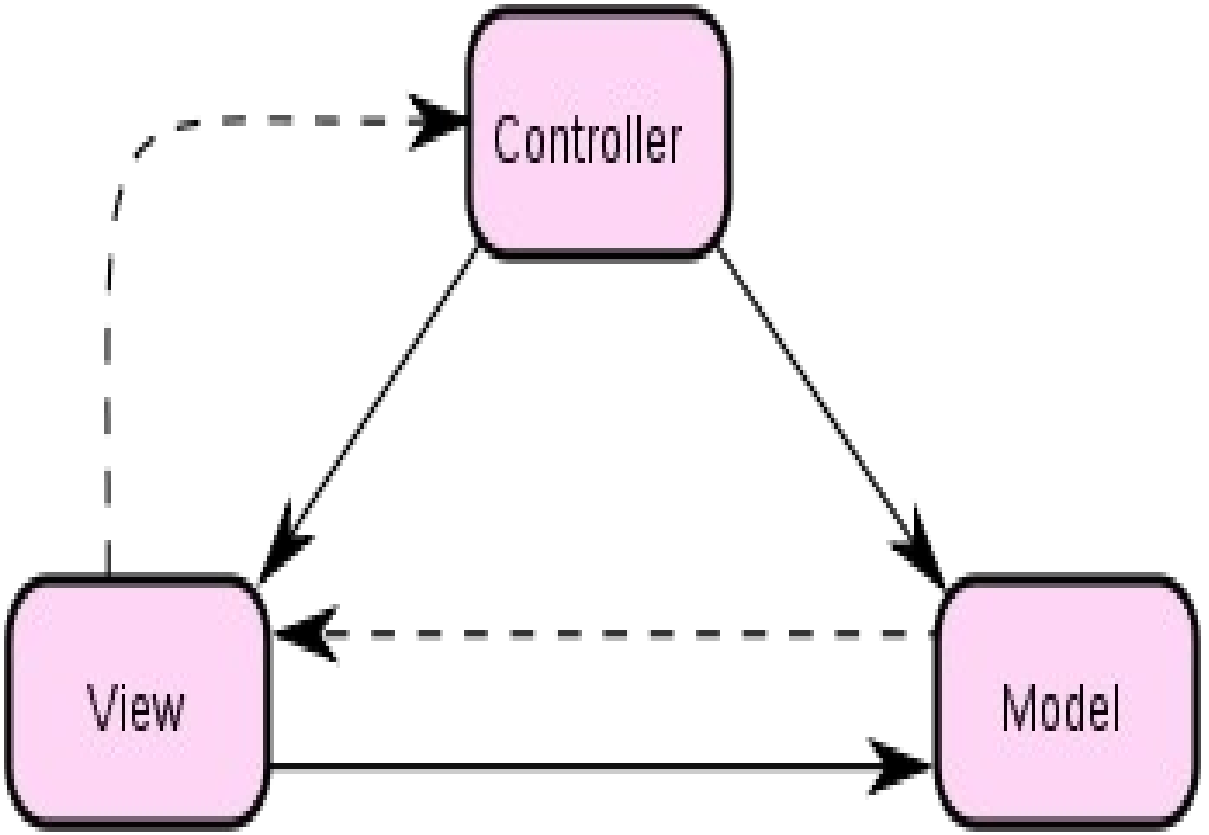
MVC Flow

- User interacts with user interface (view) in some way (mouse click)
- Controller handles the input event from the user interface (view sends info to controller), often via a registered handler/callback, and converts the event into something understandable by the model
- Controller notifies the model, possibly resulting in a change in the model's state
- View queries the model in order to generate an appropriate user interface. The view gets the data from the model (typically). In an Observer setup, Model will notify all registered observers (views)
- User interface waits for further user interactions, which restarts the control flow cycle

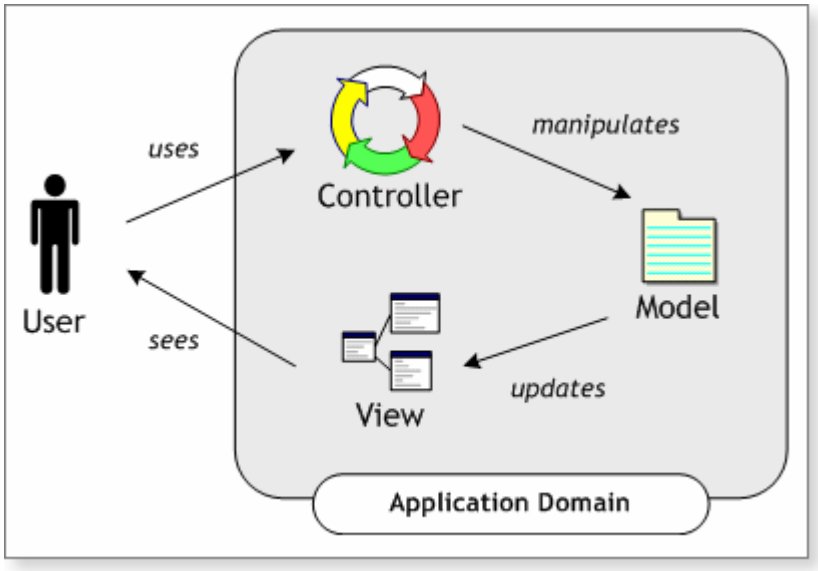
Model

Manages the behavior nad data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). It can notify observers (views) when information/state changes so they can react accordingly.

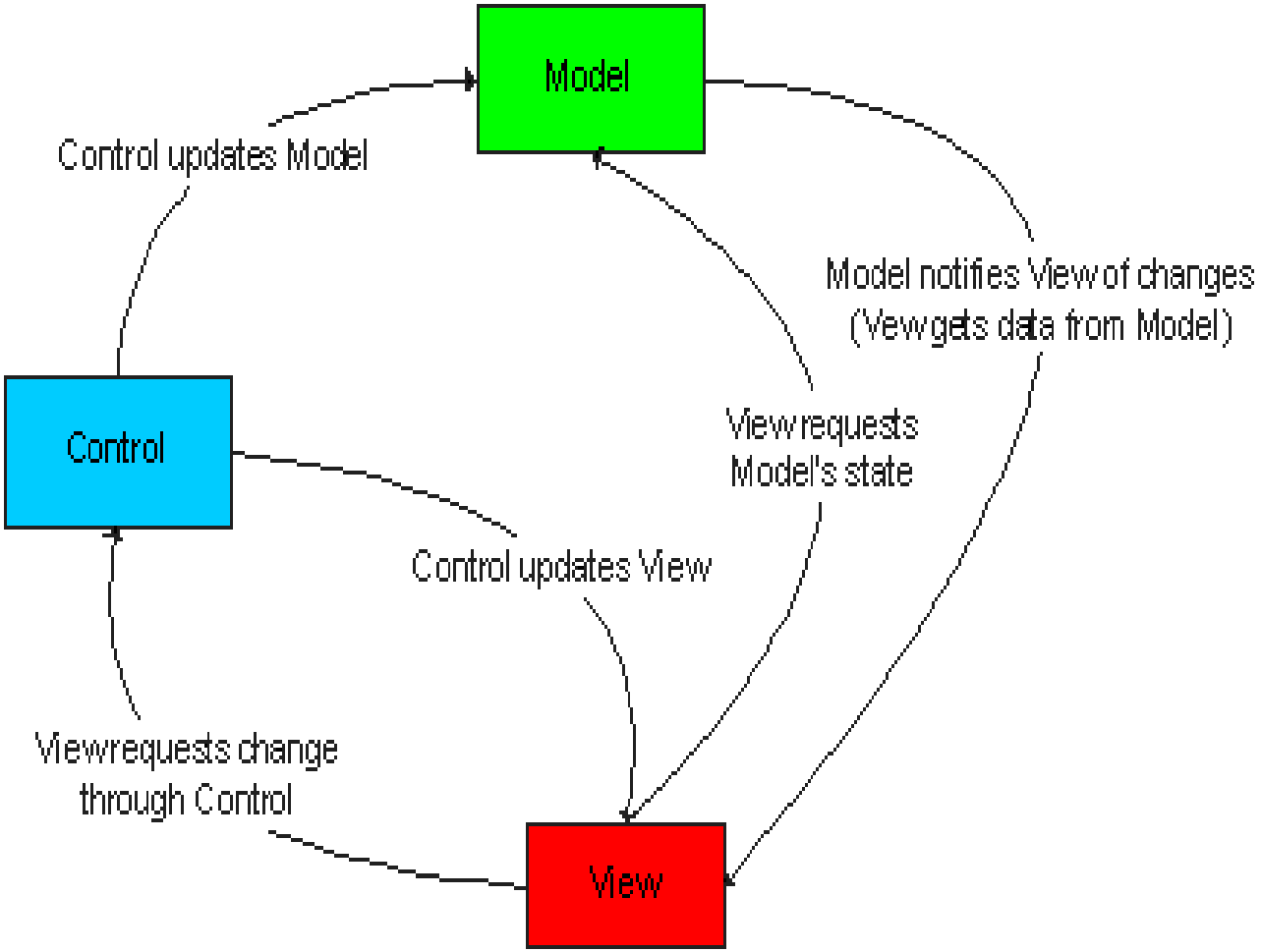
UML



MVC Graphic


uses
manipulates
Controller
User
Model
sees
updates
View
Application Domain

MVC Graphic 2



View

Renders the model into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes.

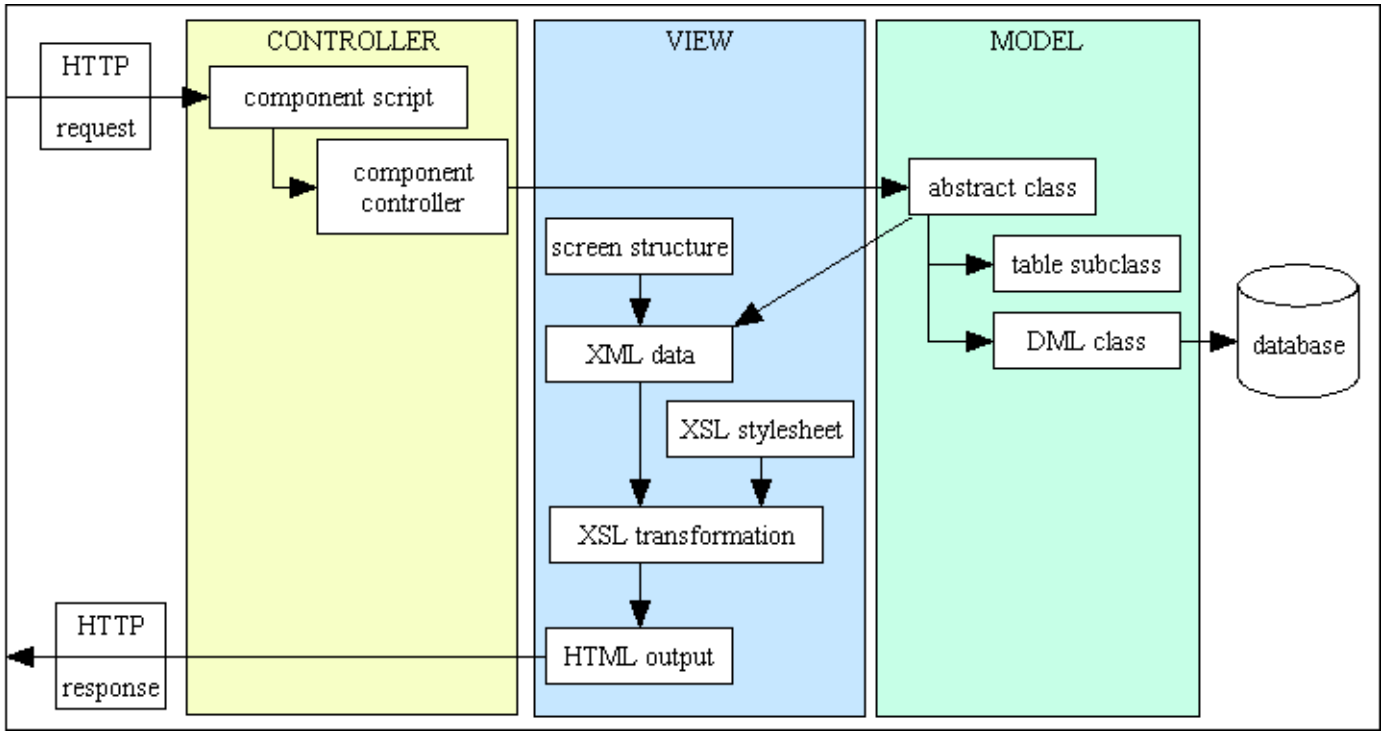
Controller

Receives input and initiates a response by making calls on model objects. It accepts input from the user (via view) and instructs the model and possibly view to perform actions based on the input.

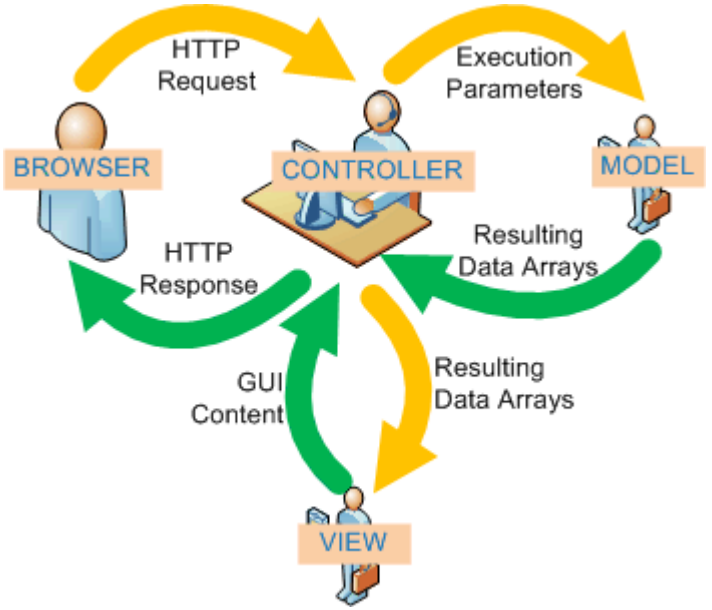
Model 2

- An adaptation of MVC for web applications
- Controller is a servlet
- JSP (using a Java model) & HTML implement the View
- Model is typically the database backend

Model 2 Graphic



Model 2 Graphic 2



MVVM

- Model View ViewModel
- Originated at Microsoft as a specialization of the Presentation Model design from Martin Fowler
- Based on MVC
- Targets modern UI development platforms (WPF and Silverlight)
- ViewModel is responsible for exposing data objects from Model in such a way that those objects are easily managed and consumed. It handles most of the View's display logic
- Designed to make use of specific functions in WPF to better facilitate the separation of the View layer development from the rest of the pattern by removing the code behind the View layer.
- Allows designers to focus on UI needs rather than programming or business logic

MVC in HFDP

Great code example, which I am NOT going to force you to follow through, but you really should read and understand. Not only will you learn about MVC in action, you may learn about some components of Java you haven't had an opportunity to work with yet.

:-)