MVC Model/View/Controller

This is an architectural pattern which splits the code into 3 parts, model, view and controller.

Model

This stores application data as well as handle all the business logic, rules (e.g. who can access a student's transcript) for the system. It is also responsible for validation (can also be in controller), data persistence, application state (for example keeping a reference to a user's shopping cart in an online session).

View

This part of the code renders the data to a format that can be presented to the user, for example for a web application the view code would generate HTML code that was sent to the user's browser.

Controller

This part of the code interprets user input (such as mouse clicks or keyboard input) and sends it to the model. For GUI interfaces, each on screen widget capable of input, has typically an associated piece of controller code.

MVC benefits

Code security
Makes it easier to change the user interface, (the VC) code without having the modify the model code. This is because the VC code can be changed without exposing the critical business logic to change.

Multiple interfaces
The MVC architecture makes it simpler to support multiple interfaces to the application. The module can present a standard public interface that the different VCs can be connected into. Using MVC the software can be developed by two separate teams, one's with skills in GUI and others in skills in working with database technologies and the application area.

MVC web-mail work flow example

The user goes to the home URL for the mail service and the VIEW sends a user a login page in Javascript/HTML to the user's browser. The view code can render different versions of the HTML or use CSS to work with different sizes screens etc. The user types in their user/name and password and presses the submit button. The controller software (Javasc ript+servlet) intercepts the submitted Form page, constructs a Login command object and sends it to the Model command handling code. The Model checks the user's credentials against the database and then makes a request to the view If the user's credentials fail, the VIEW is requested to generate a failed login page. If the login is successful, the model code retrieves the user's current email inbox from the database and makes a request to the View to display this data to the user. The VIEW generates the HTML which is sent back to the user’s browser.