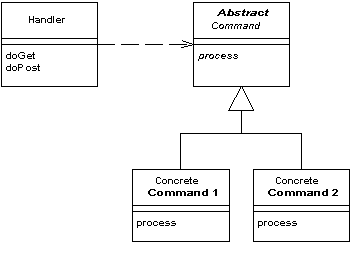
**Front Controller vs Application Controller**

While trying to study the Application Control Pattern I kept running into the Front Controller Pattern. While these two patterns seemed to be the same thing I saw enough that showed me that they are different. So I wanted to look at each so I could see the differences and when/why you would use one or the other.

**Front Controller**

*A controller that handles all requests for a Web site.*



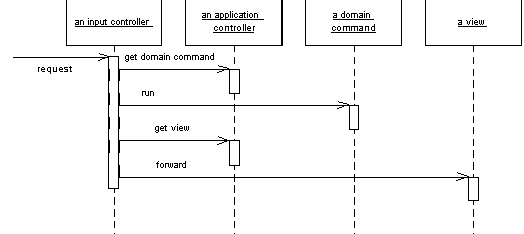
In a complex Web site there are many similar things you need to do when handling a request. These things include security, internationalization, and providing particular views for certain users. If the input controller behavior is scattered across multiple objects, much of this behavior can end up duplicated. Also, it's difficult to change behavior at runtime.

The Front Controller consolidates all request handling by channeling requests through a single handler object. This object can carry out common behavior, which can be modified at runtime with decorators. The handler then dispatches to command objects for behavior particular to a request.

The front controller may be implemented as a Java object, or as a script in a script language like PHP, Python or Ruby that is called on every request of a web session. This script, for example an index.php, would handle all tasks that are common to the application or the framework, such as session handling, caching, and input filtering. Based on the specific request, it would then instantiate further objects and call methods to handle the particular task required.

**Application Controller**

*A centralized point for handling screen navigation and the flow of an application.*



Some applications contain a significant amount of logic about the screens to use at different points, which may involve invoking certain screens at certain times in an application. This is the wizard style of interaction, where the user is led through a series of screens in a certain order. In other cases we may see screens that are only brought in under certain conditions, or choices between different screens that depend on earlier input.

To some degree the various Model View Controller input controllers can make some of these decisions, but as an application gets more complex this can lead to duplicated code as several controllers for different screens need to know what to do in a certain situation.

You can remove this duplication by placing all the flow logic in an Application Controller. Input controllers then ask the Application Controller for the appropriate commands for execution against a model and the correct view to use depending on the application context.