Front Controller

Context

The presentation-tier request handling mechanism must control and coordinate processing of each user across multiple requests. Such control mechanisms may be managed in either a centralized or decentralized manner.

Problem

The system requires a centralized access point for presentation-tier request handling to support the integration of system services, content retrieval, view management, and navigation. When the user accesses the view directly without going through a centralized mechanism, two problems may occur:

- Each view is required to provide its own system services, often resulting in duplicate code.
- View navigation is left to the views. This may result in commingled view content and view navigation.

Additionally, distributed control is more difficult to maintain, since changes will often need to be made in numerous places.

Forces

- Common system services processing completes per request. For example, the security service completes authentication and authorization checks.
- Logic that is best handled in one central location is instead replicated within numerous views.
- Decision points exist with respect to the retrieval and manipulation of data.
- Multiple views are used to respond to similar business requests.
- A centralized point of contact for handling a request may be useful, for example, to control and log a user's progress through the site.
- System services and view management logic are relatively sophisticated.

Solution

Use a controller as the initial point of contact for handling a request. The controller manages the handling of the request, including invoking security services such as authentication and authorization, delegating business processing, managing the choice of an appropriate view, handling errors, and managing the selection of content creation strategies.

The controller provides a centralized entry point that controls and manages Web request handling. By centralizing decision points and controls, the controller also helps reduce the amount of Java code, called *scriptlets,* embedded in the JSP.

Centralizing control in the controller and reducing business logic in the view promotes code reuse across requests. It is a preferable approach to the alternative—embedding code in multiple views—because that approach may lead to a more error-prone, reuse-by-copy- and-paste environment.

Typically, a controller coordinates with a dispatcher component. Dispatchers are responsible for view management and navigation. Thus, a dispatcher chooses the next view for the user and vectors control to the resource. Dispatchers may be encapsulated within the controller directly or can be extracted into a separate component.

While the Front Controller pattern suggests centralizing the handling of all requests, it does not limit the number of handlers in the system, as does a Singleton. An application may use multiple controllers in a system, each mapping to a set of distinct services.

Structure

Figure 1.1 represents the Front Controller class diagram pattern.

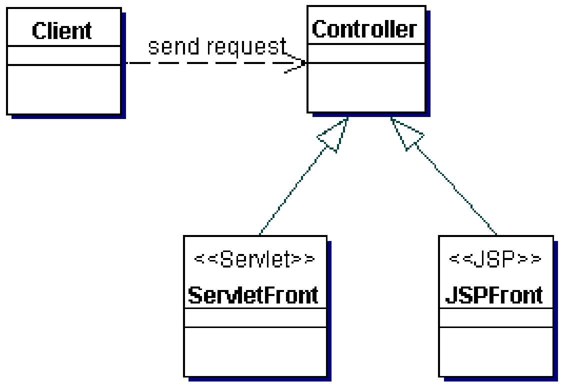

Figure 1.1 Front Controller class diagram.

Participants and Responsibilities

Figure 1.2 shows the sequence diagram representing the Front Controller pattern. It depicts how the controller handles a request.

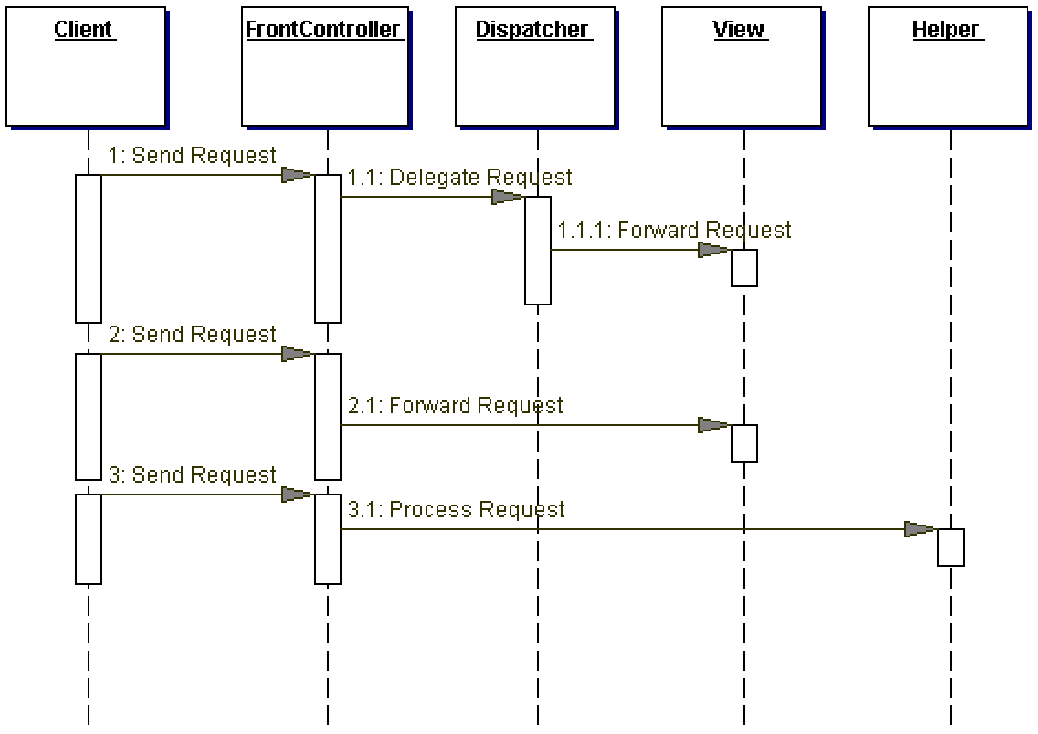

Figure 1.2 Front Controller sequence diagram.

Controller

The controller is the initial contact point for handling all requests in the system. The controller may delegate to a helper to complete authentication and authorization of a user or to initiate contact retrieval.

Dispatcher

A dispatcher is responsible for view management and navigation, managing the choice of the next view to present to the user, and providing the mechanism for vectoring control to this resource.

A dispatcher can be encapsulated within a controller or can be a separate component working in coordination. The dispatcher provides either a static dispatching to the view or a more sophisticated dynamic dispatching mechanism.

The dispatcher uses the RequestDispatcher object (supported in the servlet specification) and encapsulates some additional processing.

Helper

A helper is responsible for helping a view or controller complete its processing. Thus, helpers have numerous responsibilities, including gathering data required by the view and storing this intermediate model, in which case the helper is sometimes referred to as a value bean. Additionally, helpers may adapt this data model for use by the view. Helpers can service requests for data from the view by simply providing access to the raw data or by formatting the data as Web content.

A view may work with any number of helpers, which are typically implemented as JavaBeans (JSP 1.0+) and custom tags (JSP 1.1+). Additionally, a helper may represent a Command object, a delegate, or an XSL Transformer, which is used in combination with a stylesheet to adapt and convert the model into the appropriate form.

View

A view represents and displays information to the client. The view retrieves information from a model. Helpers support views by encapsulating and adapting the underlying data model for use in the display.