Service to Worker

Context

The system controls flow of execution and access to business data, from which it creates presentation content.

Note

The Service to Worker pattern, like the Dispatcher View pattern, describes a common combination of other patterns from the catalog. Both of these macro patterns describe the combination of a controller and dispatcher with views and helpers. While describing this common structure, they emphasize related but different usage patterns.

Problem

The problem is a combination of the problems solved by the Front Controller and View Helper patterns in the presentation tier. There is no centralized component for managing access control, content retrieval, or view management, and there is duplicate control code scattered throughout various views. Additionally, business logic and presentation formatting logic are intermingled within these views, making the system less flexible, less reusable, and generally less resilient to change.

Intermingling business logic with view processing also reduces modularity and provides a poor separation of roles among Web production and software development teams.

Forces

- Authentication and authorization checks are completed per request.
- Scriptlet code within views should be minimized.
- Business logic should be encapsulated in components other than the view.
- Control flow is relatively complex and based on values from dynamic content.
- View management logic is relatively sophisticated, with multiple views potentially mapping to the same request.

Solution

Combine a controller and dispatcher with views and helpers to handle client requests and prepare a dynamic presentation as the response. Controllers delegate content retrieval to helpers, which manage the population of the intermediate model for the view. A dispatcher is responsible for view management and navigation and can be encapsulated either within a controller or a separate component.

Service to Worker describes the combination of the Front Controller and View Helper patterns with a dispatcher component.

While this pattern and the Dispatcher View pattern describe a similar structure, the two patterns suggest a different division of labor among the components. In Service to Worker, the controller and the dispatcher have more responsibilities.

Since the Service to Worker and Dispatcher View patterns represent a common combination of other patterns from the catalog, each warrants its own name to promote efficient communication among developers. Unlike the Service to Worker pattern, the Dispatcher View pattern suggests deferring content retrieval to the time of view processing.

In the Dispatcher View pattern, the dispatcher typically plays a limited to moderate role in view management. In the Service to Worker pattern, the dispatcher typically plays a moderate to large role in view management.

A limited role for the dispatcher occurs when no outside resources are utilized in order to choose the view. The information encapsulated in the request is sufficient to determine the view to dispatch the request. For example,

http://some.server.com/servlet/Controller?next=login.jsp

The sole responsibility of the dispatcher component in this case is to dispatch to the view `login.jsp`.

An example of the dispatcher playing a moderate role is the case where the client submits a request directly to a controller with a query parameter that describes an action to be completed:

http://some.server.com/servlet/Controller?action=login

The responsibility of the dispatcher component here is to translate the logical name `login` into the resource name of an appropriate view, such as `login.jsp`, and dispatch to that view. To accomplish this translation, the dispatcher may access resources such as an XML configuration file that specifies the appropriate view to display.

On the other hand, in the Service to Worker pattern, the dispatcher might be more sophisticated. The dispatcher may invoke a business service to determine the appropriate view to display.

The shared structure of Service to Worker and Dispatcher View consists of a controller working with a dispatcher, views, and helpers.

Structure

The class diagram in Figure 1.1 represents the Service to Worker pattern.

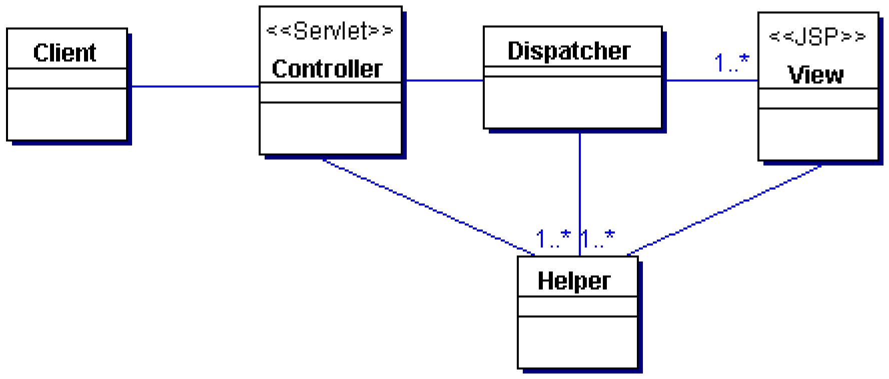

Figure 1.1 Service to Worker class diagram.

Participants and Responsibilities

Figure 1.2 shows the sequence diagram that represents the Service to Worker pattern.

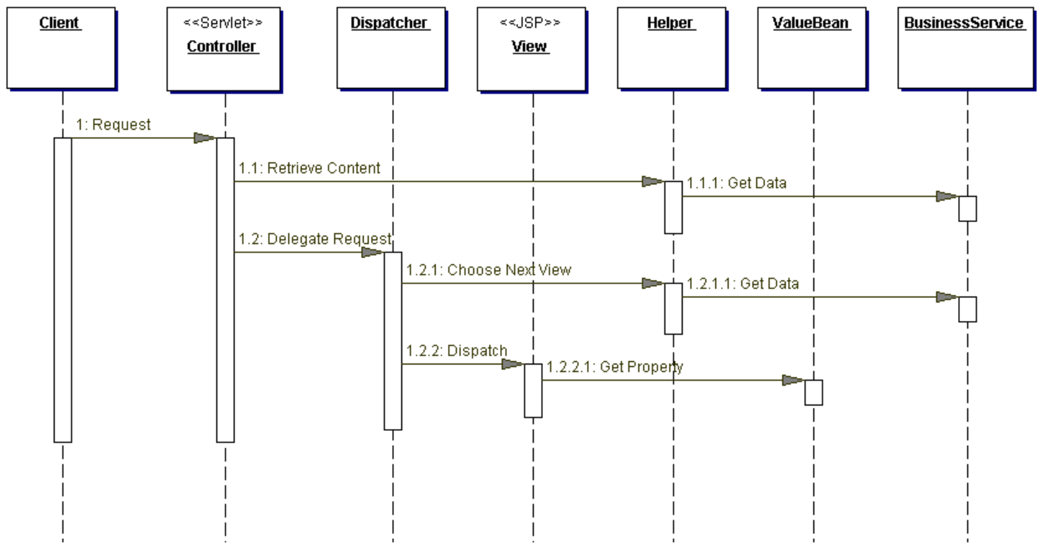

Figure 1.2 Service to Worker sequence diagram.

As stated, Service to Worker and Dispatcher View represent a similar structure. The main difference is that Service to Worker describes architectures with more behavior "up front" in the controller and dispatcher, while Dispatcher View describes architectures with more behavior moved back to the time of view processing. Thus, the two patterns suggest a continuum, where behavior is either encapsulated closer to the front or moved farther back in the process flow.

Controller

The controller is typically the initial contact point for handling a request. It works with a dispatcher to complete view management and navigation. The controller manages authentication, authorization, content retrieval, validation, and other aspects of request handling. It delegates to helpers to complete portions of this work.

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 it can be a separate component working in coordination with the controller. The dispatcher can provide static dispatching to the view or it may provide a more sophisticated dynamic dispatching mechanism.

The dispatcher uses the RequestDispatcher object (supported in the servlet specification), but it also typically encapsulates some additional processing. The more responsibilities that this component encapsulates, the more it fits into the Service to Worker pattern. Conversely, when the dispatcher plays a more limited role, it fits more closely into the Dispatcher View pattern.

View

A View represents and displays information to the client. The information that is used in a display is retrieved from a model. Helpers support views by encapsulating and adapting a model for use in a display.

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 or a delegate.

ValueBean

A value bean is another name for a helper that is responsible for holding intermediate model state for use by a view. A typical case, as shown in the sequence diagram in Figure 7.12, has the business service returning a value bean in response to a request. In this case, ValueBean fulfills the role of a Value Object.

BusinessService

The business service is a role that is fulfilled by the service the client is seeking to access. Typically, the business service is accessed via a Business delegate. The business delegate's role is to provide control and protection for the business service.