Controller Layer

The Controller layer of an MVC application is responsible for creating the abstraction between the Model and View layers. That is, the Controller layer acts as a liaison between the Model and View layers, separating one from the other. This abstraction is the foundation of the MVC design pattern. Recall that the Model contains the application's core, including the business logic and data access code, and the View contains the interface to the application. The Controller layer stands between these two layers, allowing them to change independently of one another. With this architecture, the Model (or core application code) is not limited to a singular use. This is a key advantage of MVC.

In MVC Web applications, the Controller layer serves as the central point of access to the application. All requests to an MVC Web application flow through the controller. By doing so, the Controller layer provides processing common to all requests, such as security, caching, logging, and so on. Most importantly, though, having all requests flow through the Controller allows the Controller to have complete autonomy over how requests are mapped to business processing and how the proper View is selected, based on the outcome of the business processing.

Struts and the Controller Layer

Struts provides a robust Controller layer implementation that has been designed from the ground up to be extensible. At its core is the Controller servlet, **ActionServlet**, which is responsible for initializing a Struts application's configuration from the Struts configuration file and for receiving all incoming requests to the application. Upon receiving a request, **ActionServlet** delegates its processing to the **RequestProcessor** class.
The **RequestProcessor** class processes all aspects of the request, including selecting the Form Bean associated with the request, populating the Form Bean with data, validating the Form Bean, and then selecting the correct **Action** class to execute for the request.
The **Action** class is where the Struts framework ends and your application code begins. **Action** classes provide the glue between the View and Model layers. Figure 5-1 illustrates the Controller layer lifecycle.

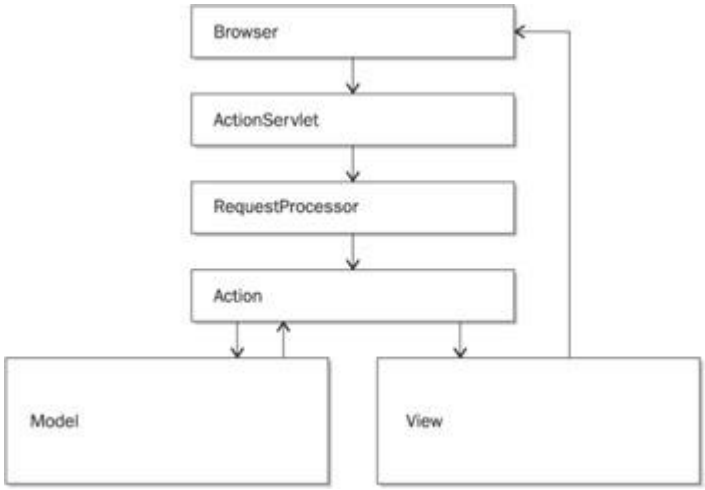

Figure 5-1: The Controller layer lifecycle