Documentation Application VIF

Servlet framework

Benno Luthiger
Bachmattstr. 39
CH-8048 Zürich

Telefon +41 1 433 18 20
benno.luthiger@id.ethz.ch

Version 1.0
Zürich, 22. Juni 2002

Table of Content

Abstract

This document describes the servlet framework used for the application VIF.

1. Separation of content and presentation

There are two main tasks that the servlet framework aims to satisfy. First the framework provides the functionality to display data and second this task is met be making a complete separation between content and presentation. The latter goal is reached by using XSL transformation.

Responding to a servlet request therefore consists of two main steps. First interpret the request and prepare the appropriate information to XML. Second transform this data with the help of XSL to HTML and write the processed data to the output stream. Most of the tasks needed for these steps are encapsulated in this framework. Basically the only thing an application developer has to know is how the requested information has to prepared and by which transformation this information has to be processed.

2. Basic interaction

Figure 1 shows the sequence of a basic interaction within the servlet framework.

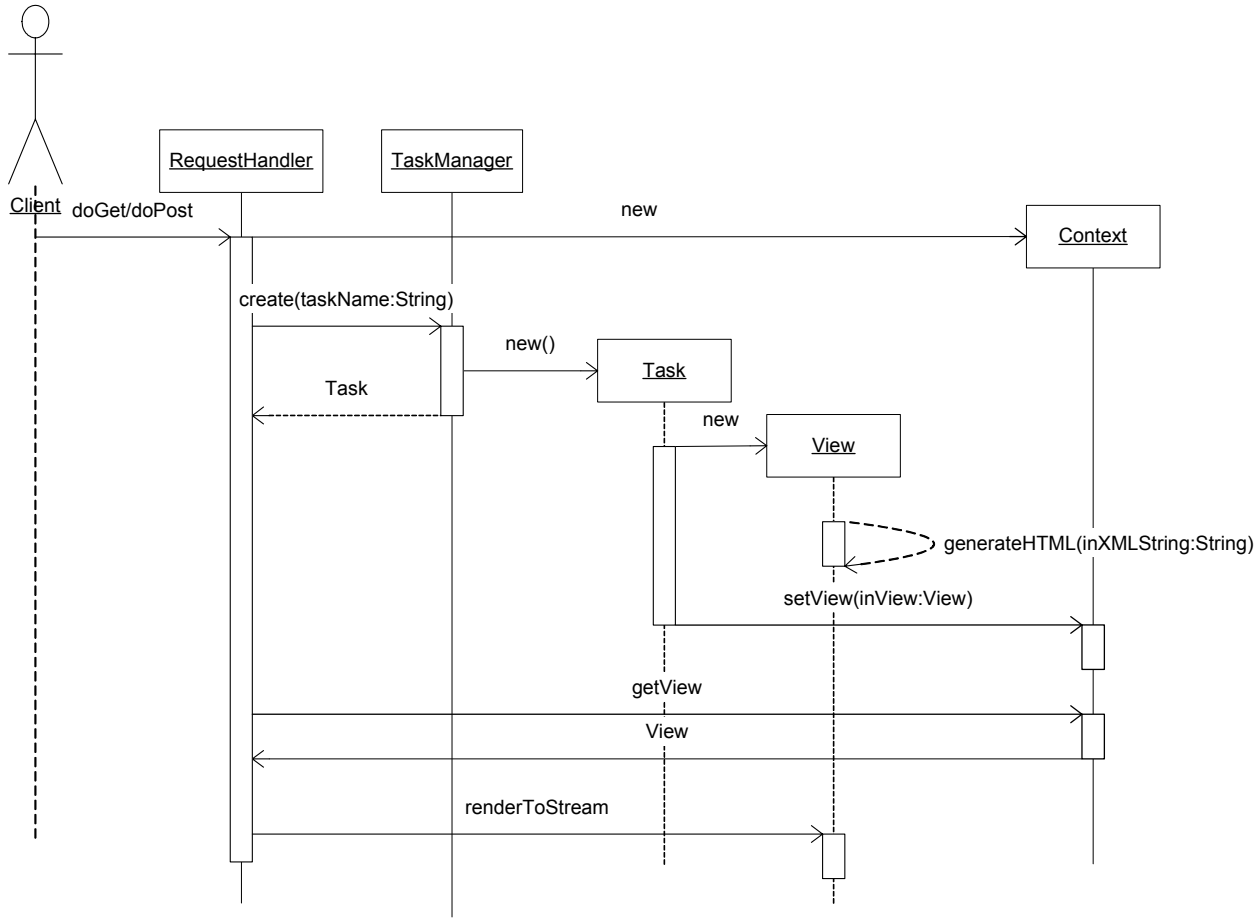

Figure 1: Sequence diagram of servlet interaction

Every request is lead to the `RequestHandler` which acts as single entry point. For each user logged to the application a `Context` is created, thus making state information available for a whole user session. The `RequestHandler` looks for request type information sent with the http request and calls the `TaskManager` for the appropriate `Task`. The `Task` then is responsible for preparing the expected information, e.g. by making a database call. Having collected the data, the `Task` calls its associated `View` and passes the data over. Each `View` knows the stylesheet which can transform the information handed over. The view prepares the data to an XML and calls the stylesheet to carry out the transformation. The result is stored to the context and a notification is given to the `RequestHandler`. The `RequestHandler` takes the resulting view and renders it to the output stream.

3. Components of the servlet framework

3.1. RequestHandler

The `RequestHandler` class is derived from `http.HttpServlet`. All request coming from a client web browsers are lead to this class. The duties of the `RequestHandler` are: checking and preparing of the request, generation and starting of the appropriate `Task`, rendering the generated `View` to the response and sending it back to the client browser.

3.2. Context

Although the http protocol is stateless, the javax servlet api provides functionality for session handling. The servlet framework makes use of this session handling. By use of the `Context` class, this session functionality is available for the application. The `Context` serves as container. Each value which has to be available for further interactions has to be stored to the `Context`.

3.3. TaskManager

Each http request identifies the task that is assigned to process the information handed over with the parameter `requestType`. The `RequestHandler` calls the `TaskManager` with the value of `requestType`. The `TaskManager` contains a mapping table that maps each request type to an appropriate `Task`. The `TaskManager` then instantiates the identified `Task` and hands it back the `RequestHandler`.

3.4. Task

A `Task` in the servlet framework is an ordinary object instance. The main purpose of a `Task` class is to process the information passed by the request and to prepare the data that has to be sent back to the client browser. Each `Task` knows the `View` that can process the prepared information contingent to the state of this information.

3.5. View

Each `View` knows the procedure how to prepare the data passed over for that a web browser can display it. The preferred way this is done is by invoking a XSL transformation. The `View`, therefore, knows the XSL stylesheet responsible for such transformation. The `View` calls the transformation on the passed data and stores the result in the `Context`. The html page sent back to the client browser is a container and may consist on various “views”, i.e. it may contain various outputs generated by different instances of `View`.

4. Cookbook

For to use the servlet framework you have to do the following steps:

1. Create a `RequestHandler` for your application. This class has to be derived from `AbstractRequestHandler`.
2. Create a `Context` for your application. This class has to subclass `AbstractContext`. Update your `RequestHandler` so that its `getContextClassName()` method returns the fully qualified class name of the applications `Context` class.
3. Create a `TaskManager` for your application. This class has to be derived from `AbstractTaskManager`. Update your `RequestHandler` so that its `getTaskManager()` method returns the singleton instance of the applications `TaskManager`. Implement the `TaskManager`'s `initialize()` method so that the mapping table is initialized.

These first steps have to be done only once for an application.

4. For each task (e.g. use case) the application has to accomplish create a `Task` class. These classes have to subclass `AbstractTask`. Update the `TaskManagers` mapping table so that the fully qualified class name of the `Task` class is mapped with the appropriate request type. Implement the `Taks`'s `run()` method so that the class can accomplish its task.
5. For each view (i.e. separable information sent back to the client browser) create a `View` class. These classes have to be derived from an abstract view class of the framework, e.g. from `AbstractXSLView`. Implement the `View`'s `getXMLName()` method so that it returns the fully qualified path and name of the XSL file used to transform the data passed to the `View`.