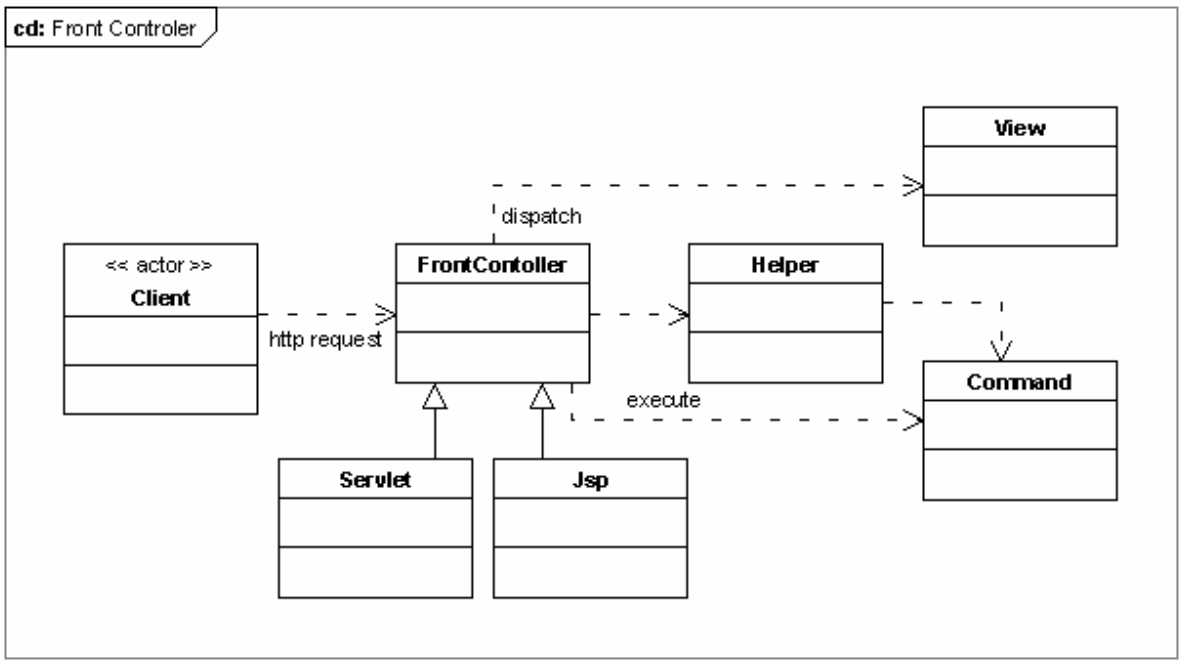
Front Controller Tasarım Şablonu

KurumsalJava.com

Özcan Acar
Bilgisayar Mühendisi
http://www.ozcanacar.com

Web aplikasyonlarında Front Controller tasarım şablonu ile sisteme yöneltilen tüm istekler (request) merkezi bir yerde toplanarak işlem görürler.



Front Controller ile, yönlendirme ve işlem yapma fonksiyonlarının birden fazla view (bir JSP sayfası olabilir) elementine dağıtılması önlenmiş olur.Tüm view elementleri yönlendirme ve işlem yapma fonksiyonlarını ortak kullanırlar. Böylece Front Controller tasarım şablonunun kullanıldığı bir proje bakımı ve geliştirilmesi daha kolay bir hale gelir. Ayrıca Front Controller ile gösterim ve navigasyon fonksiyonları birbirinden ayrıldığı için, gösterim katmanını etkilemeden navigasyon idaresi değiştirilebilir yada tamamen yenilenebilir.

Front Controller implementasyonunda Servlet ya da JSP teknolojisi kullanilabilir. Bir sonraki örnekte Servlet teknolojisi kullanilarak Front Controller implemente edilmekteidr.

```
package org.javatasarim.otelrezervasyon.presentation.ui;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.javatasarim.otelrezervasyon.presentation.command.Command;
import org.javatasarim.otelrezervasyon.presentation.helper.RequestHelper;
import org.javatasarim.otelrezervasyon.util.Logger;


/**
 * FrontController tasarim şablonu örnegi.
 * Http üzerinden gelen tüm request'ler
 * su sinif tarafindan islem görür.
 *
 * Controller sinifi gerekli command nesnesi bularak
 * islemi bu nesneye devreder ve netice göre gerekli
 * jsp sayfasina yönlendirme yapar.
 *
 * @author Oezcan Acar
 *
```

```
 */
public class FrontController extends HttpServlet
{
	private static final long serialVersionUID = 1L;


	/**
	 * FrontController sinifina gelen tüm request'ler bu
	 * metod tarafindan işlem görür.
	 * @param request HttpServletRequest
	 * @param response HttpServletResponse
	 */
	public void handle(HttpServletRequest request,
			HttpServletResponse response)
	{
		Logger.instance(this).debug("handle()");
		String nextPage = "";
		try
		{
			RequestHelper helper = new HotelRequestHelper(request,
					response);
			Command command = helper.getCommand();
			nextPage = command.execute(helper);
			dispatch(request, response, nextPage);
		}
		// CommandException ve IOExcepton olusabilir.
		// Bu durumda error.jsp sayfasina yönlendiriyoruz.
		catch (Exception e)
		{
			e.printStackTrace();
			try
			{
				dispatch(request, response, "/error.jsp");
			}
			catch (Exception e1)
			{
				e1.printStackTrace();
			}
		}
	}


	/**
	 * Jsp sayfalari arasinda yönlendirme yapmak için kullanılır.
	 * @param request
	 * @param response
	 * @param page
	 * @throws ServletException
	 * @throws IOException
	 */
	private void dispatch(HttpServletRequest request,
			HttpServletResponse response, String page) throws
			ServletException, IOException
	{
		Logger.instance(this).debug("dispatch()");
		RequestDispatcher dispatcher = getServletContext()
		.getRequestDispatcher(page);
		dispatcher.forward(request, response);
	}

	/**
	 * FrontController bir servlet sinifi oldugu için GET
```

```
         * metodu ile gelen tüm request'ler bu metod tarafindan
         * islem görür.
         */
        public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
        {
              Logger.instance(this).debug("doGet()");
              handle(request, response);
        }


        /**
         * FrontController bir servlet sinifi oldugu için POST
         * metodu ile gelen tüm
         * request'ler bu metod tarafindan islem görür.
         */
        public void doPost(HttpServletRequest request,
                    HttpServletResponse response)
        {
              Logger.instance(this).debug("doPost()");
              handle(request, response);
        }
}
```

FrontController isminde bir sınıf tanımlıyoruz. Bu sınıf **HttpServlet** sınıfını genişlettiği (extends HttpServlet) için bir Servlet sınıfı haline gelir. Bir servlet sınıfının kullanıcıdan gelen GET ve POST metodlarına cevap vermek üzere *doGet()* ve *doPost()* isimlerinde iki metoda sahip olması gerekmektedir. **FronController** sınıfında *doGet()* ve *doPost()* metodları içinde *handle(request, response)* metodunu kullanarak gelen isteği (http request) merkezi bir metoda yönlendiriyoruz. Bu Front Controller sınıfı, bir **Helper** sınıfı yardımı ile üyenin isteğine cevap vermek üzere gerekli Command[1] nesnesini edinir. Command bünyesinde gerekli business metodları çalıştırılır ve oluşan sonucun gösterimi için bir sonraki sayfa (view) geri verilir. **FrontController**, sahip olduğu *dispatch()* metodu yardımı ile, kontrolü **RequestDispatcher** sınıfına verir ve bir sonraki JSP sayfasının gösterilmesini sağlar.

Front Controller tasarım şablonunun sağladığı diğer bir avantaj da merkezi Exception Handling yapılmasını kolaylaştırıyor olmasıdır. *handle()* metodunda yer alan bir try-catch bloğu ile alt katmanlarda oluşan hatalar yakalanır ve örneğin log4j[2] ile bir log dosyasına yazılır ya da bir bilgibankasına daha sonra incelenmek üzere eklenir.

FrontController sınıfını bir web aplikasyon içinde kullanabilmek için aşağıdaki şekilde web.xml[3] kaydının yapılması gerekmektedir.

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE web-app PUBLIC '-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN'
'http://java.sun.com/JEE/dtds/web-app_2_3.dtd'>

<web-app>
```

[1] Command tasarım sablonu hakkında geniş bilgi için Command tasarım sablonu bölümüne bakınız.
[2] Logging yapmak için kullanılan open source program. Bakınız: http://logging.apache.org/log4j
[3] Web aplikasyonlarında kullanılan konfigürasyon dosyasıdır.

```
  <servlet>
    <servlet-name>FrontController</servlet-name>
    <servlet-class>

      org.javatasarim.otelrezervasyon.presentation.controller.FrontContro
ller
    </servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>FrontController</servlet-name>
    <url-pattern>/FrontController/*</url-pattern>
  </servlet-mapping>


  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

web.xml içinde <servlet> tagını kullanarak **FrontController** isminde bir Servlet tanımlıyoruz. Bu servlet http://localhost/FrontController adresinden erişilebilir hale gelir. <servlet-mapping> tagı ile http://localhost/FrontController adresine yönlendirilmiş tüm istekler (request) FrontConller sınıfına iletilir. GET metodu ile gelen istekler **FrontController**.*doGet()*, POST metodu ile gelen istekler **FrontController**.*doPost()* metoduna yönlendirilir. Aşağıda yer alan html formunda POST metodu kullanılmaktadır. Ara butonuna tıklandığında bu istek (request) */FrontController* sınıfına iletilir.

```
<form name="form1" method="post" action="FrontController">
<table width="300" border="0" cellspacing="0" cellpadding="2">
   <tr>
      <td>
         <div align="left">Sehir:</div>
      </td>
      <td><input type="text" name="city" size="15" class=text10
                 maxlength="50" value="">
      </td>
   </tr>
   <tr>
      <td>
         <input type="submit" name="Submit2" value="   Ara   "
                 class=button>
     </td>
   </tr>
   <tr>
      <td>
         <input type="hidden" name="action" value="ara">
      </td>
      <td> </td>
   </tr>
</table>
</form>
```

Front Controller tasarım şablonu ne zaman kullanılır?

- Sisteme ulaşan isteklerin (http request) merkezi bir yerde toplanıp, işlenmesi gerektiği durumlarda Front Controller kullanılır.
- JSP sayfalarında navigasyon ve diğer işlemler için oluşabilecek kod dublikasyonunu önlemek için Front Controller kullanılır.
- Gösterim katmanını oluşturan JSP sayfalarında yer alan verinin gösterilmesi için gerekli koddan daha fazla kodun yer almasını engellemek için Front Controller kullanılır. JSP sayfaları sadece verileri göstermek için programlanmalıdır. Bir JSP sayfası örneğin kesinlikle JDBC ile bilgibankasına bağlanıp, veri edinmemeli ya da veri üzerinde işlem yapmamalıdır. Bu model komponentlerinin görevidir.
- Güvenlik uygulamaları yapabilmek için merkezi bir giriş noktası gerektiği durumlarda Front Controller kullanılır.

İlişkili tasarım şanlonları:

- Intercepting Filter ve Front Controller tasarım şanlonları veri akışını merkezi bir yerden kontrol etmek amacıyla kullanılır.
- View Helper tasarım şablonu ile kullanılan business metodlar helper sınıflarında biraraya getirilir. Front Controller, helper sınıflarını kullanarak, gerekli işlemlerin yapılmasını sağlar.

EOF (End Of Fun)
Özcan Acar