Dispatcher (Draft - feature under development)

Table of contents

Warning:

The "dispatcher" is new functionality which is still in development phase. That is why it is in the "whiteboard" section of the Forrest distribution. We are working at the moment on moving this plugin from the whiteboard into the core plugins. Further all dispatcher related documents will be moved into the plugin as well. See Status of Themes: Skins and Dispatcher.

1 Introduction

As stated in the Skin documentation the aim of the Forrest skins is to provide many capabilities so that extra skins are not needed. Experience showed that many Forrest users still decided to create a new skin because the default skin did not offer the features that they wanted or they "just" needed extra content in some pages. We introduced skinconf.xml where the user could configure some features of skins but it was up to the skin to support it and did not solve the problem to add page specific extra content. That led us to develop a new concept of creating skins (we called the result "themes") which would be more easily extensible by a user.

2 Dispatcher - advanced separation of concerns

The aim of the "dispatcher" concept is to provide a flexible framework for creating site and page specific layout in different formats from different content through an advanced seperation of concerns.

The dispatcher is a filter that limits the data-model to a minimum by only requesting what the structurer (e.g. common-html.vt.xml) need. This leads to a different URL handling focus - away from document centric. A document can (but does not have to) be behind a certain URL. Like said a structurer can request any given data as input not only a document and the forrest core contracts (like navigation). It may be the main enhancement in comparison to skins that this concept lets you easily extend the default data models provided by forrest.

Since the dispatcher has implemented a fallback concept it makes maintenance of custom themes which are based on forrest core ones very easy and less time consuming. The principal is to override or extend only certain parts (contracts) of the core. This is based on the observation that normally only a small percentage of core skin contracts have been changed. At the same time the new plugin system emerged. Plugins are a way of extending Forrest to satisfy site-specific needs. This includes to provide for plugin specific contracts.

2.1 Structurer - configuration for themes

We developed **the structurer** to let the user decide where to place elements in e.g. html pages. We started this work with the `skinconf.xml` where you could configure certain

elements and their positions. These elements were known under certain names. It was up to the skin designer to support this configuration and the elements.

The work started with grouping elements (the ones from skinconf). We used css-contracts that we added as @attributes e.g. `<div id="content-main"/>` . That made it possible to use the same elements in different skins. For the full list refer to the initial contract list

Around this contracts we developed a configuration Domain Specific Language - called **the structurer**. The **structurer** allows us to define the order in which **forrest:contract**s appear, and also to group them using **forrest:hook**s.

forrest:hooks are containers that are only used for layout reasons. They **do not** add any content nor functionality to the output. They add **only** layout information to the output. Actually e.g. a `<forrest:hook name="layoutId"/>` will be transformed to `<div id="layoutId"/>`

forrest:contracts are functionality or extra content that a theme can use to display the request. Sometimes a contract delivers **format-specific markup**, other times it delivers a **format-independent string**. We decide different kind of contracts, static one (like described in the contract howto), semi static (which offer configuration parameter in the structurer) and dynamic contracts (which offer semi-static configuration and/or requesting the content).

The structurer is also a configuration file for the dispatcher. The new thinking on the dispatcher is that one can include any content from any given business service by dispatching a request against it. In "old fashion" skins and in v1 contracts we assumed a given data model. In the dispatcher there is **no** given data model any more. All data has to be defined in the structurer so that they can be dispatched.

2.2 Contracts - grouped functionality

The result of the leather-dev development was grouped functionality in named containers. We gave those code snippets names (based on their functionality) and called them **contracts**. This naming enabled us to keep the contract separate from the positioning of the code itself. Furthermore, since major parts of the code of skins has never been documented, we started to add for each contract a description and an explanation on how to use this contract. The skinconf.xml gave an excellent source for this documentation effort, since it described most features of the pelt skin.

Contracts are standalone, self explaining, configurable pieces of xsl templates created purely for ease of maintenance.

Since these contracts are working from the input given in the structurer, it works on different input sources. One can pass variables into the contracts that can be used to apply presentation logic in the xsl (like sorting order, ...).

3 Background

The problem with the forrest skins so far has been that even if "only" the design changed (html-skeleton), we still had to write a completely new skin and implement all functionality. Another problem was that the functionality was not easily extensible by a user. We decided to support a standard regarding naming conventions for css elements. This standard has been developed on the OSCOM website, where you can find some more background information.

3.1 Definition of naming conventions

"A naming convention is an attempt to systematize names in a field so they unambiguously convey similar information in a similar manner." wikipedia

3.2 leather-dev

That led to the development of the "leather-dev" skin which established a semantic container approach for div elements. Leather-dev evolved from the "pelt" skin and used almost the same functionality (contracts). We had started to encapsulate functional code into templates, but there have been still in 4 xsl files and without any documentation on what they are doing and how to use them. The problems with leather-dev was pointed out in the mail "status on leather-dev?". The main proplem was to limit users to only one html-skeleton which was way too limiting regarding design. Since we had now grouped functionality in named containers we were ready to start the dispatcher (aka forrest:views).

4 Further information

See the various How-to documents about the dispatcher, starting with the quickstart. It would also be useful to familiarise yourself with some of the terms used in the dispatcher, the Dispatcher Glossary contains some of the terms used.

Apache Forrest 1.0 Specification (Draft, not yet published): `site-author/content/xdocs/TR/2005/WD-forrest10.html`