Avalon for Apache Cocoon (2.1 legacy document)

Table of contents

Table of contents

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Goal

This document tries to provide you with the basic knowledge of Avalon that is necessary to understand Cocoon.

People trying to understand Avalon in depth, will probably not be much helped by this document. But, if you want to understand Cocoon, you have to have a basic grasp of Avalon.

The document also outlines the basic steps for configuring Avalon components within Cocoon.

Much of this document is copied and pasted from original Avalon documentation. However, I hope that by putting all things relevant to Cocoon together in one place I can help you to understand Cocoon more quickly.

For people wishing to learn Avalon in-depth, this is your starting point.

2. Overview

For a mission statement of Apache Avalon, please read the Avalon homepage.

In short, Avalon tries to take design efforts away from server-side programmers by providing a framework that

- provides basic working classes;
- provides interfaces to allow different efforts to be integrated more easily.

3. The classes and interfaces

These classes and interfaces are extensively used by Cocoon:

3.1. ServiceManager

org.apache.avalon.framework.service.ServiceManager

A ServiceManager selects Components based on a role. The contract is that all the Components implement the differing roles and there is one Component per role. If you need to select one of many Components that implement the same role, then you need to use a ServiceSelector. Roles are the full interface name.

To understand roles better let's use the the analogy of a play. There are many different roles in a script. Any actor or actress can play any given part and you get broadly the same results (same phrases spoken, same movements made, etc.), but with each actor the exact nuances of the performance are different.

3.2. Serviceable

org.apache.avalon.framework.service.Serviceable

A Serviceable is a class that needs to connect to software components using a "role" abstraction, thus not depending on particular implementations but on behavioral interfaces. This means, if you need to

use other components in your implementation, you have to implement Serviceable to be able to get other components.

3.3. Configuration

org.apache.avalon.framework.configuration.Configuration

Configuration is an interface encapsulating a configuration node used to retrieve configuration values. This is a "read only" interface preventing applications from modifying their own configurations. The contract surrounding the Configuration is that once it is created, information never changes. The Configuration is built by the ConfigurationBuilder.

3.4. Configurable

org.apache.avalon.framework.configuration.Configurable

Configurable is an interface describing a component which can be configured. This component gets a Configuration object as input.

3.5. ConfigurationBuilder

org.apache.avalon.ConfigurationBuilder

A ConfigurationBuilder builds Configurations.

4. Configuration

Most available Avalon components are configured in the cocoon.xconf.

4.1. Pooling configuration

Avalon now incorporates a couple of modifiers for a Component definition that allows you to control the number of Components in a pool. This is especially helpful in Cocoon where the defaults don't always work well.

The magic attribute is "pool-max". The defaults are:

1. pool-max: 8

What this means is that the pool for the default component initially contains zero instances. If demand exceeds this then the pool will increase, one component at a time, up to 8 instances. Beyond that the pool turns into a factory. That is, new Component instances are created, but destroyed when they are returned. This is a performance issue - but it does manage the number of instances available at one time.

1. Comments

add your comments