PipelineBuilder Service

The **PipelineBuilder Service** is a service used to create pipelines, also known as *filter chains*. An example of this is the Filter and FilterChain interfaces inside the Servlet API.

In this pattern, an existing service is decorated with a filter. The filter will delegate to the service, but has the chance to alter or replace parameters before invoking the method, and can perform operations before returning. This is similar to chain of responsibility, but differs in that there are two interfaces (the service interface and the filter interface) and that each filter invokes the next filter via the service interface. In contrast, in the chain of responsibility, the chain invokes each method, which must return before the next command in the chain is invoked.

Related Articles

- ShadowBuilder Service
- PipelineBuilder Service
- StrategyBuilder Service
- ChainBuilder Service

The service interface and the filter interface are closely related: the filter interface must match the service interface method for method, but each method of the filter interface must have an additional parameter whose type is the service interface. For example, a pipeline that performed string transformations might use the following interfaces:

```
public interface StringTransformService
{
  String transform(String input);
}

public interface StringTransformFilter
{
  String transform(String input, StringTransformService delegate);
}
```

An implementation of the filter might look like:

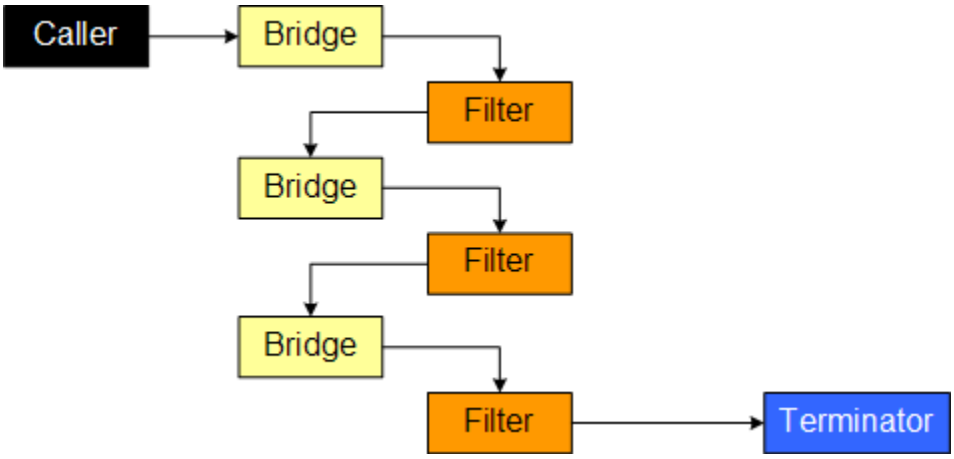
```
public class UpcasePreFilter implements StringTransformFilter
{
  public String transform(String input, StringTransformService delegate)
  {
    return delegate.transform(input.toUpperCase());
  }
}
```

Alternately, the filter could pass input to delegate unchanged, but invoke toUpperCase() on the result:

```
public class UpcasePostFilter implements StringTransformFilter
{
  public String transform(String input, StringTransformService delegate)
  {
    return delegate.transform(input).toUpperCase();
  }
}
```

The PipelineBuilder Service (API) is useful for constructing pipelines. The service is often injected into a service builder method, along with an ordered configuration of services.

What the builder accomplishes is to represent each *filter* in the pipeline as an instance of the *service* interface.



Pipeline Calling Sequence

The bridges are created by the PipelineBuilder service. The terminator must be provided. The bridges and the terminator implement the service interface.

```
public static StringTransformService build(
  @InjectService("PipelineBuilder")
  PipelineBuilder builder,
  List<StringTransformFilter> configuration,
  Logger logger)
  {

    StringTransformService terminator = new StringTransformService()
    {
      public String transform(String input)
      {
        return input;
      }
    };

    return builder.build(logger,
      StringTransformService.class, StringTransformFilter.class,
      configuration,
      terminator);
  }
```

Here, we create the terminator for the pipeline as an inner class instance, and feed that into the builder. The result is a new service that encapsulates the entire pipeline. When there are no filters, this is just the terminator.

ChainBuilder Service IOC ShadowBuilder Service