

```
fun serverStack(systemName: String, app: HttpHandler): HttpHandler =
    logTransactionFilter("IN", systemName)
        then(recordMetricsFilter(systemName))
        then(handleErrorsFilter())
        ■then(app)
fun clientStack(systemName: String): HttpHandler =
    logTransactionFilter("OUT", systemName)
        then(recordMetricsFilter(systemName))
        then(handleErrorsFilter())
```

then(ApacheClient())

## Standardised Server & Clients

By utilising the ability to "stack" Filters, we can build reusable units of behaviour

```
Filter.then(that: Filter) -> Filter
```

```
fun serverStack(systemName: String, app: HttpHandler): HttpHandler =
    logTransactionFilter("IN", systemName)
        then(recordMetricsFilter(systemName))
        then(handleErrorsFilter())
        then(app)

fun clientStack(systemName: String): HttpHandler =
    logTransactionFilter("OUT", systemName)
        then(recordMetricsFilter(systemName))
        then(handleErrorsFilter())
        then(ApacheClient())
```

## Fake Dependencies!

- Leverage Body lenses
- Simple state-based behaviour
- Run in memory or as server
- Easy to simulate failures

