```
val service = new Service[Request, Response] {
  override def apply(request: Request): Future[Response] =
{
    val response = Response(Status.Ok)
    response.content = request.content
    Future(response)
  }
}
```

Concept: Service

“It turns an Request into an (eventual) Response”

ie. it’s a function!

`Service: Request => Future[Response]`

```
val service = new Service[Request, Response] {
  override def apply(request: Request): Future[Response] =
{
    val response = Response(Status.Ok)
    response.content = request.content
    Future(response)
  }
}
```

Concept: Filter

“Provides pre and post processing on an HTTP operation”

ie. it’s a function!

```
Filter[ReqIn, RespIn, ReqOut, RespOut] =
        [ReqIn, Service[ReqOut, RespOut]] => Future[RespIn]
```