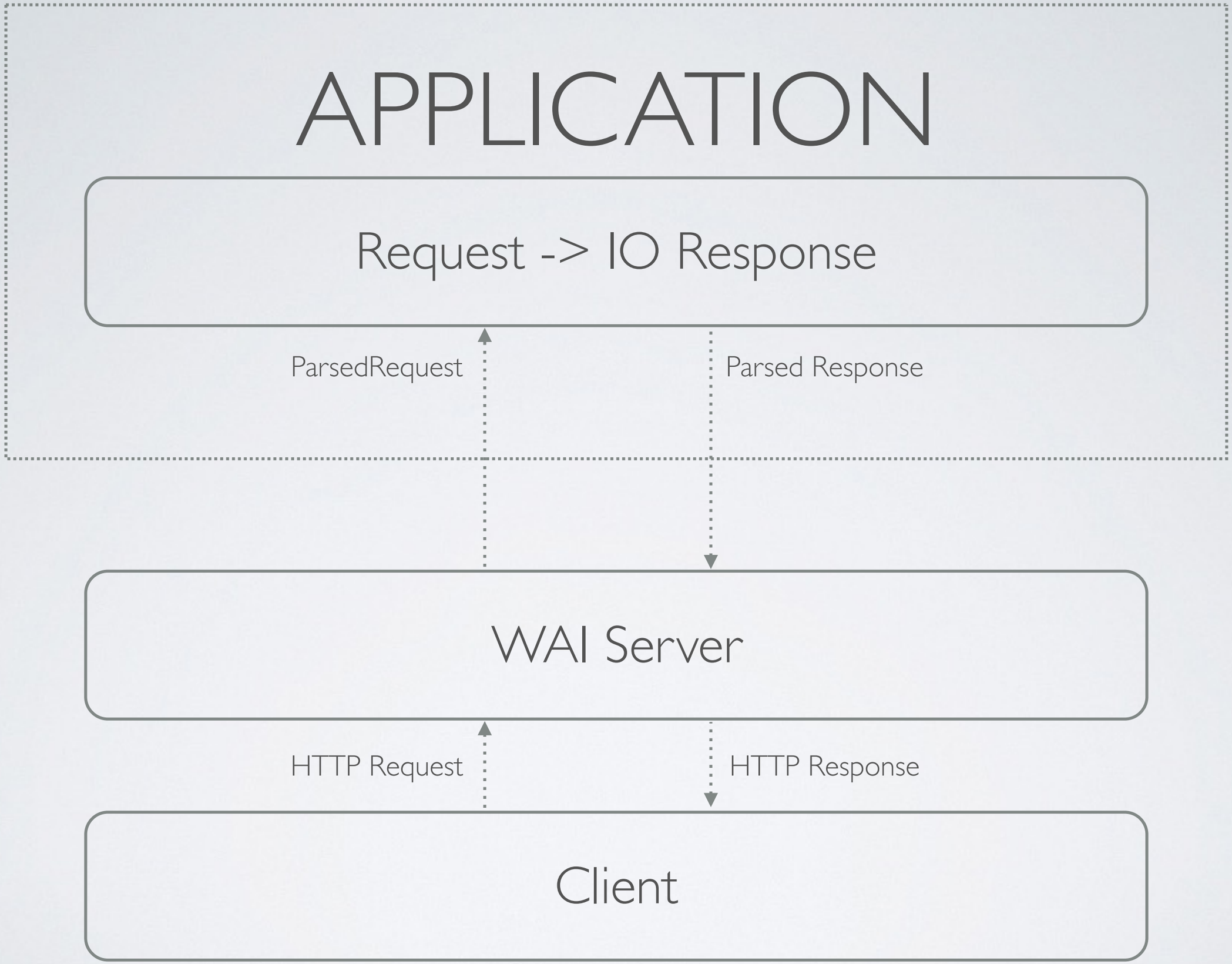
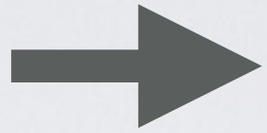
WAI-ROUTES

From Zero to Microservices


APPLICATION
Request -> IO Response
ParsedRequest
Parsed Response
WAI Server
HTTP Request
HTTP Response
Client

APPLICATION

```
type Application =
Request ->
(Response -> IO ResponseReceived)
-> IO ResponseReceived
```

REQUEST  RESPONSE

- Method
- Headers
- Body
- Path Information

- Status Code
- Headers
- Body

ROUTING

```
case getPathInfo req of
"hello" : arg : [] ->
  case getReqMethod req of
    GET  -> getHello  arg req
    POST -> postHello arg req
```

TYPE SAFE ROUTING

```
data MyRoute = HomeR | HelloR Text

renderRoute :: MyRoute -> [Text]
parseRoute  :: [Text]  -> MyRoute
```

TEMPLATE HASKELL

```
mkRoute "MyRoute" [parseRoutes|
  /              HomeR  GET
  /hello/#Text HelloR GET POST
|]
```

FLEXIBLE AND POWERFUL

- Mix and match - Routes, Middleware, and Applications
- Easy Handlers (Ball of StateT)
- Master Site Data Type
- Nested Routes
- Subsites
- Even a No-Template-Haskell, No-GHC-Extensions Mode!

http://github.com/ajnsit/wai-routes/tree/master/examples

MICROSERVICES

- Multiple Small REST Services
- Communication Through Requests
- Type Safe URLs Across Services!

USER SERVICE

- /login {Returns auth token}
- /logout
- /validate {Used internally by album service}

ALBUM SERVICE

- /albums
- /album/#id

{Both methods validate auth token via Users service}

USER SERVICE

```
data UserRoute = UserRoute

mkRouteData "UserRoute" [parseRoutes|
/login    LoginR  POST
/logout   LogoutR POST
/validate ValidateR POST
|]
```

ALBUMS SERVICE

```
data AlbumRoute = AlbumRoute

mkRouteData "AlbumRoute" [parseRoutes|
/albums       AlbumsR GET
/album/#Int AlbumR  GET
|]
```

ALBUMS SERVICE

```
import Users.Data (UserRoute(..))

getAlbumsR :: Handler AlbumRoute
getAlbumsR = runHandlerM $ do
  Just token <- reqHeader “Authorisation”
  auth <- liftIO $ post (showRoute ValidateR) token
  if isValid auth
    then status 403 >> json err
    else json albums
```

FIN