ROUTING

ROUTING

Mapping an HTTP Request to an Operation

ROUTING

Mapping an HTTP Request to a *Function*

ROUTING
IN EXISTDB

CONTROLLER.XQL

CONTROLLER.XQŁ

CONTROLLER.XQ

+somewhat declarative

+single point of entry

+solution agnostic

+efficient handling of static files

CONTROLLER.XQ

- eXist-DB specific
- complex controllers
- handling of data
- authentication and authorization
- hard to document
- easy to do wrong

RESTXQ

RESTXQ

+match URL against patterns

+pick route handler by specificity

+allow only defined routes

+function annotations

+baseX also implements this standard

+spec and documentation available

RESTXQ

- limited tooling
- one handler per output
- handlers with many parameters
- route matching issue
- decent documentation *CAN* be generated
- authentication and authorization

Research &
Failed Attempts

Some More Research & a Proof of Concept

A First Working Prototype

OAS v3

allows the **description** of a remote **API**
accessible through **HTTP**

OPEN API SPEC V3

- declarative
- documentation first

```
{
  "openapi": "3.0.0",
  "info": {
      "version": "1.0.0",
      "title": "My API",
      "description": "Play a game"
  },
  "paths": {
    "/play": {}
  }
}
```

OPEN API SPEC V3

- vendor neutral
- existing tooling

ROASTER
router

TO BUILD ON

REQUEST

ROUTE

/THINGS/{ID}

```
{
  "/play": {
      "get": {
        "operationId": "app:ping-pong",
        "parameters": {
            "ping": { "in": "query", "type": "boolean" }
        }
      }
  }
}
```

PARAMETERS

Must be declared otherwise the router will not receive it

Where?

- path
- query
- cookie
- body
- header

Type?

- boolean
- number
- date

BODY

```
{
  "post": {
    "operationId": "auth:login",
    "requestBody": {
      "required": true,
      "content": {}
    },
    "responses": { "200": {}, "401": {} }
  }
}
```

```
{
    "multipart/form-data": {
        "schema": {
            "properties": {
                "user": { "type": "string" },
                "password": {
                    "type": "string",
                    "format": "password"
                }
            }
        }
    }
}
```

ROUTE HANDLER

```
function(map(xs:string, *)) as item()*
```

```
declare function app:ping-pong (
    $request as map(xs:string, *)
) as item()* {
  if ($request?parameters?ping)
  then "pong"
  else "ping"
};
```

RESPONSE

```
(:~
 : random number determined
 : by a regular throw of a dice
 :)
declare function my:random ($request) {
  4
};
```

```
(: Make a scene :)
declare function my:drama ($request) {
  error(
    $errors:BAD_REQUEST,
    "I CAN'T work like that",
    $request (: pass additional data :)
  )
};
```

```
(: artisanal, custom-made response :)
declare function my:upload ($request) {
  roaster:response(
    201,
    xmldb:store(
      "/db/apps/roasted/uploads",
      $request?parameters?path,
      $request?body
    )
  )
};
```

```
(: look elsewhere :)
declare function my:redirect ($request) {
  roaster:response(
    301,
    "text/plain",
    "redirecting",
    map { "Location": "?ping" }
  )
};
```

```
(: look elsewhere :)
declare function my:redirect ($request) {
  roaster:response(
    301,                                  (: CODE :)
    "text/plain",                         (: TYPE :)
    "redirecting",                        (: BODY :)
    map { "Location": "?ping" } (: HEADERS :)
  )
};
```

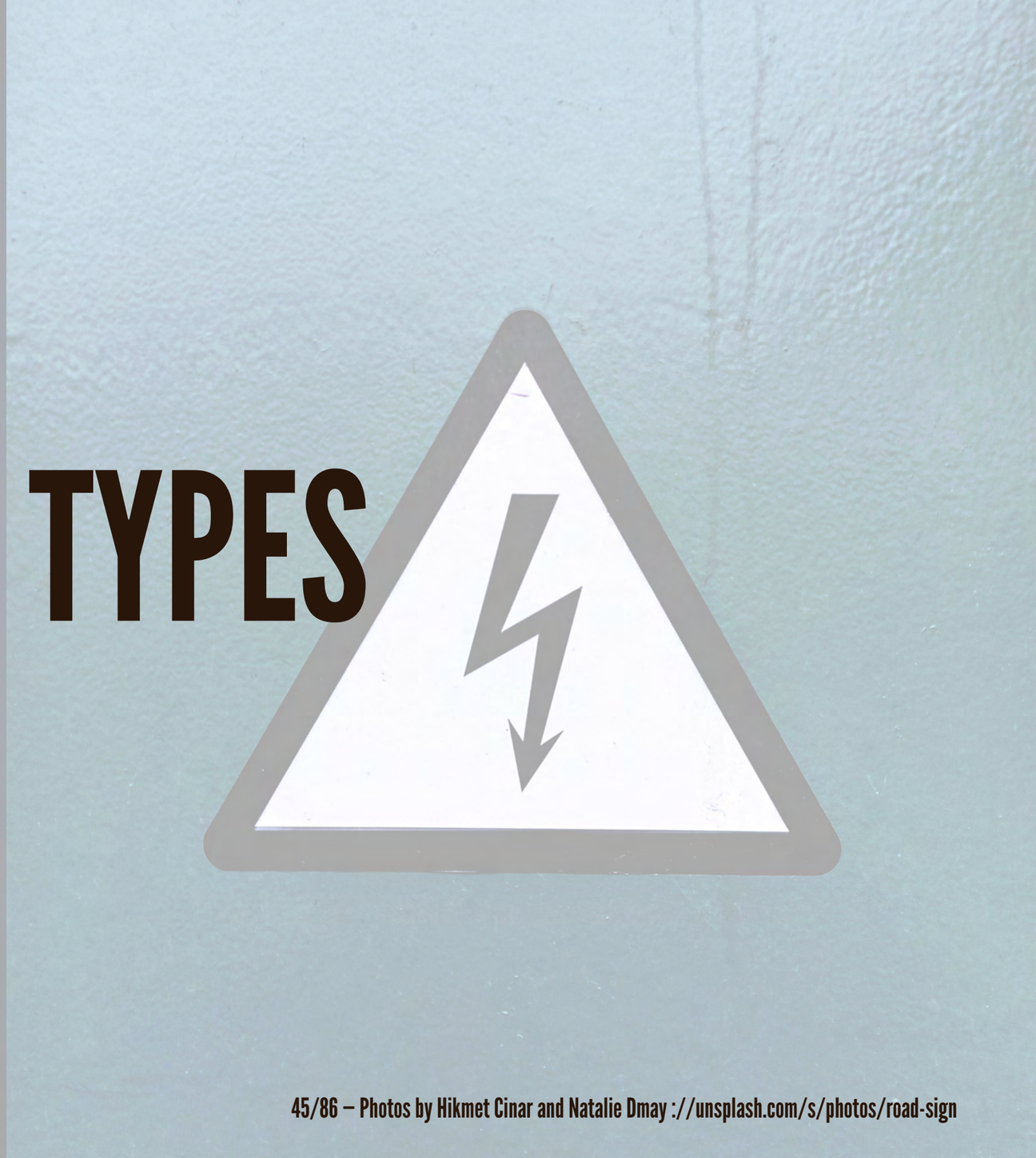
CODE
TYPE
BODY
HEADERS

```
router:RESPONSE_CODE
router:RESPONSE_TYPE
router:RESPONSE_BODY
router:RESPONSE_HEADERS
```

```
map {
                    router:RESPONSE_CODE    : xs:integer,
                    router:RESPONSE_TYPE    : xs:string,
                    router:RESPONSE_BODY    : item()*,
                    router:RESPONSE_HEADERS : map(xs:string, xs:string)
}
```

```
declare variable $router:RESPONSE_CODE    := xs:QName("router:RESPONSE_CODE");
declare variable $router:RESPONSE_TYPE    := xs:QName("router:RESPONSE_TYPE");
declare variable $router:RESPONSE_HEADERS := xs:QName("router:RESPONSE_HEADERS");
declare variable $router:RESPONSE_BODY    := xs:QName("router:RESPONSE_BODY");
```

MEDIA TYPES



45/86 – Photos by Hikmet Cinar and Natalie Dmay ://unsplash.com/s/photos/road-sign

Does this route allow this media to be sent?

```
{
  "content": {
    "text/plain": {
      "schema": { "type": "string" }
    }
  }
}
```

Determine serialization method from declared media type.

xml

- application/xml
- text/xml

json

- application/json

html5

- `text/html`

xhtml

- `application/xml+xhtml`

text

- image/jpeg
- application/javascript
- application/octet-stream

xml

- application/tei+xml
- application/*+xml
- image/svg+xml

json

- application/ld+json
- application/*+json

anything else is text

- `*/*`

```
{
  "requestBody": {
    "content": {
      "*/*": {
        "schema": { "type": "string", "format": "binary" }
      }
    }
  }
}
```

ERROR HANDLING

- central handling of errors
  - custom error handlers

```
{
  "get": {
    "x-error-handler": "my:html-error"
  }
}
```

```
declare function my:html-error(
  $error as map(*)
) as element(html) {
  <html><body>
    <h1>Ah Snap!</h1>
    <p>{$error?description}</p>
    <code>{$error?code}</code>
  </body></html>
};
```

AUTHENTICATION

60/86 – Photo by Tim Mossholder https://unsplash.com/s/photos/road-sign

```
{
  "securitySchemes": {
      "basicAuth": {
        "type": "http",
        "scheme": "basic"
      },
      "cookieAuth": {
        "type": "apiKey",
        "name": "roasted.com.login",
        "in": "cookie"
      }
  }
}
```

```
{
  "security": [ "cookieAuth", "basicAuth" ]
}
```

AUTHORISATION

63/86 – Photo by Bernd Dittrich https://unsplash.com/s/photos/road-sign

```
{
  "x-constraints": { "groups": "physics" }
}
```

```
{
  "x-constraints": { "groups": ["physics", "dba"] }
}
```

```
{
  "x-constraints": { "user": "admin" }
}
```

TO BUILD WITH

67/86 – Photo by ? (edit) https://unsplash.com/

LOGGING

68/86 – Photo by Ali Kazal https://unsplash.com/s/photos/road-sign

```
2021-11-01 12:34:56,370 ... request post /login
2021-11-01 12:34:56,388 ... post /login: 200
2021-11-01 12:34:56,394 ... request get /api/parameters
2021-11-01 12:34:56,400 ... get /api/parameters: 200
2021-11-01 12:34:56,431 ... request get /logout
2021-11-01 12:34:56,437 ... get /logout: 301
2021-11-01 12:34:56,455 ... request get /logout
2021-11-01 12:34:56,458 ... get /logout: 200
```

```
... [...351ab6cf7fb8] request post /login
... [...351ab6cf7fb8] post /login: 200
... [...56f2b155c571] request get /api/parameters
... [...56f2b155c571] get /api/parameters: 200
... [...645dd65dff94] request get /logout
... [...645dd65dff94] get /logout: 301
... [...715c59e53396] request get /logout
... [...715c59e53396] get /logout: 200
```

```
... INFO [...ddf283193aee] request post /api/errors/handle
... ERROR [...ddf283193aee] get /api/errors/handle: 500
{
    "code": "err:XPDY0002",
    "description": "[...] variable '$undefined' is not set.",
    "value": null,
    "module": "String/-2343316086301534543",
    "line":42, "column":5
}
```

MIDDLEWARE

- receives both request and response data
  - is allowed to modify *both*
  - widely adopted concept

```
function(map(*), map(*)) as map(*)+
```

```
declare function my:beep-boop (
    $request as map(*), $response as map(*)
) as map(*)+ {
    (: modify request :)
    map:put($request, "beep", "boop"),
    (: modify response :)
    map:put($response, $router:RESPONSE_HEADERS,
        map:merge((
            $response?($router:RESPONSE_HEADERS),
            map { "x-beep" : "boop" }
        ))
    )
};
```

CUSTOM AUTHORIZATION

```
{
    "JWTAuth": {
        "type": "apiKey",
        "in": "header",
        "name": "X-Auth-Token"
    }
}
```

```
{
  "security": [ "cookieAuth", "JWTAuth" ]
}
```

TODO

OPENAPI SPEC

- References with `$ref`
  - Schema Validation

TOOLING

- app generation with generator-exist
  - app scaffolding
- compiling API definitions to XQuery

SERVER SIDE

- Static file handling
- Roaster as a servlet

XQueryURLrewrite
controller.xq

~~XQueryURLrewrite~~
~~controller.xq~~

… AND MORE DOCUMENTATION

- roaster repository
- Readme
- Tests
- Example Application `roasted`
- OpenAPI Documentation

85/86 – Photo by Matt Botsford https://unsplash.com/s/photos/road-sign

JURI LEINO

@LINE-0

E-EDITIONES.ORG

EXIST-DB.ORG

EXIST-SOLUTIONS.COM

JINNTEC

86/86 – Photo by Jonathan Kemper https://unsplash.com/s/photos/road-sign