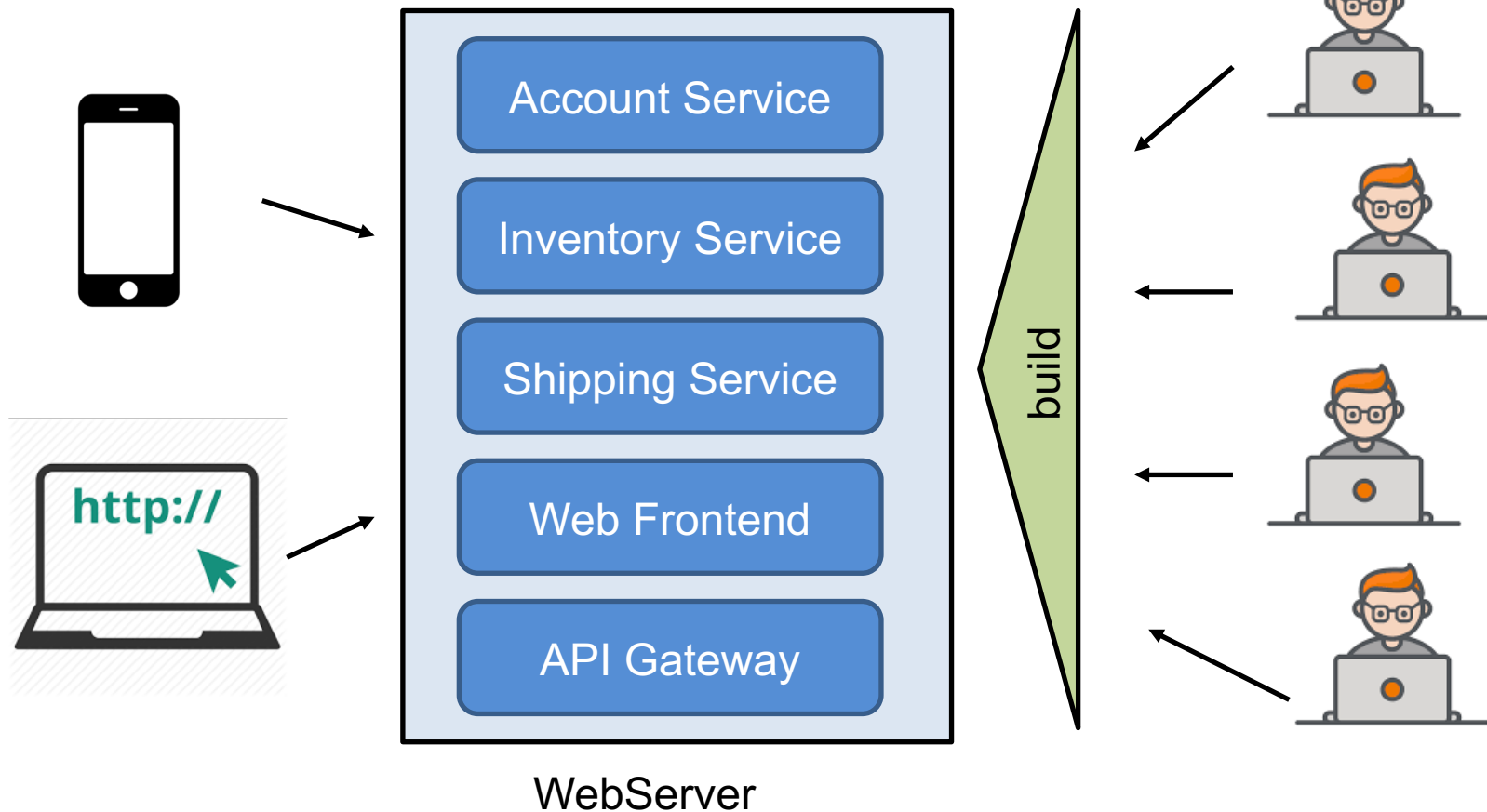


Microservices

Modul 14

Microservice Architecture

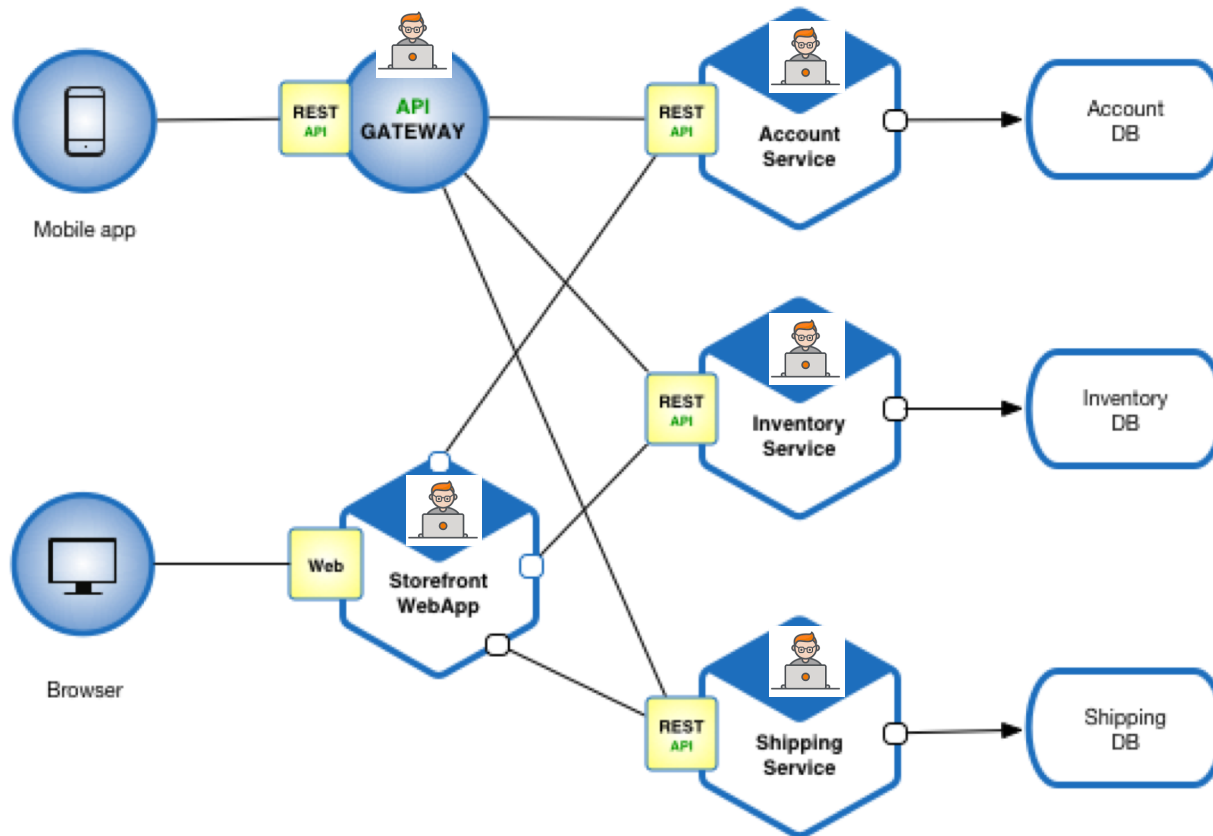
From monolithic application ...



Modul 14

Microservice Architecture

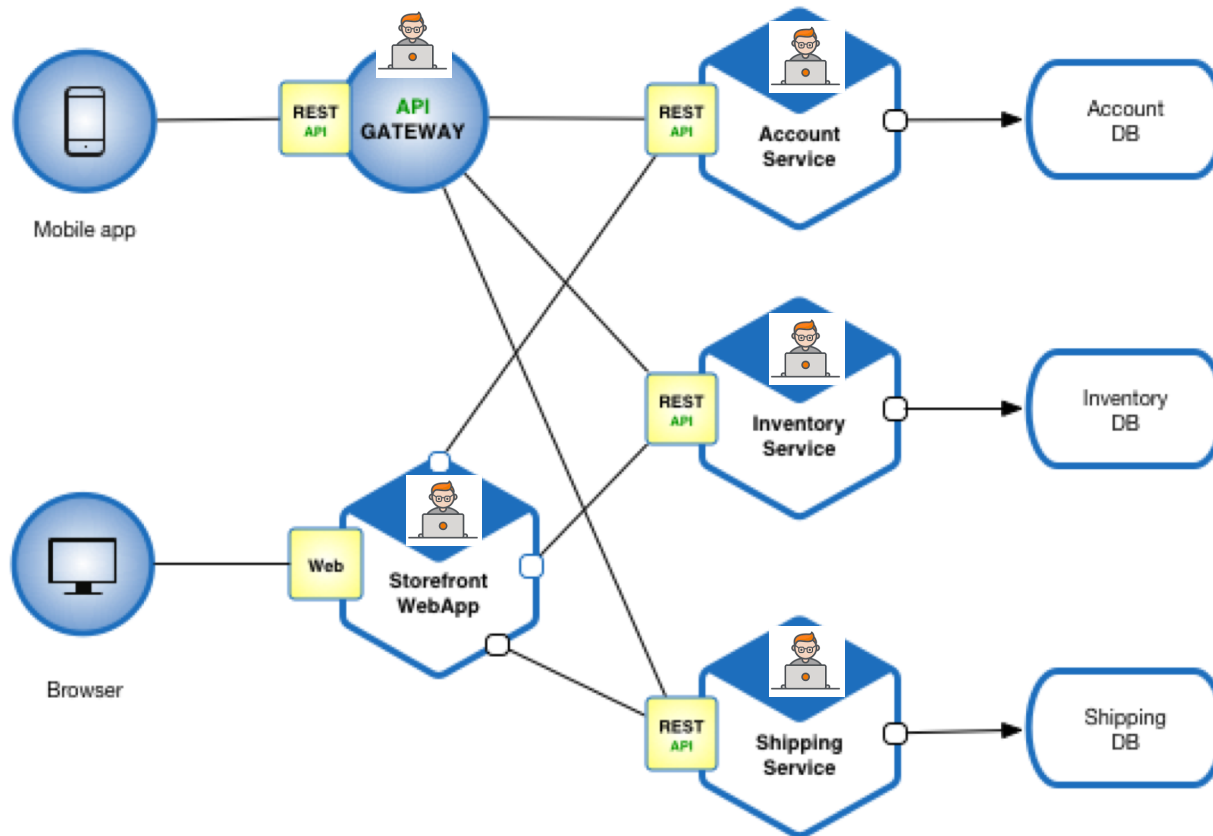
... to a collection of (independent) services.



Modul 14

Microservice Architecture

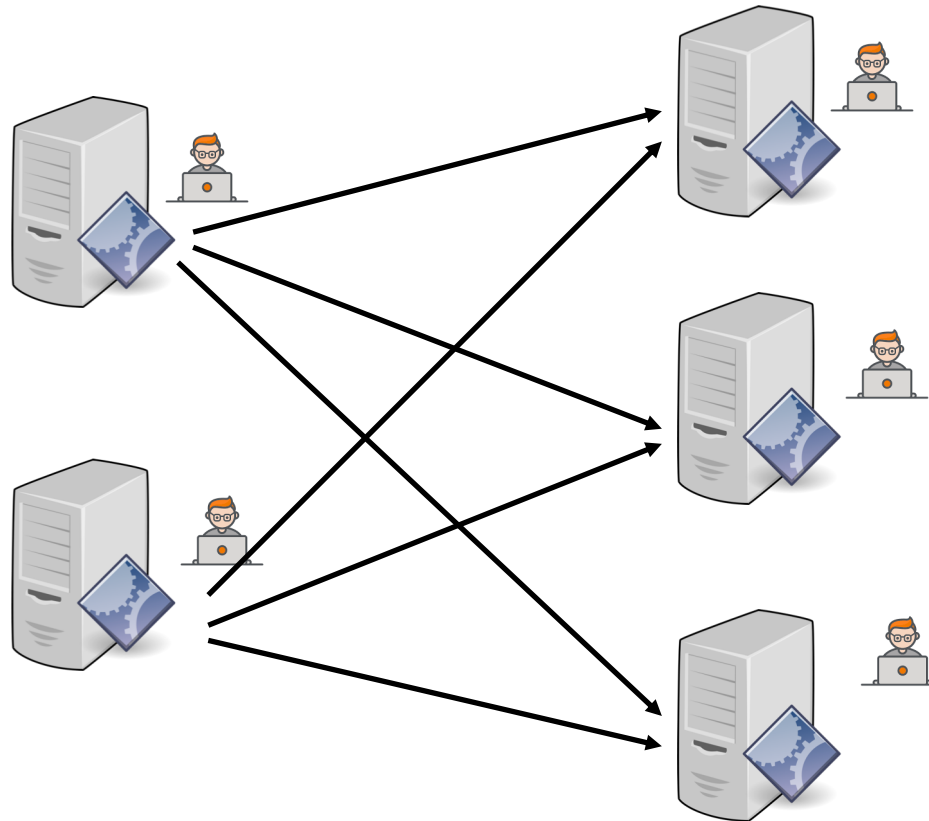
... to a collection of (independent) services.



Modul 14

Microservice

Each service is implemented as a Restful Webservice

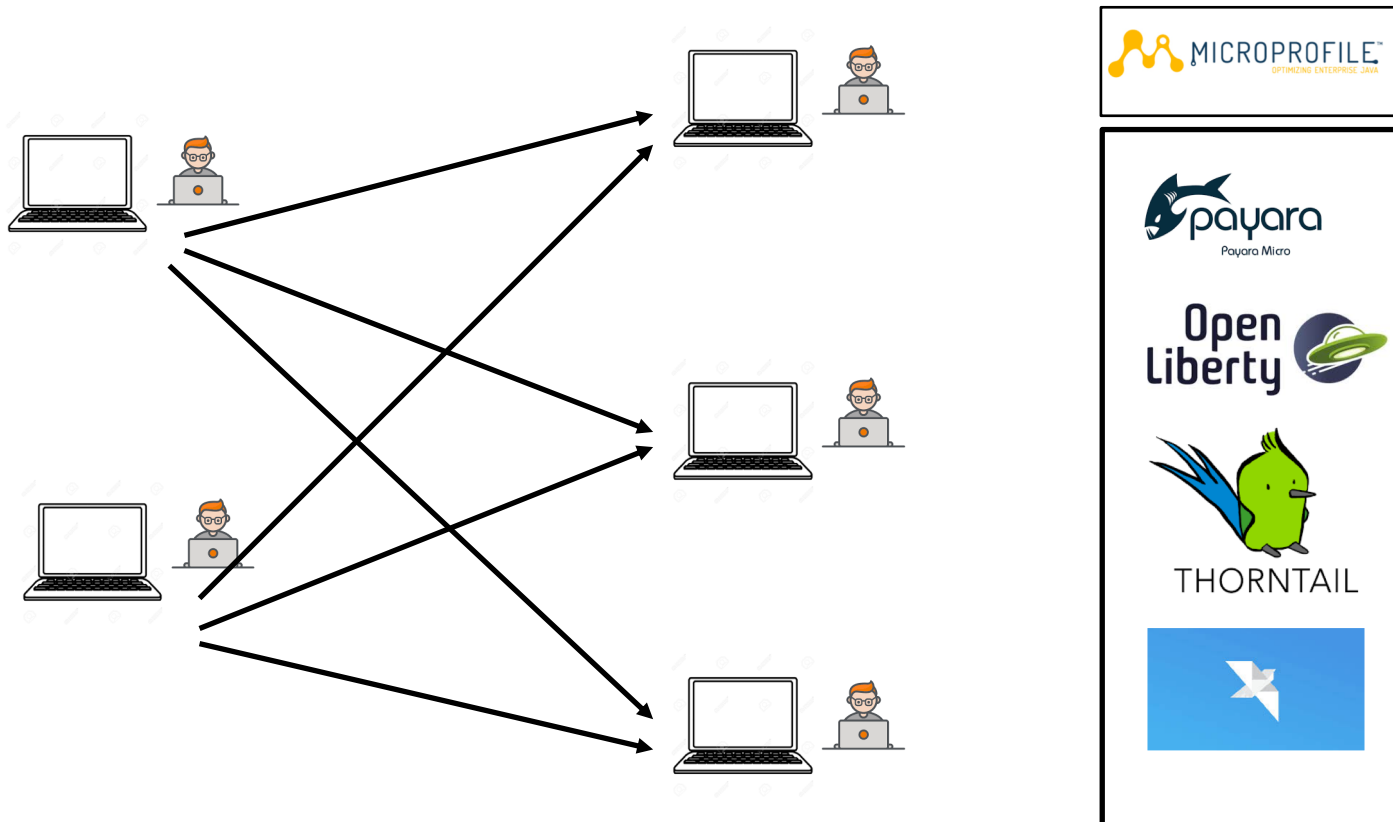


Problem:
A "big" application
server (~400 MB)
for each restful
webservice is
oversized ... and
makes deployment
too complex.

Modul 14

Microservice

Solution: Microservice Frameworks (Microprofile 2.x) with small embedded webserver



Modul 14

Restful with ThornTail (Getting started)

Change defaults

```
<packaging>war</packaging>
```

Properties

```
<properties>  
  <version.thorntail>2.4.0.Final</version.thorntail>  
  <failOnMissingWebXml>>false</failOnMissingWebXml>  
</properties>
```

Plugin

```
<groupId>io.thorntail</groupId>  
<artifactId>thorntail-maven-plugin</artifactId>  
<version>${version.thorntail}</version>
```

The Plugin come with the goals **package** and **run**
package generates a *fat-jar* (by default)
run starts the application (including the embedded webserver)

Modul 14

Restful with ThornTail (Getting started)

Dependency

```
<dependency>  
  <groupId>io.thorntail</groupId>  
  <artifactId>jaxrs</artifactId>  
  <version>${version.thorntail}</version>  
</dependency>
```

Microprofile is based on JAX-RS (Jersey)

Modul 14

Restful with ThornTail (Getting started)

Create a Java Application with Annotation

```
@ApplicationPath("/myApp")  
public class HalloApplication extends Application {  
}
```

Create a Rest-Endpoint

```
@Path("")  
public class HalloEndPoint {  
    @GET  
    @Produces("text/plain")  
    @Path("/name")  
    public String retHallo() {  
        return "Hello World!";  
    }  
}
```

Test

```
Via CLI: mvn thortail:run  
Im Browser: localhost:8080/myApp/name
```

Modul 14

Restful Endpoints – POST-Example

Example: POST a String, use Jackson ObjectMapper for parsing JSON

```
@POST
@Produces("application/json")
@Consumes("application/json")
@Path("/image")
public String retImg(String body) {
    String imgstr = "???";
    ObjectMapper mapper = new ObjectMapper();
    try {
        JsonNode root = mapper.readTree(body);
        imgstr = root.get("image").asText();
    } catch (IOException ex) {
        Logger.getLogger(CalcImage.class.getName()).
            log(Level.SEVERE, null, ex);
    }
    return "received: " + imgstr ;
}
```

Modul 14

Using the Response-Object

```
public class HalloEndPoint {  
    @GET  
    @Produces("text/plain")  
    @Path("/hi")  
    public Response retHallo3() {  
  
        return Response  
            .status(Response.Status.OK)  
            .entity("Hallo World!")  
            .type(MediaType.TEXT_PLAIN)  
            .build();  
    }  
}
```

<https://developer.mozilla.org/de/docs/Web/HTTP/Status>

Swagger
==
OpenApi

Modul 14

Swagger – OpenAPI Specification

OpenAPI Specification (OAS) == Swagger Specification(old):

- Standard for defining RESTful interfaces



Field Pattern	Type	Description
/ {path}	Path Item Object	A relative path to an individual endpoint. The field name MUST begin with a slash. The path is appended to the basePath in order to construct the full URL. Path templating is allowed.
^x-	Any	Allows extensions to the Swagger Schema. The field name MUST begin with x- , for example, x-internal-id . The value can be null , a primitive, an array or an object. See Vendor Extensions for further details.
...		

OAS 2.0 -> Swagger 1.5.X

OAS 3.0 -> Swagger 2.X

Modul 14

Swagger - OpenAPI Specification

Example

```
public class HalloEndPoint {  
    @GET  
    @Produces("text/plain")  
    @Path("/hi")  
    public Response retHallo3() {  
  
        return Response  
            .status(Response.Status.OK)  
            .entity("Hallo World!")  
            .type(MediaType.TEXT_PLAIN)  
            .build();  
    }  
}
```

Modul 14

Swagger – OpenAPI Specification

JSON

```
"/hi": {  
  "get": {  
    "operationId": "retHallo3",  
    "produces": [  
      "text/plain"  
    ],  
    "parameters": [],  
    "responses": {  
      "default": {  
        "description": "successful operation"  
      }  
    }  
  }  
}
```

Modul 14

Swagger – OpenAPI Specification

YAML

```
/hi:  
  get:  
    operationId: retHallo3  
    produces:  
      - text/plain  
    parameters: []  
    responses:  
      default:  
        description: successful operation
```

Modul 14

Swagger – OpenAPI Specification

Example

```
@GET
@Produces("text/plain")
@Path("/function/{nr}")
public Response retFuctName(@PathParam("nr") String nr) {
    String str = "";
    if (nr.equals("1") ) str = "OrigImg";
    else if (nr.equals("2") ) str = "GrayScaleImg";
    else str = "error";
    if (str.equals("error")) return
        Response.status(Response.Status.NOT_FOUND).
            entity(str).type(MediaType.TEXT_PLAIN).build();
    else return
        Response.status(Response.Status.OK).
            entity(str).type(MediaType.TEXT_PLAIN).build();
}
```

Modul 14

Swagger – OpenAPI Specification

JSON

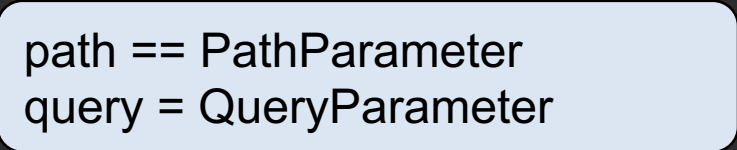
```
"/function/{nr}": {
  "get": {
    "operationId": "retFuctName",
    "produces": [
      "text/plain"
    ],
    "parameters": [
      {
        "name": "nr",
        "in": "path",
        "required": true,
        "type": "string"
      }
    ],
    "responses": {
      "default": {
        "description": "successful operation"
      }
    }
  }
}
```

Modul 14

Swagger – OpenAPI Specification

YAML

```
'/function/{nr}':  
  get:  
    operationId: retFuctName  
    produces:  
      - text/plain  
    parameters:  
      - name: nr  
        in: path  
        required: true  
        type: string  
    responses:  
      default:  
        description: successful operation
```



path == PathParameter
query = QueryParameter

Modul 14

Swagger – OpenAPI Specification

Example

```
@GET
@Produces("text/plain")
@Path("/function/{nr2}/{nr3}")
public Response retFuctName(@QueryParam("nr1") String nr,
    @PathParam("nr2") String n1,
    @PathParam("nr3") String n3) {
    String str = "";
    if (nr.equals("1") ) str = "OrigImg";
    else if (nr.equals("2") ) str = "GrayScaleImg";
    else str = "error";
    if (str.equals("error")) return
        Response.status(Response.Status.NOT_FOUND).
            entity(str).type(MediaType.TEXT_PLAIN).build();
    else return
        Response.status(Response.Status.OK).
            entity(str).type(MediaType.TEXT_PLAIN).build();
}
```


Modul 14

Swagger – OpenAPI Specification

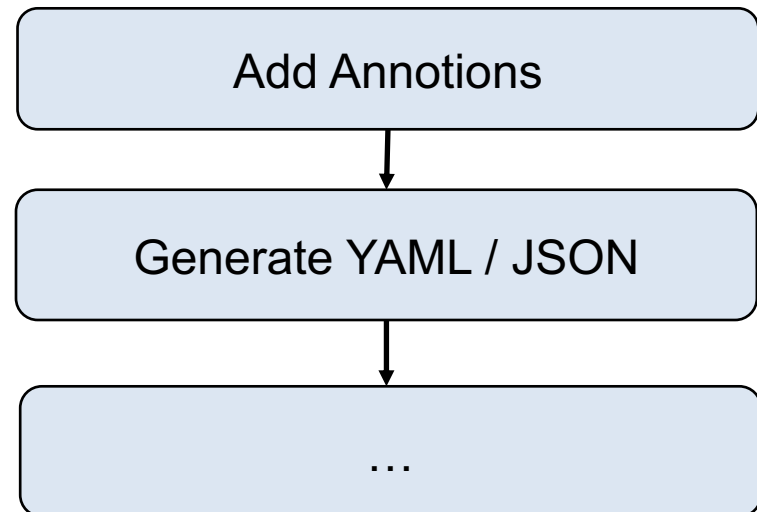
```
'/function/{nr2}/{nr3}':  
  get:  
    operationId: retFuctName  
    produces:  
      - text/plain  
    parameters:  
      - name: nr1  
        in: query  
        required: false  
        type: string  
      - name: nr2  
        in: path  
        required: true  
        type: string  
      - name: nr3  
        in: path  
        required: true  
        type: string  
    responses:  
      default:  
        description: successful operation
```

Modul 14

Swagger – CodeFirst



Generating OAS From Code



Modul 14

Swagger – CodeFirst

For ThornTail add dependency

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>swagger</artifactId>
  <version>${version.thorntail}</version>
</dependency>
```

Add Annotations to your code ...

```
@API( ... )
@SwaggerDefinition( ... )
@ApiOperation( ... )
@ApiResponses( ... )
@ApiResponse( ... )
```

Modul 14

Swagger – @API

@API must be set before each class which defines a Restful endpoint

```
@Api( "/myDemo" )  
@Path( "mini" )  
public class MiniRest {  
    @GET  
    @Path( "/demo" )  
    public String retName() {  
        return "Grabowski";  
    }  
}
```

Modul 14

Swagger – @API

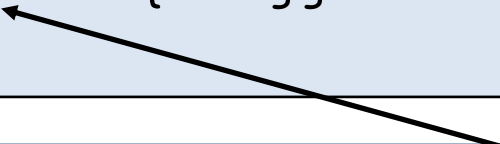
Generated OAS (YAML)

```
/mini/demo:
  get:
    tags:
      - mini
    operationId: retName
    parameters: []
    responses:
      '200':
        description: successful operation
        headers: {}
        schema:
          type: string
```

Modul 14

Swagger – @SwaggerDefinition(...)

```
@SwaggerDefinition (
    info = @Info (
        title = "Demo für Swagger",
        description = "A simple example here",
        version = "1.0.0",
        contact = @Contact (
            name = "Hartwig Grabowski",
            email = "hag@web.de",
            url = "https://www.hs-offenburg.de"
        )
    ),
    schemes = {SwaggerDefinition.Scheme.HTTP}
)
```



How to access the restful endpoint (Http or Https)

Modul 14

Swagger – @SwaggerDefinition(...)

Generated OAS (YAML)

```
swagger: '2.0'
info:
  description: A simple example here
  version: 1.0.0
  title: Demo für Swagger
  contact:
    name: Hartwig Grabowski
    url: 'https://www.hs-offenburg.de'
    email: hag@web.de
basePath: /myApp
tags:
  - name: mini
schemes:
  - http
```

Modul 14

Swagger – @ApiOperation(...)

```
@GET
    @Produces("text/plain")
    @Path("/stud1")
    @ApiOperation(
        value = "Retruns a name",
        notes = "Only surname")
    public String retName1() {
        return "Grabowski";
    }
```



```
/stud1:
  get:
    summary: Retrurns a name
    description: Only surname
    operationId: retName1
    produces:
      - text/plain
    parameters: []
    responses:
      '200':
        description: successful operation
        schema:
          type: string
```

Modul 14

Swagger – @ApiOperation(...)

```
@GET
@Produces("text/plain")
@Path("/stud2")
@ApiOperation(
    value = "Retruns a student",
    notes = "JSON of student-object",
    response = Student.class)
public String retName2() {
    return "... json-of-student-obj ...";
}
```

```
public class Student {
    String name;
    String matNr;
    // Setter & Getter
}
```

Modul 14

Swagger – @ApiOperation(...)

```
/stud2:
  get:
    summary: Retrurns a student
    description: JSON of student-object
    operationId: retName2
    produces:
      - text/plain
    parameters: []
    responses:
      '200':
        description: successful operation
        schema:
          $ref: '#/definitions/Student'
```

Modul 14

Swagger – @ApiOperation(...)

```
/stud2:  
  get:  
    summary: Retrurns a student  
    description: JSON of student  
    operationId: retName2  
    produces:  
      - text/plain  
    parameters: []  
    responses:  
      '200':  
        description: success  
        schema:  
          $ref: '#/definitions/Student'
```

```
definitions:  
  Student:  
    type: object  
    properties:  
      name:  
        type: string  
      matNr:  
        type: string
```

```
@GET
@Produces("text/plain")
@Path("/name")
@ApiOperation(
    value = "Retruns a list of students",
    notes = "array of JSON of student-object",
    response = Student.class,
    responseContainer = "List")
    //Only List, Set or Map
public String retName3() {
    return "... List of json-of-student-obj ...";
}
```

Modul 14

Swagger – @ApiOperation(...)

```
/name:  
  get:  
    summary: Retrurns a list of students  
    description: array of JSON of student-object  
    operationId: retName3  
    produces:  
      - text/plain  
    parameters: []  
    responses:  
      '200':  
        description: successful operation  
        schema:  
          type: array  
          items:  
            $ref: '#/definitions/Student'
```

Modul 14

Swagger – @ApiResponses(...)

```
@GET
@Produces("text/plain")
@Path("/function/{nr2}/{nr3}")
@ApiResponses(value = {
    @ApiResponse(code = 200, message = "Alles klar"),
    @ApiResponse(code = 404, message = "Nr > 2") })
public Response retFuctName(@QueryParam("nr1") String nr,
    @PathParam("nr2") String n1,
    @PathParam("nr3") String n3) {
    String str = ""; ...
    if (str.equals("error")) {
        return Response.status(Response.Status.NOT_FOUND).
            entity(str).type(MediaType.TEXT_PLAIN).build();
    } else {
        return Response.status(Response.Status.OK).
            entity(str).type(MediaType.TEXT_PLAIN).build();
    }
}
```

Modul 14

Swagger – @ApiResponses(...)

```
'/function/{nr2}/{nr3}':  
  get:  
    operationId: retFuctName  
    produces:  
      - text/plain  
    parameters:  
      - name: nr1  
        in: query  
        required: false  
        type: string  
      - name: nr2  
        in: path  
        required: true  
        type: string  
      - name: nr3  
        in: path  
        required: true  
        type: string  
    responses:  
      '200':  
        description: Alles klar  
      '404':  
        description: Falsche Nummer >2
```

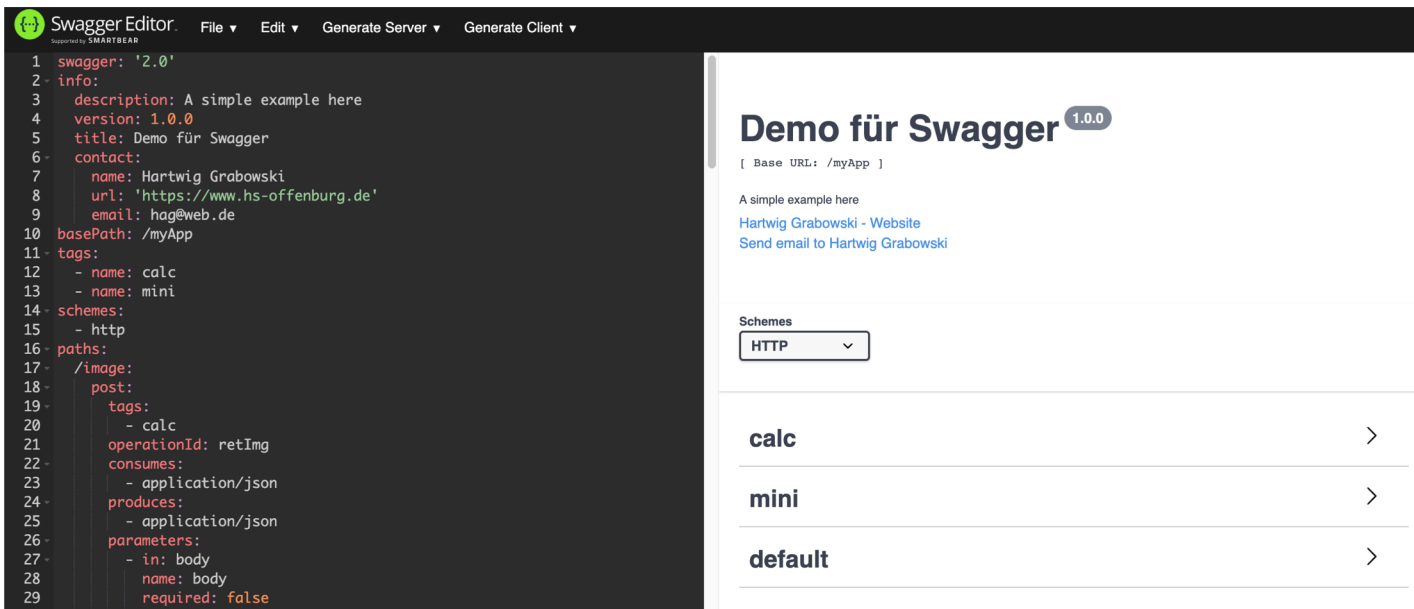

Modul 14

Swagger - editor

Swagger-core generates the OAS in JSON under swagger.json
(e.g.: <http://localhost:8080/myApp/swagger.json>)

```
{ "swagger": "2.0", "info": { "description": "A simple  
example here." ... } } }
```

Copy and paste the content into <http://editor.swagger.io/>



The screenshot shows the Swagger Editor interface. On the left, a code editor displays a JSON definition for a Swagger API. The JSON includes fields for version (2.0), description, title, contact information, base path, tags, and a single endpoint (POST /image) with a body parameter. On the right, the rendered API preview is shown. It features a title 'Demo für Swagger' with a version badge '1.0.0', a base URL, a description, and links to the author's website and email. Below this, a 'Schemes' dropdown is set to 'HTTP'. At the bottom, a list of tags ('calc', 'mini', 'default') is displayed with expandable arrows.

```
1 swagger: '2.0'
2 info:
3   description: A simple example here
4   version: 1.0.0
5   title: Demo für Swagger
6   contact:
7     name: Hartwig Grabowski
8     url: 'https://www.hs-offenburg.de'
9     email: hag@web.de
10  basePath: /myApp
11  tags:
12    - name: calc
13    - name: mini
14  schemes:
15    - http
16  paths:
17    /image:
18      post:
19        tags:
20          - calc
21        operationId: retImg
22        consumes:
23          - application/json
24        produces:
25          - application/json
26        parameters:
27          - in: body
28            name: body
29            required: false
```

Demo für Swagger 1.0.0

[Base URL: /myApp]

A simple example here
[Hartwig Grabowski - Website](#)
[Send email to Hartwig Grabowski](#)

Schemes
HTTP

calc >

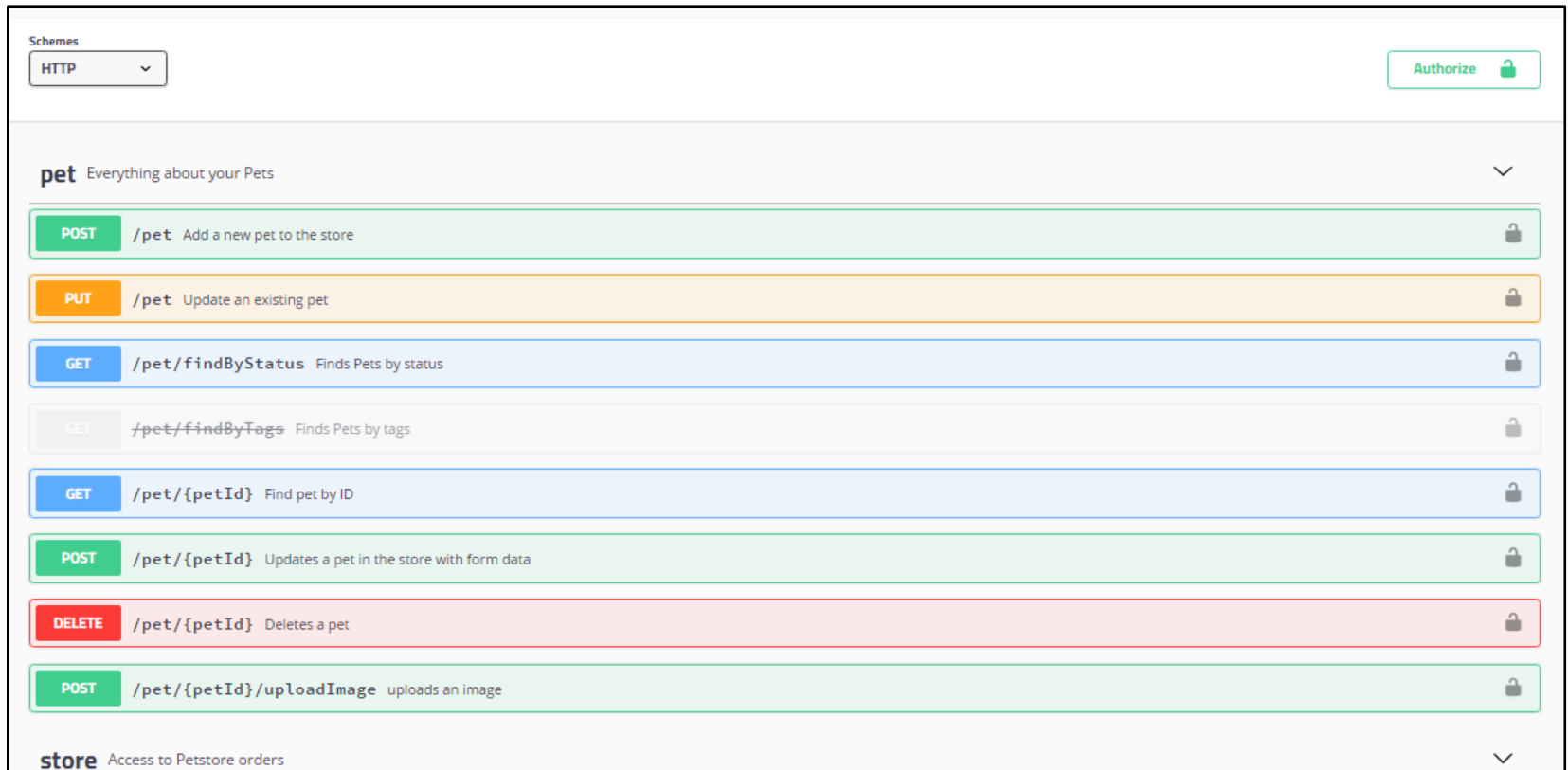
mini >

default >

Modul 14

Swagger – Swagger-UI

Explore and test the generated Restful services



Modul 14

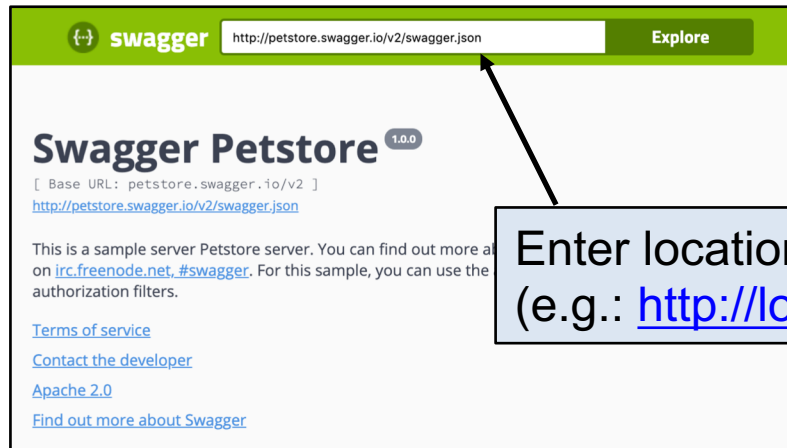
Swagger – Swagger-UI

For ThornTail add dependency

```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>swagger</artifactId>
  <version>${version.thorntail}</version>
</dependency>
```

Build the project and open ...

<http://localhost:8080/swagger-ui/index.html>



Enter location of swagger.json in input filed
(e.g.: <http://localhost:8080/myApp/swagger.json>)

Modul 14

Swagger – Swagger-UI

Restful services can be tested in the browser

The image shows the Swagger-UI interface for a REST API endpoint. The endpoint is `GET /function/{nr2}/{nr3}`. The interface includes a "Parameters" section with three parameters: `nr1` (string, query), `nr2` (string, path, required), and `nr3` (string, path, required). Each parameter has a corresponding input field. Below the parameters is an "Execute" button and a "Clear" button. The "Responses" section shows the "Response content type" set to "text/plain". The "Curl" section is at the bottom.

GET /function/{nr2}/{nr3}

Parameters Cancel

Name	Description
nr1 string (query)	2
nr2 * required string (path)	1
nr3 * required string (path)	2

Execute Clear

Responses Response content type text/plain

Curl