NVK SLIDE COLLECTIONS

AARHUS
UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING

HENRIK BITSCH KIRK
ASSOCIATE PROFESSOR

FORMATS

JSON:

```
{
  "data":[
    {
      "type":"articles",
      "id":"1",
      "attributes":{
        "title":"JSON:API paints my bikeshed!",
        "body":"The shortest article. Ever.",
        "created":"2015-05-22T14:56:29.000Z",
        "updated":"2015-05-22T14:56:28.000Z"
      },
      "relationships":{
        "author":{
          "data":{
            "id":"42",
            "type":"people"
          }
        }
      }
    }
  ]
}
```

XML:

```
<CATALOG>
    <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
    </CD>
    <CD>
        <TITLE>Hide your heart</TITLE>
        <ARTIST>Bonnie Tyler</ARTIST>
        <COUNTRY>UK</COUNTRY>
        <COMPANY>CBS Records</COMPANY>
        <PRICE>9.90</PRICE>
        <YEAR>1988</YEAR>
    </CD>
</CATALOG>
```

FRAMEWORK

- Library: Like buying IKEA furniture, you need a new table, but don't won't to make it from scratch
  - Software: You decide where and how to use the library code
- Framework: Like building a hose from a blueprint, where you have limited choices.
  - Software:
    - Control is moved to Framework (inversion of control)
    - Flow is therefore 'dictacted' from framework

WHAT IS REST

- Set of guiding principles that an Web API should adhere to to be an REST API
- Architectural style (like 3 layer model)

GUIDING PRINCIPLES

1. Uniform interface
    1. Resource identification
    2. Initial URL
2. Client-server
    1. Separation of concern
3. Stateless
    1. The server doesn't hold any state
4. Cacheable
    1. The client can save the response
5. Layered System
    1. Only depended on the next immediate layer
6. Code on Demand (optional and not often used in practice)
    1. Possibility to extend clients

SECURITY

- HTTPS
  - Always use HTTPs when creating Web API (And web pages in general)
- API keys
  - Hide this key
- OAUTH

BUILDING REST

ASP.Net WebAPI

AARHUS UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NVK
16 NOVEMBER 2022

HENRIK BITSCH KIRK
ASSOCIATE PROFESSOR

DESIGNING REST API

1. Identify resources
2. Create URI
3. Determining representation
4. Assigning HTTP Verbs

IDENTIFYING RESOURCES

Can typically be found in

- Class Diagram
- Entity Relationship Diagram

- Should properly have an ID

CREATING URIS

- Should be **nouns** only
- Resource normally in plural
  - And {id} to select specific
  - `/patients` vs `/patients/{cpr}`
- Sub-collections can be specified in URL by specifying collection URL
  - `/patients/{id}/observations`

RESOURCE REPRESENTATION

- Either XML or JSON
- Returning the most important data - especially when returning a collection
- Returning a single resource
  - Include all data
  - Include relevant links
  - E.g. link to patients observations

  /patients/1234567890/oberservations/123

  /patients/1234567890/oberservations/180

  /patients/1234567890/oberservations/231

HTTP VERBS

- Use GET to browse data
  - Big collections should have pagination
  - Browser secondary collection

  /patients/123456789/observations
- Use POST to create
- Use PUT to update
- Use DELETE to delete
  - Could be necessary to also use PUT to update (remove from) subcollections

ASP.NET WEBAPI

- Controllers in `WebAPI` should inherit from `ControllerBase`
- You annotate each controller with `[APIController]` and `[Route]`

```
[ApiController]
[Route("[controller]")]
public class WeatherForecastController : ControllerBase
```

- Normal is one controller per resource

HTTP ENDPOINTS

- Each method in a Controller that is annotated with [HTTP*] will be 'translated' as an endpoint

```
[HttpPost]
public IActionResult CreatePatient(Patient patient) // or
[HttpGet]
public IActionResult<List<Patient>> Get() // or
[HttpGet]
[Route("Patients")]
public IActionResult<Patient> GetById(string id)
```

- Using `[FromBody]`, `[FromQuery]`, etc parses parameters from either request body or parameters in URI.

HATEOAS

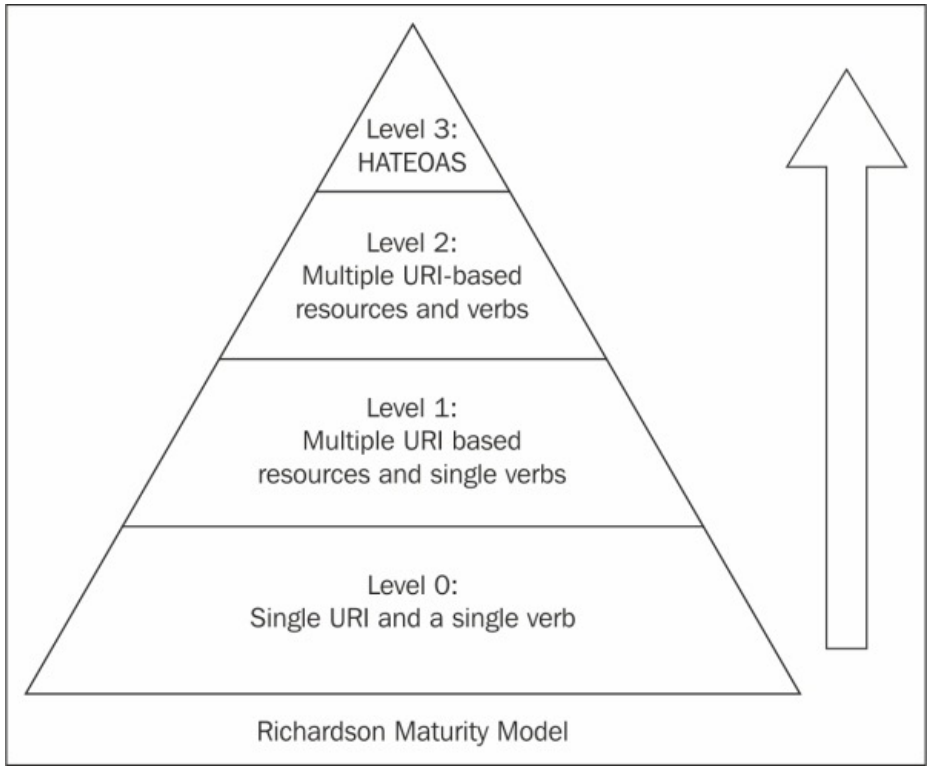
Hypermedia **A** **T**he **E**ngine **O**f **A**pplication **S**tate

```
{
 "departmentId": 10, "departmentName": "Administration", "locationId": 1700, "managerId": 200,
        "links": [ {
                    "href": "10/employees",
                    "rel": "employees",
                    "type" : "GET”
         } ]
}
```

- A dynamic way of navigating to related resources
- Clients don’t need to be hard-coded
- Don’t need to be in the body – could also be in the header

REST MATURITY LEVELS

- Level 0:
  - Single word and single VERB
  - E.g. only a single http endpoint using post or get
- Level 1:
  - Level 0, but with multiple resources
- Level 2:
  - Multiple resources (endpoints) and uses different VERBs for CRUD operations
  - This you get from WebAPI if you use it like in the tutorial
- Level 3:
  - Level 2 + HATEOAS



IMPLEMENTING HETEOAS

- Return type:

  media type: application/hal+json

  media type: application/hal+xml

DATA

```
public class Link
{
    public string? Rel { get; set; }
    public string? HRef { get; set; }
}

public abstract class LinkedResource
{
    public List<Link> Links { get; set; }
    public string HRef { get; set; }
}
```

SETTING DATA

```
[HttpGet(Name = "GetAllPatients")]
public IEnumerable<Patient> GetAll()
{
    return PatientService.GetAll().Select(patient =>
    {
        Console.WriteLine(patient.Identifier.First().Value);
        patient.Links = new List<Link>()
        {
            new()
            {
                Rel = "self",
                HRef = Url.Link("GetPatientFromCpr", new { cpr = patient.Identifier.First().Value})
            },
        };
        return patient;
    });
}
```

HL7 FHIR

AARHUS
UNIVERSITY
DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING

NVK
16 NOVEMBER 2022

HENRIK BITSCH KIRK
ASSOCIATE PROFESSOR

FHIR

- Fast Healthcare Interoperability Resources
- Consists of modular set of Resources
- Usage
  - Mobile application, Cloud communication, EHR, …
  - Human and veterinary
  - Clinical care, public health, clinical trials, administration, financial

- Built with web standards
  - XML and JSON
  - HTTP
  - Oauth
- RESTfull architecture

RESOURCES

- Resource identity (ID) and Metadata
- Human readable summary (XHTML)
- Extensability URL -> definition
  - Handle local differences (due to law, practices etc.)
- Standard data

RESOURCE DESIGN

Designed for

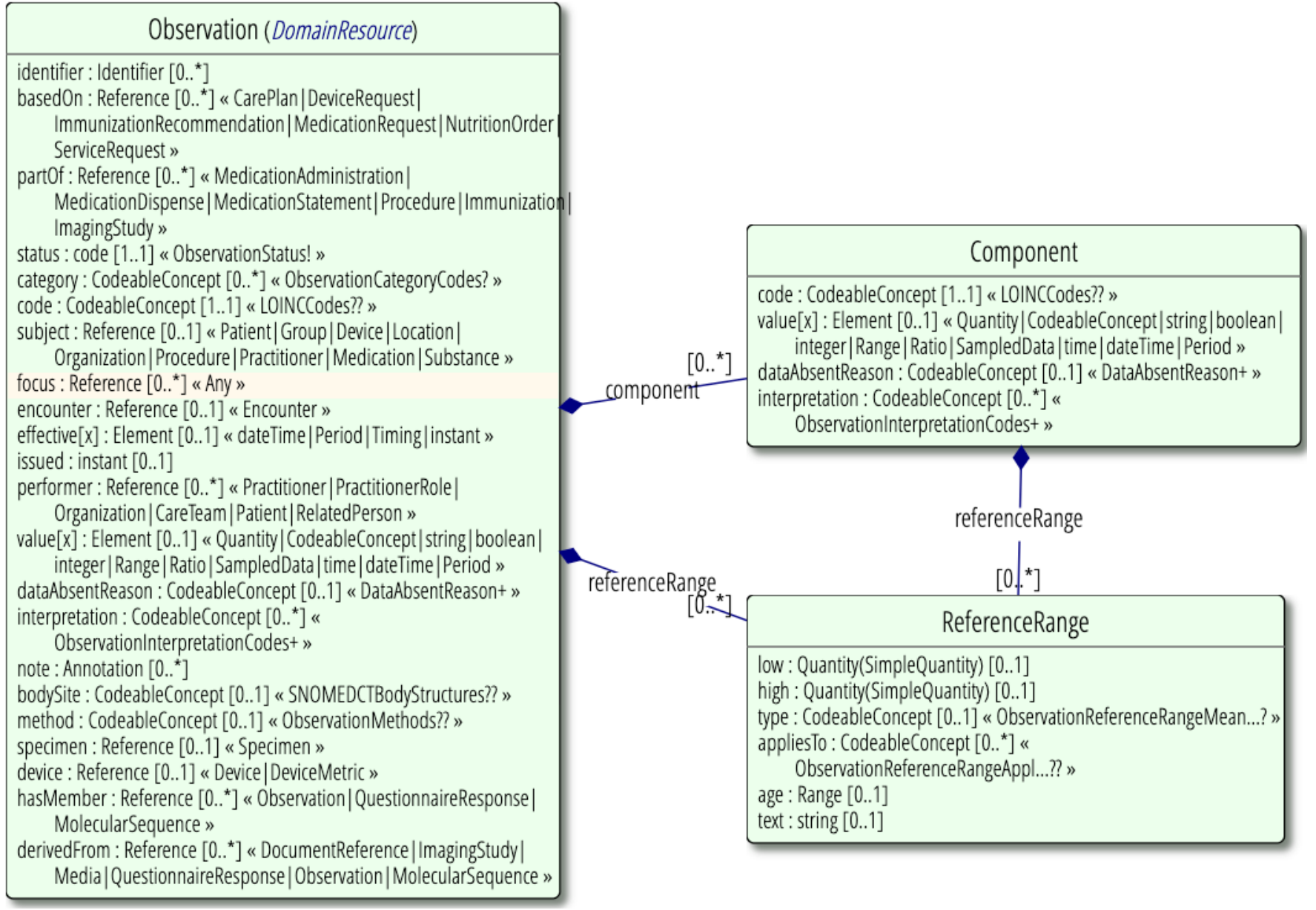
- Reuse and composability
- Scalability
- Performance
- Usability
- Data fidelity
- Implementability

RESOURCE ORGANIZATION

6 layers:

- Foundation
  - Foundation resource – mostly (but not only) used for infrastructure and mostly (not only) referenced by other resources
- Clinical
  - Covers most common use cases
- Financial
- Specialized
  - Less common use cases
- Resource contextualization
  - Do not contain resources – used to ”extend, constrain and contextualize resources for a given purpose”

OBSERVATION



AARHUS
UNIVERSITY