Naked-FHIR
Code-generation using HL7 FHIR
Oliver Krauss

HL7 International 2015
10. May 2015
Introduction

- HL7 FHIR provides several base resources
- **Conformance** resources provide **Conformance Statements** for client applications
  > Describes offered resources (Profiles), operations, valueSets, etc.
  > Client requirements eg. „Contract“ between Server and Client

- Question: How to generate a client application based on a **Conformance Statement**?
„Old school“ application architecture

- Application logic intermediates between domainobjects and presentation
- Application-tier cannot be automatically generated

> Additional information is required to map domainobjects (FHIR resources) to application logic
Naked Object Pattern

- Presentation-tier as a direct representation of domain objects
- User interface can be generated 100% from the domain object definitions
- Metainformation on available operations is required

Source: http://de.wikipedia.org/wiki/Naked_Objects
Concept

- Abstraction of **Naked Objects Pattern** regarding the support of FHIR resources
- Automated generation of client applications using the information contained in **Conformance Statement**, **Profile**, *etc.*
- Structure of resources (elements, datatypes, cardinality)
  > Operations on resources (Read, Create, Search, etc.)
Concept
Process of code generation

1. Retrieve Conformance
   > Get **Conformance** of FHIR-Server we want the client to communicate with

   ```
   GET [base]/metadata {?_format=[mime-type]}
   ```

2. Select Supported Profiles
   > Selection of the **Profiles**, that the application needs process.
Concept
Process of code generation

3. Model Transformation (M2M)
   > Mapping and transformation of selected profiles to ApplicationModel
     - Aggregates selected profiles as well as structural information of the application to be generated
     - Is an abstract representation of the Naked Objects Pattern
Concept
Process of code generation

4. Generate Code (M2C)
   > Generates application source code from the ApplicationModel
   > Templates are used for mapping the model to a user interface
Concept
Mapping of FHIR resources to the user interface

- Combination of resource and operation defines layout of views in application
- Elements of the resource define composition of UI
- Operation defines possible interactions
  - Create, Update, Delete, etc.
Concept
Mapping of FHIR resources to the user interface

- Resources consist of elements
  - Elements have types (datatype, resource, element) and a cardinality
- Datatypes are the minimal unit of any element
- Provide templates for each possible combination of
  - Datatype (primitive, complex) and
  - Applicable operations (read, search, create, etc.)
- UI for resource is built by aggregating templates for all elements of a resource

[Quelle: http://www.fhir.org]
Concept
Mapping of FHIR resources to the user interface

- Example: operation READ on resource **Patient**

```plaintext
Patient (Resource)

- identifier : Identifier 0..*
- name : HumanName 0..*
- telecom : Contact 0..*
- gender : CodeableConcept 0..1 <<AdministrativeGender>>
- birthDate : dateTime 0..1
- deceased[x] : boolean | dateTime 0..1
- address : Address 0..*
- maritalStatus : CodeableConcept 0..1 <<MaritalStatus>>
- multipleBirth[x] : boolean | integer 0..1
- photo : Attachment 0..*
- communication : CodeableConcept 0..1 <<Language>>
- careProvider : Resource(Organization | Practitioner) 0..*
- managingOrganization : Resource(Organization) 0..1
- active : boolean 0..1

[Quelle: http://www.fhir.org]
```

[View of generated application (Android)]
Example: Patient resource

- Generation of an android application for resource **Patient** and the operations **READ** and **SEARCH**
Example: Patient resource

Select resource to be generated

Select operations to be supported
Example: Patient resource

- Result of code generation
  > One Android **Activity** for **SEARCH** and **READ** on resource **Patient** respectively

**Operation**
- SEARCH

**HumanName**

**Operation**
- READ

**HumanName**

**Contact**
Possible areas of application

- **Rapid Application Development (RAD)**
  > Easily generate a first prototype as part of an iterative incremental process model
  > Adapt generated source code

- Because of the abstraction of the **Naked Object Pattern** other applications can be implemented by adding additional templates
  > Ex.: automated test generation
    - Check if a FHIR-Server implements everything published in its **Conformance Statement**
Status

- Naked FHIR prototype supports only FHIR DSTU1
  > Adaptation to FHIR DSTU2 by *Bootstraping*

- Problem with resource elements that are typeless
  > Ex.: *PatientAnimal*

- Primarily support for JSON
- Currently only android templates
- Missing support for search-queries

[Quelle: http://www.fhir.org]
Questions?

**Andreas Schuler, MSc.**  
Research Associate  
e-Health – Integrated Care  
FH OÖ F&E GmbH  
*Andreas.Schuler@fh-hagenberg.at*

**Barbara Franz, MSc.**  
Assistant Professor – Deputy leader of research group e-Health  
e-Health – Integrated Care  
Upper Austria University of Applied Sciences  
*Barbara.Franz@fh-hagenberg.at*

**Oliver Krauss, MSc.**  
Research Associate  
e-Health – Integrated Care  
FH OÖ F&E GmbH  
*Oliver.Krauss@fh-hagenberg.at*