Hay on FHIR

Thoughts and comments on the emerging HL7 FHIR standard, from a FHIR Evangelist

clinFHIR Profiling Walk Through

NOVEMBER 6, 2016 LEAVE A COMMENT (HTTPS://FHIRBLOG.COM/2016/11/06/CLINFHIR-PROFILING-WALK-THROUGH/#RESPOND)

So I'm in the middle of getting ready for the Furore devdays (http://www.fhirdevdays.com/) event – part of the preparation being a writing series of posts on the various stages we'll be exploring as part of that event.

Quite co-incidentally, Mark Braunstein (https://www.linkedin.com/in/mbraunstein) from Georgia Tech (https://www.linkedin.com/edu/school?id=18158) asked me to do a presentation for their FHIR class which is happening in a couple of days, so I thought it a good idea to write this walk through of the complete process from end to end – from a requirement to a profiled resource instance.

We'll return to the more detailed consideration of the steps after this.

Background

Here are the overall steps we'll follow:

1. Gather requirements
2. Build Logical Model/s
3. Build / find valuesets
4. Create Profiles
    1. Build / find Extension Definitions
    2. Build Profile
5. Build Example resources

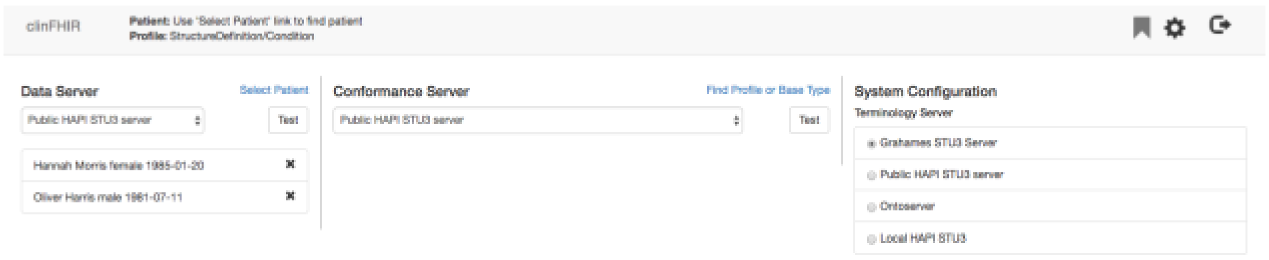
Pre requisites

There are a couple of things to set up first.

- Load clinFHIR (http://clinfhir.com/).

- Login using the icon to the upper right. You just need an email for now – I promise it's only for use within the app!
- Servers: We'll use the public HAPI STU3 server as the data and the profile server, and Grahames STU3 server as the terminology server. However, this should work on any STU3 server (and probably STU2 though I haven't tested that yet)

Here's a screen shot of the configuration:



Patient: Use 'Select Patient' link to find patient
Profile: StructureDefinition/Condition

Data Server: Public HAPI STU3 server

Hannah Morris female 1985-01-20	✖
Oliver Harris male 1961-07-11	✖

Conformance Server: Public HAPI STU3 server

System Configuration
Terminology Server

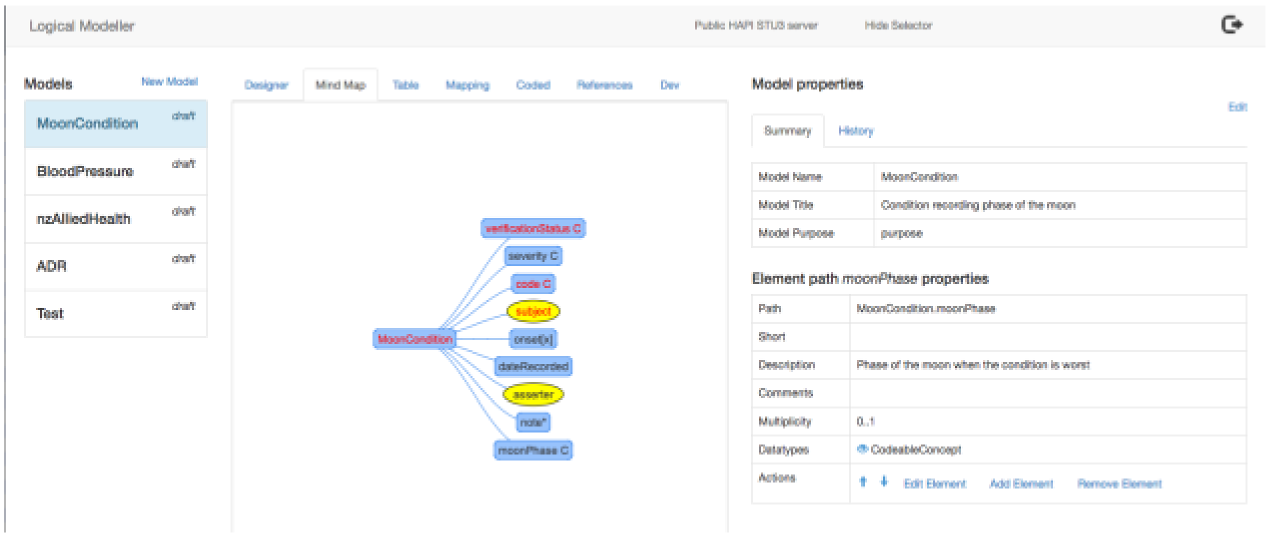
- ● Grahames STU3 Server
- ○ Public HAPI STU3 server
- ○ Ontoserver
- ○ Local HAPI STU3

Requirements

We want to add a profile that allows us to record the phase of the moon when a condition is at its worst (I've been watching Harry Potter again). There's a fixed set of values for the phase of the moon: new, waxing, full, waning. We'll also take the opportunity to remove some of the elements from Condition that we don't want to support.

Logical Model

Here's the logical model:



Models	
MoonCondition	draft
BloodPressure	draft
nzAlliedHealth	draft
ADR	draft
Test	draft

Model properties

Model Name	MoonCondition
Model Title	Condition recording phase of the moon
Model Purpose	purpose

Element path moonPhase properties

Path	MoonCondition.moonPhase
Short	
Description	Phase of the moon when the condition is worst
Comments	
Multiplicity	0..1
Datatypes	CodeableConcept
Actions	↑ ↓ Edit Element Add Element Remove Element

And the steps to produce were:

1. Create new model
2. Model type = single resource (shows the 'Initial Content' tab)
3. Initial content from Condition resource
4. Complete other fields and save

This creates a model that is basically a simplified copy of the core resource type that we can then modify in the modeler. Here is more detail (https://fhirblog.com/2016/11/05/more-on-fhir-logical-models/) on the Logical Modeler.

Analysing the model, we realize we'll need:

- An Extension Definition for an element in Condition representing the phase of the moon
- A valueSet that has the permissible values for that extension
- A profile on Condition that includes the extension definition, and removes some elements we don't want.

ValueSet

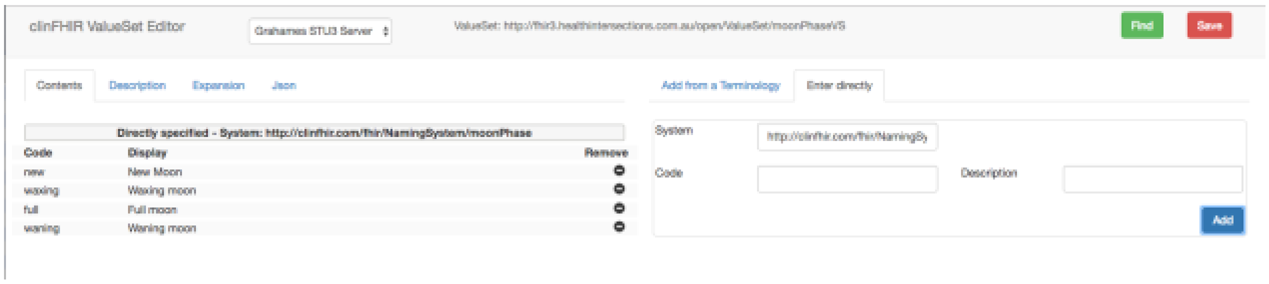
We'll need to create a valueSet for the values of the phase of the moon. This will be bound to the extension definition that we'll create next and will be stored on the terminology server (Grahames in our case). (In real life we'd go looking for an existing one that matches, but we'll assume that we're the first to think of this).

Note that we have to create the ValueSet first, as it needs to be present when we create the binding in the Extension Definition.

Steps to produce:

1. From the main clinFHIR screen click the gear icon and select 'ValueSet Editor'. The Editor screen is displayed (Note that it is in a separate browser tab to clinFHIR).
2. Click the 'New' button on the top right. A dialog appears where we can enter the name of new ValueSet and a description.
3. Enter the name (see the bottom of this post for thoughts on naming conventions) and description, then the 'check name' button that appears. A save button should appear upper right. Click it.
4. Click the 'Enter directly' tab to the upper right. Then enter the possible values into the input boxes one at a time clicking 'Add' for each one. Set the 'system' value to 'http://clinfhir.com/fhir/NamingSystem/moonPhase'
5. When all the values have been entered, click the 'Save' button to the upper right.

Here's a screen shot of the ValueSet after it has been saved.



ValueSet: http://fhir3.healthintersections.com.au/open/ValueSet/moonPhaseVS

Directly specified - System: http://clinfhir.com/fhir/NamingSystem/moonPhase

Code	Display	Remove
new	New Moon	⊖
waxing	Waxing moon	⊖
full	Full moon	⊖
waning	Waning moon	⊖

System: http://clinfhir.com/fhir/NamingSy

Code:

Description:

For a ‘real’ ValueSet we’d likely choose values from a proper Terminology like SNOMED, but for the purposes of education this approach is fine.

If we wanted to, we could always go back to the logical modeler and bind the ValueSet there as well. In practice, this is something that we’re likely to do commonly as the Logical Model is refined

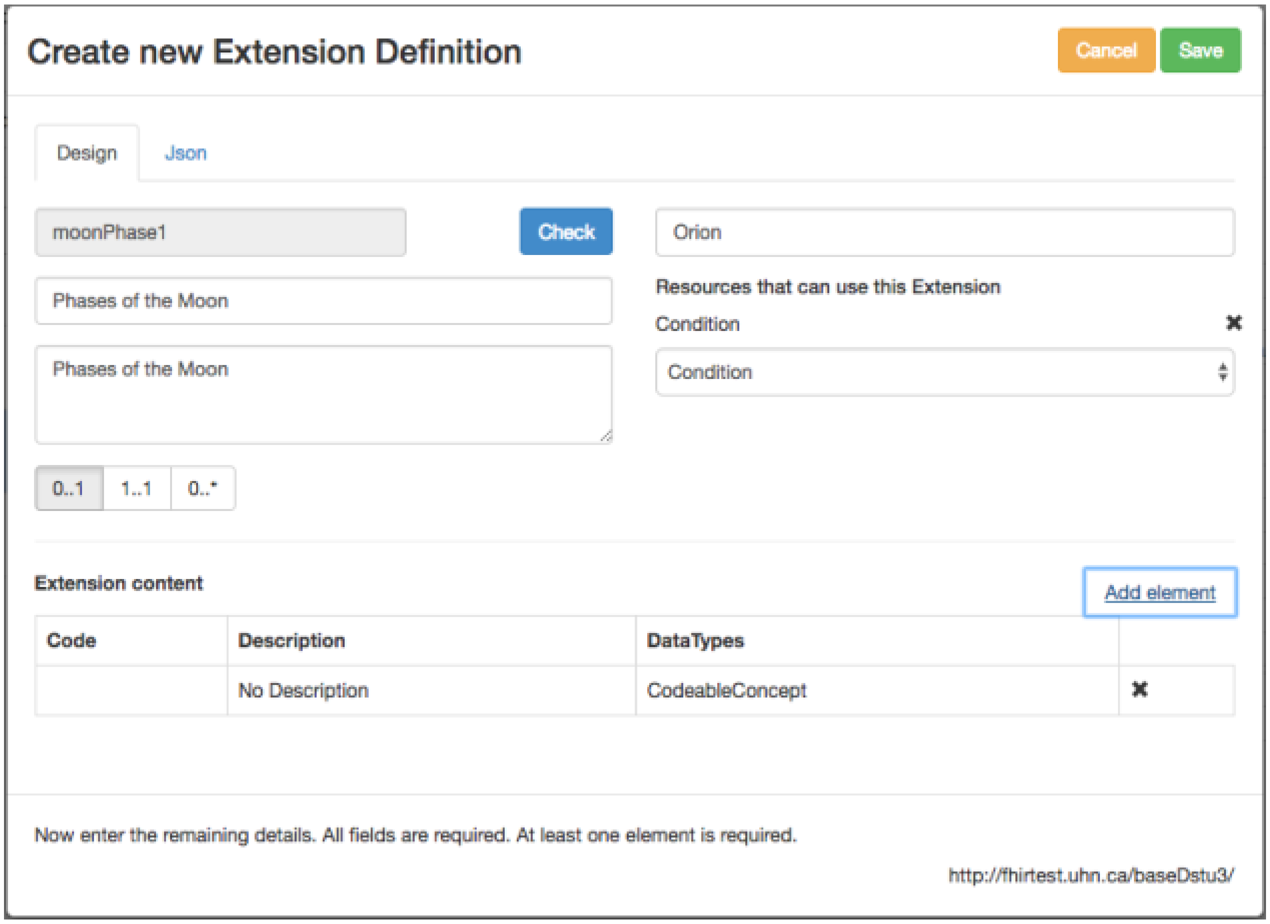
Extension Definition

The Extension definition defines the new element we are adding to the Condition. It will be a simple extension (one value only) and will be of type CodeableConcept, bound to the ValueSet we just created with a binding strength of preferred.

The steps:

1. From the main clinFHIR screen click the gear icon and select ‘Extension Definitions’. The ED Explorer screen is displayed.
2. Click the ‘new Extension Definition’ button on the top right
3. Enter a name into the dialog and click ‘check’ to make sure the name isn’t already being used. The Save button will appear (upper right) if it is new.
4. Select ‘Condition’ from the ‘Resources that can use this extension’ drop down, and fill in the description fields (short & long).
5. Click the ‘add element’ link (lower right). A dialog appears titled ‘add child element’
    1. Set the Datatype to CodeableConcept. A link labeled ‘Bind ValueSet’ appears. Click it.
    2. Another dialog appears titled ‘Bind Valueset’. Enter the name of the Valueset you created above and click ‘find’.
    3. Your ValueSet should be displayed. Click the select link alongside it, then the ‘select’ button to the upper right.
    4. The ‘Add Child’ dialog is re-displayed. Click the ‘Add’ button to the upper right. The ‘Create new ExtensionDefinition’ dialog re-appears.
6. Click the ‘Save’ button (upper right) to save the new Extension Definition.

Here’s a screen shot of the new Extension Definition just before clicking Save.


Create new Extension Definition

moonPhase1

Orion

Phases of the Moon

Resources that can use this Extension

Condition ✖

Condition

Phases of the Moon

0..1 1..1 0..*

Extension content

Code	Description	DataTypes	
	No Description	CodeableConcept	✖

Now enter the remaining details. All fields are required. At least one element is required.

http://fhirtest.uhn.ca/baseDstu3/

Profile

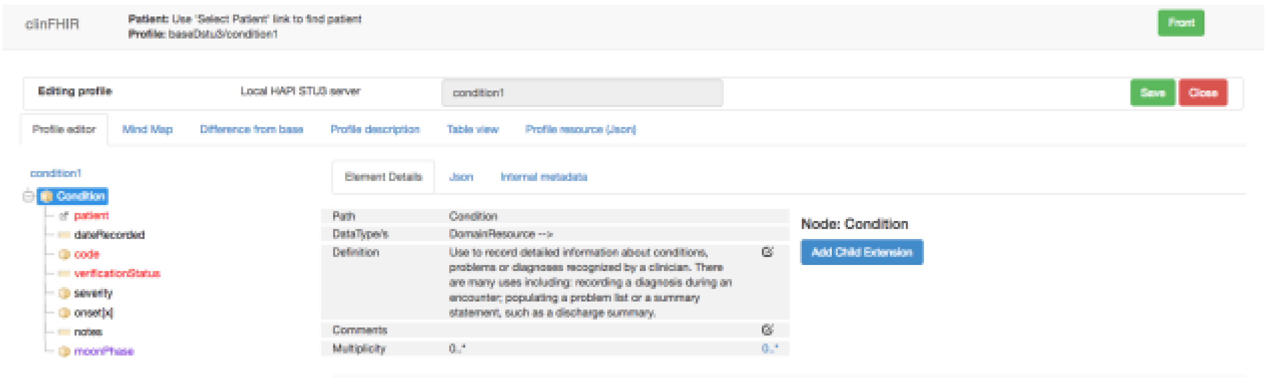
The profile is going to be the 'template' that we'll use to create the Resource Instance from (ie the real resource that is attached to a patient).

The steps:

1. From the main clinFHIR screen click the gear icon and select 'Profiles'. The Profile Explorer screen is displayed.
2. Enter your name (or something else) as the publisher. Click the 'magnifying glass' icon. There probably won't be any profiles.
3. Click the 'new Profile' button on the top right
4. Enter a name into the dialog and click 'check' to make sure the name isn't already being used.
5. Set the 'Resource Type being profiled' to 'Condition'. The Save button will appear (upper right)
6. Enter the descriptions then click save.
7. The profile explorer is displayed. Click the 'magnifying glass' icon. The new profile is listed in the left. Click it.
8. The details of the profile are displayed on the right, along with buttons at the bottom. Click the 'Details/Edit' button.
9. The profile details screen is displayed. Click the 'Edit this profile' link (upper right).
10. Click the Condition node that is on the top of the tree. A description of the node appears in the right, along with a button labeled 'Add extension'. Click it.

11. A dialog should appear with the list of extensions appropriate for this resource type, including yours. Select it in the list, then click the Select button to the upper right. The profile screen is re-displayed, with the new extension at the bottom.
12. Remove elements you don't want to support by selecting them in the tree, then clicking the 'remove from profile' button that appears.
13. When complete, click the Save button (upper right) to save the profile.

Here's the tree view of the profile:



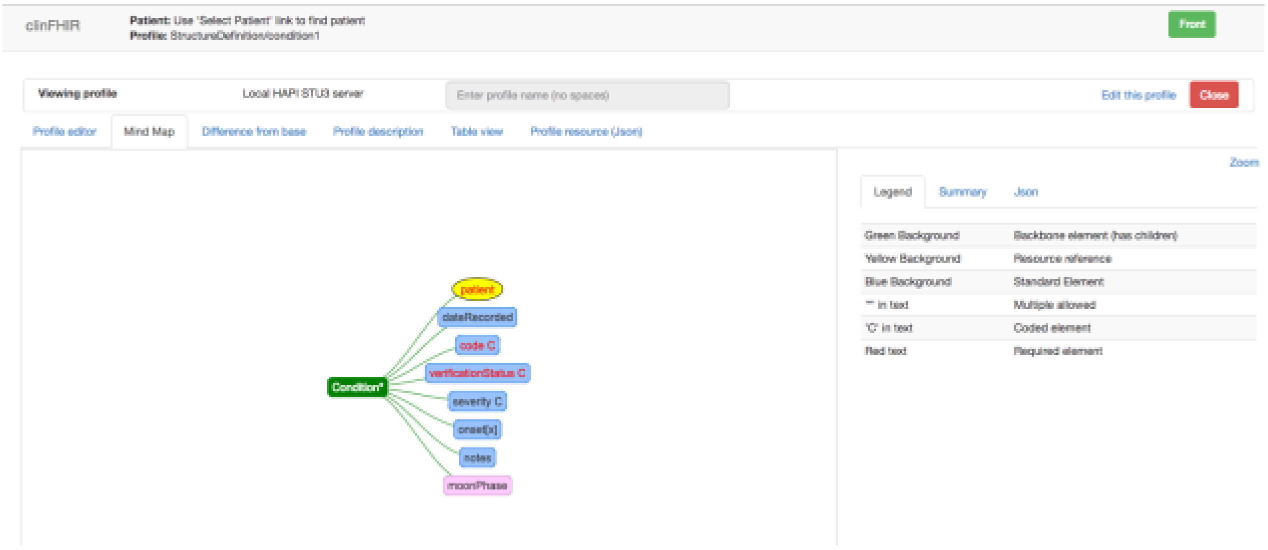
Patient: Use 'Select Patient' link to find patient
Profile: baseDstu3/condition1

Editing profile Local HAPI STU3 server condition1

Path	Condition
DataType/s	DomainResource -->
Definition	Use to record detailed information about conditions, problems or diagnoses recognized by a clinician. There are many uses including: recording a diagnosis during an encounter; populating a problem list or a summary statement, such as a discharge summary.
Comments	
Multiplicity	0..*

Node: Condition

and the mindmap (note that the mindmap is not updated as the profile is updated – I had to save the profile and re-edit it to get it. I'll fix that…)



Patient: Use 'Select Patient' link to find patient
Profile: StructureDefinition/condition1

Viewing profile Local HAPI STU3 server Enter profile name (no spaces)

Green Background	Backbone element (has children)
Yellow Background	Resource reference
Blue Background	Standard Element
'*' in text	Multiple allowed
'C' in text	Coded element
Red text	Required element

Build example resource

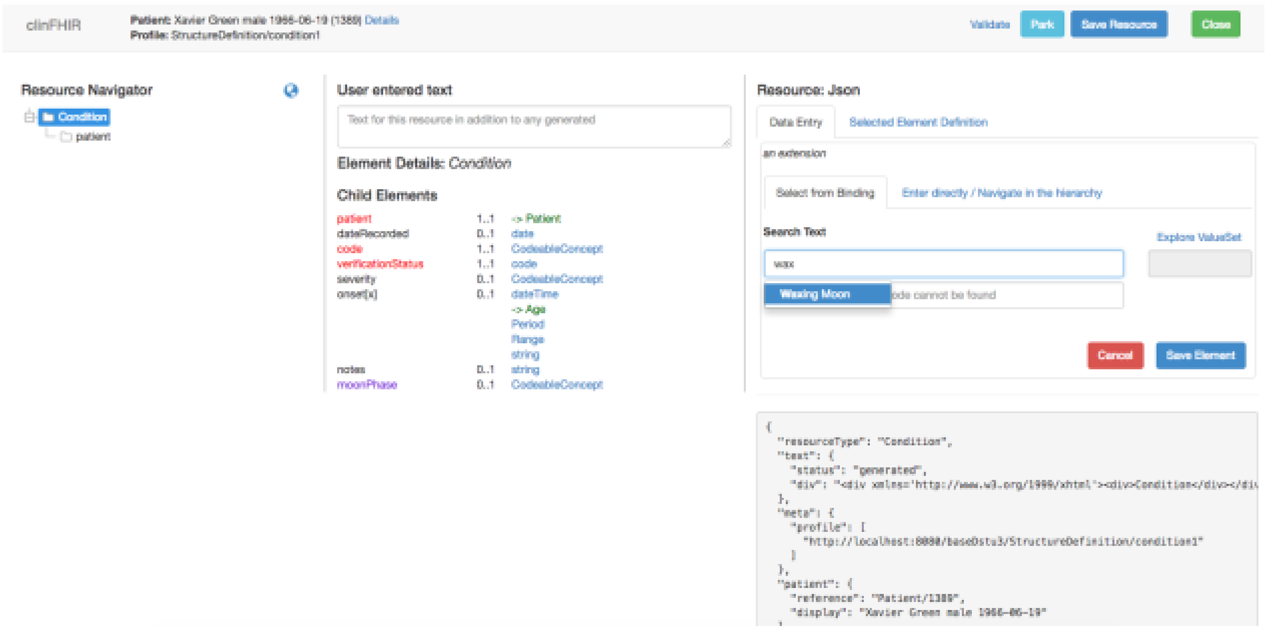
Finally you are ready to build an example resource using this profile.

Steps:

1. Re-load clinFHIR. In fact, it's often a good idea to do this every now & then.
2. The profile you created should be in the middle list. Click to select it.

3. Select a patient from the left list (or create a new one). The 'New Resource Instance' button should appear to the upper right.
4. Click the 'New Resource Instance' button. The resource builder screen should be displayed, with the profiled resource it it.
5. Click on the moon phase extension datatype. The CodeableConcept editor should appear in the upper right. Entering the first 3 letters of any of the values from the ValueSet should trigger the autocomplete. Allowing you to select from the list of possible values. You can also click the 'Explore ValueSet' link to look at the whole ValueSet. The 'Compose' tab in the dialog shows the permissible values. (The Expand tab will not work).

Here's a screen shot of the resource editor with the profile selected and a value for the extension being selected.



Patient: Xavier Green male 1966-06-19 (1389) Details
Profile: StructureDefinition/condition1

User entered text

Text for this resource in addition to any generated

Element Details: Condition

Child Elements

patient	1..1	-> Patient
dateRecorded	0..1	date
code	1..1	CodeableConcept
verificationStatus	1..1	code
severity	0..1	CodeableConcept
onset[x]	0..1	dateTime
		-> Age
		Period
		Range
		string
notes	0..1	string
moonPhase	0..1	CodeableConcept

Resource: Json

an extension

Search Text

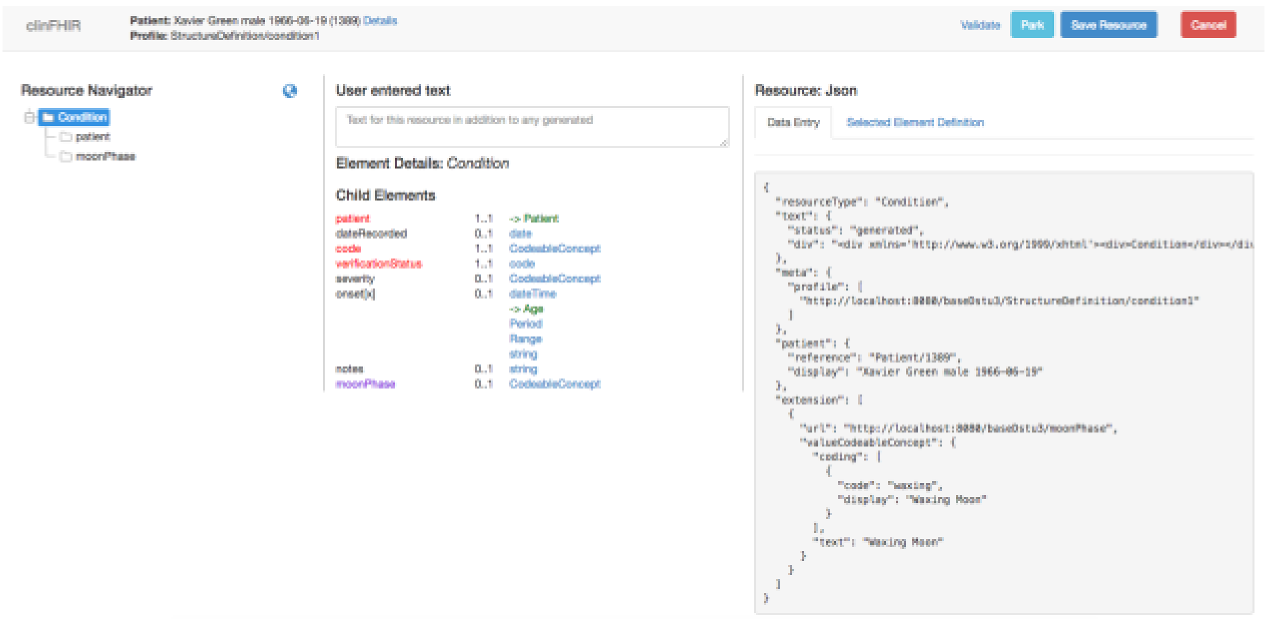
wax

Waxing Moon

ode cannot be found

```
{
  "resourceType": "Condition",
  "text": {
    "status": "generated",
    "div": "<div xmlns='http://www.w3.org/1999/xhtml'><div>Condition</div></di
  },
  "meta": {
    "profile": [
      "http://localhost:8080/baseDstu3/StructureDefinition/condition1"
    ]
  },
  "patient": {
    "reference": "Patient/1389",
    "display": "Xavier Green male 1966-06-19"
```

And here's the resource instance itself. Note that the instance has a profile tag indicating that it claims conformance to that profile.



Patient: Xavier Green male 1966-06-19 (1389) Details
Profile: StructureDefinition/condition1

User entered text

Text for this resource in addition to any generated

Element Details: Condition

Child Elements

patient	1..1	-> Patient
dateRecorded	0..1	date
code	1..1	CodeableConcept
verificationStatus	1..1	code
severity	0..1	CodeableConcept
onset[x]	0..1	dateTime
		-> Age
		Period
		Range
		string
notes	0..1	string
moonPhase	0..1	CodeableConcept

Resource: Json

```
{
  "resourceType": "Condition",
  "text": {
    "status": "generated",
    "div": "<div xmlns='http://www.w3.org/1999/xhtml'><div>Condition</div></di
  },
  "meta": {
    "profile": [
      "http://localhost:8080/baseDstu3/StructureDefinition/condition1"
    ]
  },
  "patient": {
    "reference": "Patient/1389",
    "display": "Xavier Green male 1966-06-19"
  },
  "extension": [
    {
      "url": "http://localhost:8080/baseDstu3/moonPhase",
      "valueCodeableConcept": {
        "coding": [
          {
            "code": "waxing",
            "display": "Waxing Moon"
          }
        ],
        "text": "Waxing Moon"
      }
    }
  ]
}
```

And that concludes the walk through. I noted a number of things as I did this:

- There is not a lot of consistency between the different components – a result of the organic growth of clinFHIR. At some point, we'll need to give it some UX attention, which means that over time the details of this walk through will change.
- If you have odd things happening then re-load the page. ClinFHIR is quite complicated under the hood and can get confused. This can happen especially when you are modifying a profile – use the 'clear Profile cache' from the gear menu to clear everything out and re-select the profile (in this walkthrough you'd look for profiles on Condition and yours should be in the list).
- It's easy to forget where the different resources are – especially if you use different servers. Remember:
  - Data server has the resource instance
  - Conformance server the extension definition and the profile (they are both StructureDefinition resources)
  - Terminology server has the ValueSet
- It's also easy to get confused between the names of ExtensionDefinitions and Profiles – as they are the same resource on the same server. I suggest a naming convention like {your name}{name of object}{object type} – eg dhMoonPhaseSD for the structure definition and dhMoonPhaseProfile for the profile.
- There are a lot of moving parts here! Do let me know through the FHIR chat of any issues you uncover.

And a reminder that clinFHIR is not a fully featured authoring tool like Forge – it's primarily a training tool, which means that functionality is confined to a specific sub-set (especially in profiling) and that it works best with resources created with the tool. (and can be temperamental at times…)

If you do try this out, do let me know of any errors or inconstancies you find.

FILED UNDER CLINFHIR, CLINICAL

About David Hay
I'm a Product Strategist at Orion Health, Chair of HL7 New Zealand and co-Chair of the FHIR Management Group. I have a keen interest in health IT, especially health interoperability with HL7 and the new FHIR standard.