06 - Test Definition - Connectathon Track

Test Driven Development II - Advanced - Exercise

Track: Test Driven Development II - Advanced
Track lead: **Richard Ettema**
Email: richard.ettema@aegis.net

During this hands-on session of the Test Driven Development II - Advanced tutorial we will examine the proposed HL7 FHIR Connectathon 17 Test Track for Patient Match and begin the analysis and design of a TestScript based on one of the track's test scenarios.

TestScript Definition Approach

TestScript definition begins with defining the use case that will be tested. Once defined, the use case can then be analyzed and used to define corresponding test case(s) or TestScript(s). A useful process to follow is to simply follow the TestScript structure and "fill in the blanks". Here are some guidelines for filling them in:

Fixtures
A set of resource instances that will be used during TestScript execution.
- Represent the expected, known and valid test data
- Will typically require use case domain knowledge to define

Profiles
A set of FHIR profiles (StructureDefinitions) that will be used during TestScript execution of profile asserts.
- Each profile defines a StructureDefinition instance by URI where the URI SHALL reference a known FHIR profile
- Use the FHIR Resource base profiles for validation against the base FHIR specification
- If external profiles are needed, you will need to coordinate with Touchstone Support

Variables
A set of expressions whose evaluations will be used in substitutions during TestScript execution.
- Extracts data values from static fixtures (defined in the fixtures section/element)
- Extracts data values from dynamic fixtures (operation request and/or response payloads)
- Used in operations and asserts to provide context relevant data values; e.g. operation parameters, assert comparison values

Setup
The setup section is executed to establish the testing environment.
- The purpose of the setup section is typically to pre-load required data or delete data on the FHIR system(s) under test
- The setup operations are executed once before all the tests are run
- All operations in a setup section (including assertions) must complete successfully for the subsequent tests to be executed

Test
Each test section is a set of actions – operation and assert – that are executed and evaluated.
- Operations are always executed regardless of any asserts that may follow
- When an operation or assertion fails, subsequent operations and assertions are not executed
- When a test fails, subsequent tests are still executed
- Because tests are typically interdependent, it is best practice to fix the cause of the first test failure instead of analyzing all test failures for a single test execution

Patient Match - Scenario 1 - Analysis

http://wiki.hl7.org/index.php?title=201801_Patient_Match

First Test Scenario - Match existing patient resource (same server)
Action: Search for an existing patient using an entire patient resource from the same server
Precondition: A patient record exists on the server
Success Criteria: Only that patient record is returned
Bonus point 1: no other patient resources are returned
Bonus point 2: a confidence weighting of 1.0 is returned

Patient Match - Scenario 1 - Re-defined

Using the TestScript Definition Approach described on page 1 we initially define the section contents with the following information. Note, we have added an Actors section to help identify the systems and the interactions between them.

Actors

- Touchstone (T) - Test Platform, default FHIR Client
- Patient System (PS) - FHIR Server supporting Patient resource create, read, update, delete, search and $match operations

Test Data (Fixtures)

- patient-create1 - Patient static fixture used to create the Patient 1 instance on (PS); this patient represents the certain (1.0) match
- patient-create2 - Patient static fixture used to create the Patient 2 instance on (PS)
- patient-create3 - Patient static fixture used to create the Patient 3 instance on (PS)
- patient-create4 - Patient static fixture used to create the Patient 4 instance on (PS)
- patient-create5 - Patient static fixture used to create the Patient 5 instance on (PS)
- parameters-match - Parameters static fixture used as $match operation payload; resource parameter defined with copy of 'patient-create1' fixture

Variables

- patientIdentifier - static variable; source = patient-create1
- patientFamilyName - static variable; source = patient-create1
- patientGivenName - static variable; source = patient-create1
- patientGender - static variable; source = patient-create1
- patientBirthDate - static variable; source = patient-create1

Setup

- Delete any existing Patient resources that match the new Patients to be created on (PS)

Test 01 - Create Patients

Create new Patient resource instances on the target Patient System

- (T) sends Patient resource to (PS)
  - Operation 'create'
  - HTTP POST [SS Base URL]/Patient
  - Sent Patient payload is the static fixture '*patient-create1*'
- (PS) responds with conformant FHIR API success

- Repeat previous steps to create the Patient 2-5 resource instances on the target Patient System

Test 02 - Match Patient

Execute the Patient $match operation on the target Patient System searching the new Patient resource instances created in *Test 01*; expected outcome is a single match on the Patient 1 instance

- (T) sends Patient $match request to (PS)
    - Operation '$match'
    - HTTP POST [PS Base URL]/Patient/$match
    - Sent Parameters payload is the static fixture '*parameters-match*'
- (PS) responds with conformant FHIR API success and payload of a searchset Bundle containing the found Patient
    - Validate returned Bundle type is '*searchset'*
    - Validate contents of returned Bundle first entry matches the Patient 1 instance from *Test 01*
        - Use the variables populated from the static fixture to compare against the response payload
    - Validate returned Bundle first entry contains entry.search extension for 'match-grade'
    - [Bonus point 1] Validate returned Bundle total is one (1)
    - [Bonus point 1] Validate returned Bundle entry.count() is one (1)
    - [Bonus point 2] Validate returned Bundle first entry contains confidence weighting element (entry.search.score) with a value of '1.0'

TestScript Implementation

The TestScripts (one each for XML and JSON formats) that implement the definition above are found in the ***FHIRSandbox/AEGIS/FHIR3-0-1-DevDays18/TDD-2-Advanced/06-TestDefinition*** *Test Definitions folder.*

***

Have fun! And, remember to ask for help if you get stuck.

***

Please note:

- The exercises can be made in the hands-on area, where each track has its own table, indicated with a track sign. The track lead will be present for guidance and review.
- Exercises will only be discussed or reviewed during the HL7 FHIR DevDays 18 in Boston
- Any questions or remarks after the conference can be addressed in the FHIR chat on Zulip: https://chat.fhir.org