Test Driven Development I - Intro - Exercises

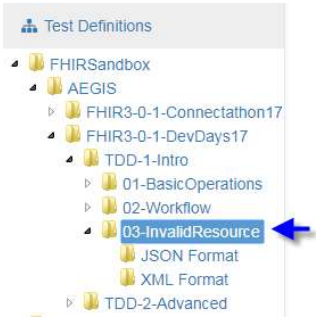
03 - Touchstone TestScript Execution - Invalid Resource

Track lead: Richard Ettema

During this hands-on session of the Test Driven Development I - Intro tutorial we will explore TestScript Execution within the Touchstone Project environment. This exercise examines a single create FHIR operation where the request payload is an invalid Patient resource instance and is processed by the FHIR Validation Engine. The following generic steps performed in sequence will be used to execute the TestScript:

1. Create and execute a Test Setup
2. Examine the Test Execution interface
3. Examine the TestScript Execution interface

TestScripts for this exercise are in the **FHIR3-0-1-DevDays17/TDD-1-Intro/03-InvalidResource Test Definitions*
Please refer to the Touchstone User Guide section or the previous exercise **Touchstone TestScript Execution - Basic Operations for help with the generic steps.*



1. Create and execute a Test Setup
Related online document: https://touchstone.aegis.net/touchstone/TouchstoneUserGuide#page=12, Section 'Executing Tests'.

2. Test Execution Interface
3. TestScript Execution Interface
Related online document: https://touchstone.aegis.net/touchstone/TouchstoneUserGuide#page=15, Section 'Test Execution Results'.

TestScript Description

All TestScripts for this exercise use the Patient resource type.

03-InvalidResource
Test FHIR create operation using an invalid Patient resource instance. Touchstone will send an invalid Patient fixture to the system under test.
Features

- Illustrates a simple negative test case.
- Uses the FHIR Patient resource base profile; *asserts invoke the FHIR Validation Engine using profiles*
- ***Note: This test is expected to fail.***

The focus of this test is to illustrate a failure based on the FHIR Validation Engine. The profile assert is testing the HTTP request payload which is an intentionally invalid Patient resource instance sent by Touchstone acting as the FHIR client system.

- *Does the profile assert against the HTTP request payload make sense?*
- Examine the TestScript asserts following the create operation; *how could the profile assert be handled so the following assert(s) would evaluate? What additional asserts would be useful/needed to provide better test coverage?*
- *How could this TestScript be modified to represent a more real-world scenario?*

Have fun, and remember to ask questions if you need help!