sb11-test-auth

The Test Authoring Component is responsible for test (assessment) design, construction, and publishing.

The authenticated and authorized user can define subjects, computation rules (scoring functions), item selection algorithms, and build publication relationships, all of which are then used to author tests (assessments).

Authoring of a test involves construction and configuration of segments, blueprints, test item import, forms, scoring rules, performance levels, reporting measures, and finally garnering approvals from users with appropriate authorization. The result is a test specification that can be published to Test Specification Bank (tsb) and made available further downstream for Test Registration, Test Packaging, and Test Delivery.

Usage

Rest Module

The REST module is a deployable WAR file (`test-auth.rest-VERSION.war`) that provides REST endpoints that can be used to access and modify Test Authoring data.

In order to run and use the REST WAR application, several supporting applications must be running and accessible: sb11-program-management, sb11-monitoring-alerting (mna), sb11-test-item-bank (tib), and sb11-test-spec-bank (tsb).

In addition, the following opentestsytem applications are also required at runtime: opentestsystem permissions, opentestsystem SSO, and opentestsystem corestandards.

REST layer setup must be performed before deploying the WAR to a Tomcat-compatible application server. Specifically, virtually all of the startup and runtime parameters that the REST module needs are stored in sb11-program-management using its profile property configuration feature.

To execute the REST module and connect to sb11-program-management, runtime parameters are passed to the Tomcat server running the sb11-test-auth REST module:

`-Dspring.profiles.active="progman.client.impl.integration,mna.client.integration"`

This runtime parameter specifies which spring profile to use for the program management and monitoring & alerting client interfaces.

`-Dprogman.baseUri="http://sb11-progman-stable.drc-ec2.com/rest/"`

This runtime parameter specifies the REST endpoint of a running sb11-program-management instance that will be accessed during REST module startup.

`-Dprogman.locator="testauth,dev,dev_testauth_overrides"`

This runtime parameter specifies the property configuration set stored in the running sb11-program-management instance which contains all of the needed properties to get this instance of sb11-test-auth running. The third optional value example '`dev_testauth_overrides`' is a useful feature that allows for overrides: in this case, all properties contained in the property group '`testauth`' for level '`dev`' are used by default, except where property group '`dev_testauth_overrides`' has an overriding property value.

The REST module contains all of the domain beans used to model the Test Authoring data as well as code used as search beans to create Mongo queries.

The REST module is also responsible for persistence of application data. This includes all business rules, validation, XML configuration, and publishing.

Frequently during the authoring of a test, items are searched and imported from sb11-test-item-bank. However, with more and more items imported into an assessment, the publishing of that assessment can be affected when the number of items gets extremely large, because the XML grows geometrically as more and more metadata about the item metadata is tallied and included. Extremely large item pool also can have negative effects downstream in Test Spec Bank, Test Packager, Test Administration, and Test Delivery. To mitigate this, Test Authoring is built to look for and utilize a property saved into Program Management within the testauth profile property configuration, with a key of "testauth.item.count.max.limit" and numeric integer value representing the maximum allowable number of items per assessment (e.g. '30000'). If the property is not found, or is not a valid integer, it will not be used, and the item import has no upper limit.

Once an assessment has garnered all necessary approvals it is transformed into XML and published to sb11-test-spec-bank; however, during publishing, validation of that XML by use of the related DTD is not performed by default, as it is a very costly operation in both time and memory usage. However, if needed it can be activated during application restart by saving a property into Program Management within the testauth profile property configuration, with a key of "testauth.dtd.validation" and value of "true".

Webapp Module

The Webapp module is a deployable WAR file (`test-auth.webapp-VERSION.war`) that provides the rich UI for Test Authoring functionality. As with several other sb11 applications, this is a single-page application (SPA) built using AngularJS for a robust, reactive user interface. The Webapp module uses the REST module for all data access, but this is a runtime dependency through a REST endpoint and not a direct code dependency.

Build

These are the steps that should be taken in order to build all of the Test Authoring related artifacts.

Pre-Dependencies

- Mongo 2.0 or higher
- Tomcat 6 or higher
- Maven (mvn) version 3.X or higher installed
- Java 7
- Access to sb11-shared-build repository
- Access to sb11-shared-code repository
- Access to sb11-rest-api-generator repository
- Access to sb11-program-management repository
- Access to sb11-monitoring-alerting repository
- Access to sb11-test-spec-bank repository
- Access to sb11-test-auth repository

Build order

Dependencies

Test Authoring has a number of direct dependencies that are necessary for it to function. These dependencies are already built into the Maven POM files.

Compile Time Dependencies

- Apache Commons IO
- Apache Commons Beanutils
- Jackson Datatype Joda
- Google Guava
- Hibernate Validator
- Apache Commons File Upload
- Jasypt
- SB11 Shared Code
    - Logback
    - SLF4J
    - JCL over SLF4J
    - Spring Core
    - Spring Beans
    - Spring Data MongoDb
    - Mongo Data Driver
    - Spring Context
    - Spring WebMVC

    - Spring Web
    - Spring Aspects
    - AspectJ RT
    - AspectJ Weaver
    - Javax Inject
    - Apache HttpClient
    - JSTL API
    - Apache Commons Lang
    - Joda Time
    - Jackson Core
    - Jackson Annotations
    - Jackson Databind
    - SB11 REST API Generator
    - JSTL

Test Dependencies

- Spring Test
- Hamcrest
- JUnit 4
- Mockito
- Fongo
- Podam
- Log4J over SLF4J

Runtime Dependencies

- Servlet API
- Persistence API