sb11-test-spec-bank

The Test Specification Bank Component is responsible for test (assessment) specification storage, searching, and initiation of packaging.

The authenticated and authorized user can search for test specifications that were published from sb11-test-auth (Test Authoring), view the actual specification XML, and initiate a packaging of that XML via call out to sb11-test-packager (Test Packaging), which will bundle the XML with all related Test Item data to be compressed and stored in a remote Secure FTP host location.

Usage

Rest Module

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

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

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-spec-bank 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="tsb,dev,dev_tsb_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-spec-bank running. The third optional value example '`dev_tsb_overrides`' is a useful feature that allows for overrides: in this case, all properties contained in the property group '`tsb`' for level '`dev`' are used by default, except where property group '`dev_tsb_overrides`' has an overriding property value.

Domain Module

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

Persistence Module

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

Upon receiving a new XML test specifcation to be stored, validation of that XML by use of the related DTD is not performed as it is a very costly operation in both time and memory usage. However, if needed it can be activated during application startup by saving a property into Program Management within the tsb profile property configuration, with a key of "tsb.dtd.validation"

and value of "true".

Webapp Module

The Webapp module is a deployable WAR file (`test-spec-bank.webapp-VERSION.war`) that provides the UI for Test Spec Bank 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.

Client Modules

There are two modules that make up the client:

- The Client Interfaces module contains the interface classes for the client, AOP advice and point cuts, bootstrap initialization
- The Integrated Client sends all messages to the advertised REST endpoints from the REST module

In order to use one of the client implementations:

- Add the Maven dependencies to the POM of the project that is using Test Spec Bank.

```
<dependency> <groupId>org.opentestsystem.authoring</groupId> <artifactId>test-spec-bank-client<artifactId> <version>${tsb-client.version}<version> </dependency>
```

- Add a context initializer class name to the web.xml

```
<context-param> <param-name>contextInitializerClasses</param-name> <param-value>org.opentestsystem.authoring.testspecbank.client.init.TestSpecBankClientContextInitializer</param-value> </context-param>
```

Build

These are the steps that should be taken in order to build all of the Test Spec Bank 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

Build order

Dependencies

Test Spec Bank 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
- Flapdoodle
- Podam
- Log4J over SLF4J

Runtime Dependencies

- Servlet API
- Persistence API