Test Plan Template

Blizzard Test Engineer Interview

|  |  |
| --- | --- |
| Full Name | Mitchell Turlington |
| Email Address | mitchell.turlington@gmail.com |
| Date | 04/14/2016 |

# Scope

In this section, you should identify what testing will be covered by this test plan, what will be covered elsewhere, and what will not be tested. Please indicate why you have scoped the test plan as you did. After reading this section, readers should be able to clearly indicate the components for which the test engineer will be responsible.

What will be tested is retrieving the default id item for both item and item set, and retrieving items that contain whatever fields the default item is missing (item spells, allowable classes, bonusSummary). The language differences will be tested, and jsonp will be tested. A “bad” item id will be tested, and whether or not the items in an item set correspond with real items will be tested (and vice versa). What won’t be tested is whether or not any item goes “missing” (an id worked previously but has since stopped working), because there are far too many items for this to be done within a reasonable time by one machine. The scope is defined this way so that we can be reasonably sure the API is well-tested, within a short amount of time. If Gson were excluded, we would have much more purity in our tests, but I took this part of the e-mail to heart: “If you find new or interesting testing libraries or strategies, I’d love to hear about it in the document!” Gson certainly seems interesting to me!

# Test Strategy

In this section, you should describe your approach of testing. Please include the types of tests to be performed, and base your strategy on project goals and assessed risk. After reading this section, readers should have a clear understanding of the types of tests to be performed.

Two approaches to testing are taken: string comparison on the json strings returned, and auto-conversion to Java objects using Gson before comparing the created object values with expected values. The json string allows for simple and complete comparison, but the java objects make it easier to write/read for the developer when comparing single aspects of the items (id, name, etc). To reiterate: the API itself is tested *mostly* through string comparison. The gson auto-conversion is tested separately, but also used to help test language differences on the returned json strings.

The weakness of the design is that the gson testing and json testing are slightly co-dependent. One failing can cause parts of the other to fail, but it should be clear as to which is the root cause.

# Entry and Exit Criteria

In this section, you should identify the criteria for test entry as well as the criteria that identify when to stop testing. Be as specific as possible. After reading this section, readers should be able to identify when the product is ready to be tested and when enough testing has been performed.

Entry Criteria includes: all APIs being hit with valid and invalid input for each parameter, each field that can be returned is checked against an item that contains it, and languages should only change language-specific text. Gson objects should contain expected values that match up with the returned json strings. Exit Criteria includes: code coverage of 90%+, and all tests pass.

# Test Cases

In this section, you should include a list of test cases for each test type defined in the Test Strategy section. Test cases should include input values, conditions, and expected results. After reading this section, readers should know precisely what is to be tested and the expected results of each test case.

* Default item id should give back expected json.
* Default item set id should give back expected json.
* Non-existent ids for both should return the expected FileNotFoundException.
* Each language for the default id should give back expected json.
* Each language should change language-dependent text, and nothing else.
* Jsonp should wrap the results in the expected fashion.
* An invalid language should return expected results (turns out to be English).
* A missing language value should return expected results (turns out to also be English).
* A bad jsonp should return expected results (turns out to be 403 – Account Inactive??)
* Bad API key should return expected results (403 error).

For Gson:

* Items should auto-convert to Java objects with the appropriate filled values.
  + This should be done for an item with each type of field, as the default item does not fill all fields.
  + Use gson to easily retrieve item ids for item sets, and check them to be actual items. Do the same for the item set retrieved from item ids.
  + Make sure language values are filled correctly for these, as well.

# Dependencies

In this section, you should identify any significant constraints on testing as well as identify any tools on which your test cases will be dependent.

* Gson
* Log4j
* Junit
* Maven
* Java 8