Running Item Scoring Engine

1. Supported Item Types:

Currently the following item formats may be scored in Item Scoring Engine:

|  |  |  |  |
| --- | --- | --- | --- |
| MC | MI | QTI | EBSR |
| HTQ | GI | EQ | TI |
| ER | NL |  |  |

Some of these item types require additional resources or libraries e.g. ER and EQ

Scorers for these are configured by providing item-format-to-item-scorer mapping. All of these mappings go into servlet-context.xml file as shown in figure 1 below.

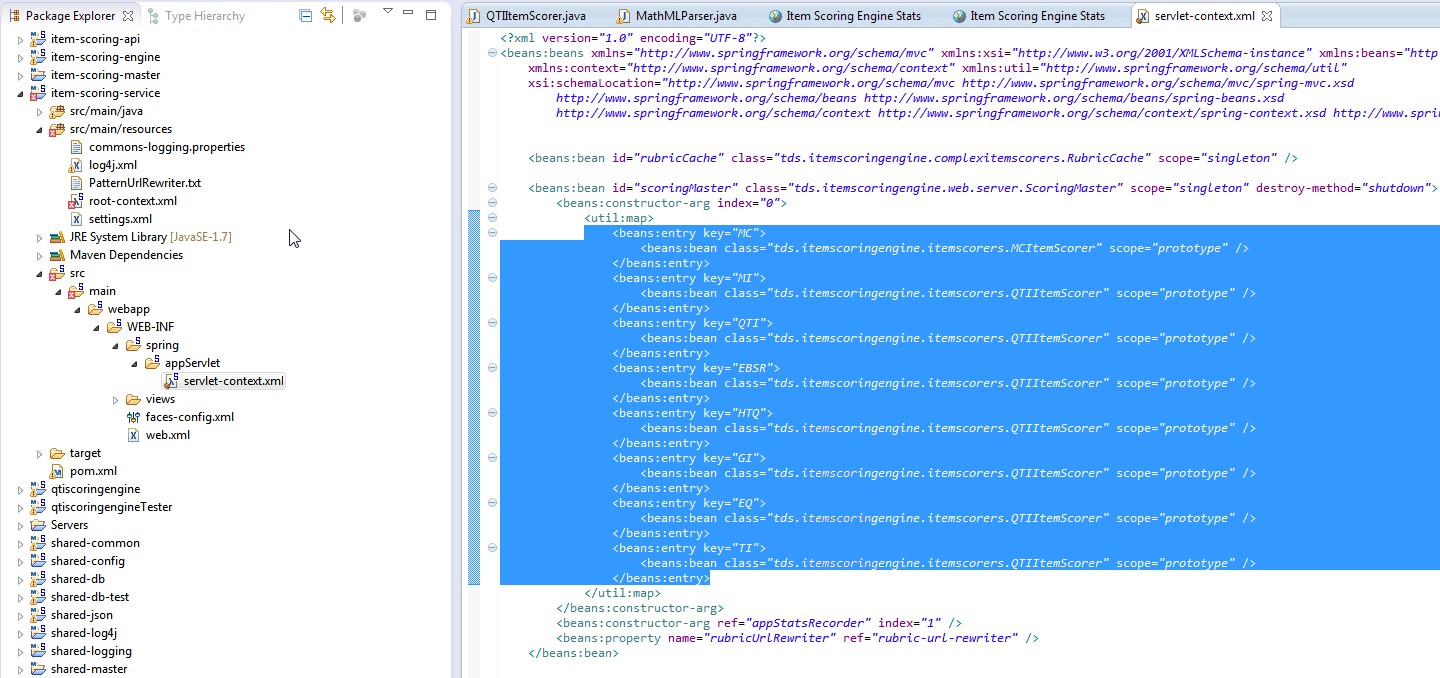


Figure 1: Item scorers’ configuration

All item scorers need to implement the IItemScorer interface. As part of the publicly accessible libraries there are two that have been provided: QTIItemScorer and MCItemScorer. More detailed discussion of the specifics of IItemScorer interface is beyond the score of this document.

1. Additional Requirements - EQ

To score equation items there is an additional requirement to provide access to a server running the custom equation scoring code that makes use of SymPy libraries. This particular script is stored under the *sympy-scripts* subfolder in the checked out code: EqScoringWebService.py.

Python 2.7, Sympy 0.7.1 and Bottle 0.12.8 are prerequisites for running this script.

This script by default runs on port 8080. You would need to change that by modifying the following line to reflect the desired port:

|  |
| --- |
| run(host='localhost', port=8080, debug=True) |

1. Dependencies

For developers: other than any additional requirements for supporting particular item formats, the only mandatory libraries required that are provided by AIR may be found in the SharedMultiJar project. The Mercurial repository of that may be found at <https://bitbucket.org/sbacoss/sharedmultijardev>. If you are using stable releases then all dependencies will be taken care of by Maven.

1. Modules

Currently the item-scoring-service is undergoing massive development and the contents of this section are prone to change. Currently, there are 6 maven modules under this source tree.

|  |  |
| --- | --- |
| **Module / Project** | **Purpose** |
| Item-scoring-api | Provides abstractions for all important classes. It is the minimum library required if you intend to implement an IItemScorer interface. |
| qtiscoringengine | Contains all the core libraries for QTI item scoring. |
| item-scoring-engine | Provides implementations of all important interfaces and is required for running the full web service. |
| item-scoring-service | The main web project. This is the one that you will be deploying. All configuration changes need listed in sections below need to be made to files under this project. |
| qtiscoringengineTester | An internal test harness. The details of this are beyond the scope of this document. |
| item-scoring-student-simulator | Will be removed or modified heavily in the future. Please ignore. |

1. Sample requests

Once item scoring service is up and running it exposes a web service end point to make calls. If the site is running on localhost at port 8081 at the context “item-scoring-service” then the web service may be access at the URL <http://localhost:8081/item-scoring-service/Scoring/ItemScoring>.

Below are a few sample requests for different item types. The variables for each request are identified in color codes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Item Id** | **Item Format** | **Student Response** | **Rubric Path** | | |
| EQ | <ItemScoreRequest > <ResponseInfo itemIdentifier="item1" itemFormat="EQ"> <StudentResponse encrypted="true"><![CDATA[ <response><math xmlns="http://www.w3.org/1998/Math/MathML"> <mstyle displaystyle="true"> <mn>1</mn> <mn>1</mn> <mn>5</mn> <mn>0</mn> <mn>0</mn> </mstyle> </math></response> ]]></StudentResponse><Rubric type="Uri" cancache="true" encrypted="true">file:///C:/WorkSpace/JavaWorkSpace/TinyScoringEngine/DataFiles/item1.qrx</Rubric><ContextToken><![CDATA[xxxxxxx]]></ContextToken> </ResponseInfo></ItemScoreRequest> |
| MI | <ItemScoreRequest > <ResponseInfo itemIdentifier="item2" itemFormat="MI"> <StudentResponse encrypted="true"><![CDATA[ <itemResponse><response id="RESPONSE"><value>1 b</value><value>2 a</value><value>3 a</value></response></itemResponse> ]]></StudentResponse><Rubric type="Uri" cancache="true" encrypted="true">file:///C:/WorkSpace/JavaWorkSpace/TinyScoringEngine/DataFiles/item2.qrx</Rubric><ContextToken><![CDATA[xxxxxxx]]></ContextToken> </ResponseInfo></ItemScoreRequest> |
| GI | <ItemScoreRequest > <ResponseInfo itemIdentifier="item3" itemFormat="GI"> <StudentResponse encrypted="true"><![CDATA[ <?xml version="1.0" encoding="utf-16"?> <!-- MACHINE GENERATED 4/28/14 14:30 PM. DO NOT EDIT --> <!DOCTYPE AnswerSet [ <!ELEMENT AnswerSet (Question+)> <!ELEMENT AtomicObject (#PCDATA)> <!ELEMENT EdgeVector (#PCDATA)> <!ELEMENT GridImageTestPoints (TestPoint\*)> <!ELEMENT LabelList (#PCDATA)> <!ELEMENT Object (PointVector,EdgeVector,LabelList,ValueList)> <!ELEMENT ObjectSet (Object,AtomicObject+)> <!ELEMENT PointVector (#PCDATA)> <!ELEMENT Question (QuestionPart)> <!ATTLIST Question id NMTOKEN #REQUIRED> <!ELEMENT QuestionPart (LabelList,GridImageTestPoints,ObjectSet)> <!ATTLIST QuestionPart id NMTOKEN #REQUIRED> <!ELEMENT TestPoint (#PCDATA)> <!ELEMENT ValueList (#PCDATA)> ]> <AnswerSet><Question id=""><QuestionPart id="1"><ObjectSet><RegionGroupObject name="group" numselected="1"><RegionObject name="a1" isselected="false"/><RegionObject name="a2" isselected="false"/><RegionObject name="a3" isselected="true"/><RegionObject name="a4" isselected="false"/><RegionObject name="a5" isselected="false"/><RegionObject name="a6" isselected="false"/><RegionObject name="a7" isselected="false"/><RegionObject name="a8" isselected="false"/></RegionGroupObject></ObjectSet><SnapPoint></SnapPoint></QuestionPart></Question></AnswerSet> ]]></StudentResponse><Rubric type="Uri" cancache="true" encrypted="true">file:///C:/WorkSpace/JavaWorkSpace/TinyScoringEngine/DataFiles/item3.qrx</Rubric><ContextToken><![CDATA[xxxxxxx]]></ContextToken> </ResponseInfo></ItemScoreRequest> |

Additionally, there are other parameters that are part of the request but they won’t be discussed here.

These score requests are nothing but serialized versions of tds.itemscoringengine.ItemScoreRequest class instances. Similarly the scoring responses returned by the item-scoring-service are serialized versions of tds.itemscoringengine.ItemScoreResponse class instances.

1. Configuring Item Scoring Engine

Item Scoring Service has no dependency on Program Management and this will stay as such in the foreseeable future. Currently, there are three properties that may be configured:

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| itemscoring.qti.sympyServiceUrl | String | Indicates the web address of the Python equation scoring service. Example: http://localhost:8080/ |
| itemscoring.qti.sympyMaxTries | Integer | Max number of attempts to connect to Python equation scoring service. Default is 3. |
| itemscoring.qti.sympyTimeoutMillis | Integer | This single value sets upper limits for socket timeout, response timeout and connection timeout when connecting to Python equation scoring service. |
| QThreadCount | Integer | The number of threads for scoring. |

All of these are configurable in settings.xml under src/main/resources folder as shown in figure 2 below.

1. Rubric mapping

Additionally you may want to remap rubric paths for operational reasons. You may achieve this by updating PatternUrlRewritter.txt that may be found under src/main/resources in item-scoring-service project as shown above.

This is a tab separated file and works on regexes. First column is the pattern and the second column is the target transformation. Multiple patterns may be listed – one on each line. The higher up a <pattern, transformation> pair is in the file the higher is its priority. There are sample patterns in the file.

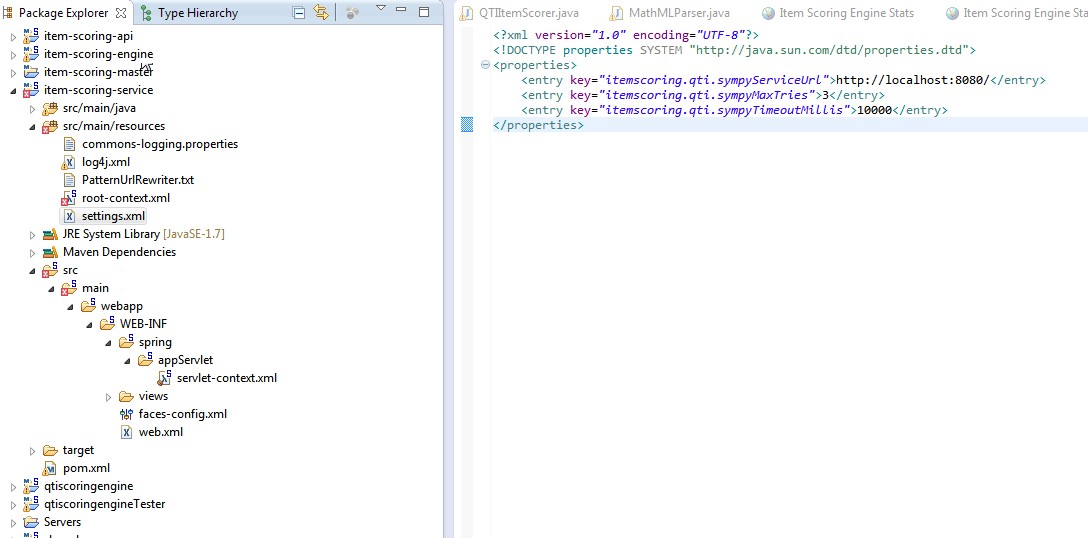


Figure 2: Item scoring settings

1. Configuring Student Application for Scoring

Configuring the student application to score items is beyond the scope of this document but some pointers will be provided here.

Student application may either do the scoring itself – if it has access to all the rubrics etc. – or it may send scoring requests to a separate dedicated item scoring service. For item formats that are resources intensive it may be better to offload to a separate item scoring service e.g. in the case of ER item types or NL item types.

For scoring to be undertaken by the student application itself, its scorer mapping will have to be updated in the root-context.xml file similar to what has been show in section 1 of this document.

For scoring to be offloaded to a separate item scoring service, additional database tweaks are required in the configs database to provide URL mappings for different item formats.