Specification for Evaluating Items

This specification is structured as followed:

Since the conditions for the four levels of evaluating the answers are similar, the whole description merges them to one routine and just differentiates between the certain sub-routines at the specific stages (they will be marked red).
I also integrated a description of the file system paths, which will be necessary for finding the correct input files

Preconditions

Parameters (required):

- user Number
- subject/dimension number
- level number
- evaluation-mode (1: What do I know?, 2: What can be improved?, 3: What I will be able of, if I will do some more?, 4: Teacher's Report)
- threshold for fulfilling a competency*
- max. number of listings (to limit the output to a fixed set of reported competencies or needs for improvement)

(* some competencies will be tested by more than one task/item, so you have to temp. store ALL competencies of a testcollection and then evaluate if they exceed the threshold. If for example the threshold is 100% and a competence is tested 3 times, then you will only include it into the report of “what can I do” if all 3 are backed by the correct answers).

Main Processing

Get Input Files:

- get paths to the files:
  - (application) root: /otulea/
  - user folder: /otulea/data/user/*usernumber*/ (where *usernumber* is placeholder for the certain user number like X0AT2 etc.)
  - main user file: *usernumber.xml (*inside the *usernumber* folder)
  - alphalist: /otulea/data/item/alphalevel/alphalist.xml
- load input files:
  - main user file
  - test-result-files (paths are written in the main user file and are inside the *usernumber* folder, the certain dimensions and levels are specified by the parameters, if no dimension and level is specified, the whole lists of test will be evaluated)
  - alphalist file

Information Extraction:

- read global user file to get a list of taken tests and the related urls to the test-result-files
- look at the latest entry, which represents the latest test-collection, users went through
- read all linked test-result-files
- read the marking section of every test-result-file

Derivation of Results:

Case A1 (What do I know?):

- take those marking entries with points, exceeding the minimal required threshold, indicating, that this item was solved correctly and look at their alphalevel entries (**again:* it is important, to have the threshold exceeded to definitely include the alphalevel into the list of the report)
- look inside the alphalist for the nodes with matching alphalevel attribute and look at their userdescription attribute, optionally (see parameters) look also at the example attribute
- add the userdescription to the list of results (leave the marking values, since participants should not be able to see them)
- sort the list by the **highest** alpha-level and keep only the highest x entries (where x is given by the parameter max. number of listings)

Case A2 (What can be improved?):

- take those marking entries which did not exceed the threshold for being a right answer, indicating, that this item was not solved correctly and look at their alphalevel entries (important: include also those, which emergy multiple times but did not exceed the threshold)
- look inside the alphalist for the nodes with matching alphalevel attribute and look at their userdescription attribute, optionally (see parameters) look also at the example attribute
- add the userdescription to the list of results (leave the marking values, since participants should not be able to see them)
- sort the list by the **lowest** alpha-level and keep only the lowest x entries (where x is given by the parameter max. number of listings)

Case A3 (What I will be able of, if I will do some more?):

- take those marking entries which exceeded the threshold, indicating, that this competence was solved correctly and look at their alphalevel entries
- IF the alphalist contains an alphalevel which is exactly one step higher AND this alphalevel is not included inside the actual test-result-file then add its

userdescription value to the result list (leave the marking values, since participants should not be able to see them)
- filter the result list by the max. number of listings parameter

Case B (Teacher's Report):

- teacher's report should be a **complete** list of all test-result-files, sorted historically (first order) and by dimensions(subjects (second order)
- for this case, the exact marking values are also important, therefore it is important to list alpha-level and marking value for each test-file

Output Files

There are two output formats required: one declarative file, which tells the flex application what to display (preferably xml) and a printable file (pdf) which will also be linked inside the declarative one. Both should be stored in the user-related folder.

Layout Considerations for the PDF

- include the lea. icon and the usernumber at the top-left of each page
- list descriptions vertically, headlines are given by the dimension/subject and level parameters
- case 1-3: include for each dimension the dimension-button image and the background color (see examples from Ilka)
- case 4 allows to be more text-based since its the teacher's report

Structural Considerations for the XML

- include a link to the pdf (since its relative path, the name of the file is needed)
- include a timestamp if possible
- include characteristics of the result (dimension,level,evaluation-mode)
  - include alpha-level codes

A possible result file could look like the following:

```
<results>
<print file=”20120504_14_13_X0AT2.pdf” />
<timestamp order=”YMDhms” value=”20120504141332” />
<dimension value=”2” />
<level value=”2” />
<eval mode=”1”>
      <alphaid value=”2.3.01” />
      <alphaid value=”2.1.03” />
</eval>
<eval mode=”2”>
      <alphaid value=”2.1.09” />
</eval>
<eval mode=”2”>
      <alphaid value=”2.3.02” />
      <alphaid value=”2.1.04” />
</eval>
</results>
```