Main functional parts of Multilevel supporting system for special forms of learning and their usage

[1]DALIBOR SLOVÁK, [2]PETR LÁTAL
[1]Computers and Communications System Department, Faculty of Applied Informatics Tomas Bata University, Zlín, CZECH REPUBLIC
[2]Computers and Communications System Department, Faculty of Applied Informatics Tomas Bata University, Zlín, CZECH REPUBLIC

Abstract— First part of the article is focused on work with Vexflow AP as a main functional part of Multilevel supporting system for special forms of learning . There will be described some basics how to create music staves, how the rendering works, how the modifiers works and many other useless features. Vexflow API is based on java-script web technology with possibility to render all types of music staves. Second part of the article is focused on "How to work with MSFL guide"and we described creation of stave and quiz module usage. The main aim of our research approach was to make simply and easy system for music teaching and learning.

Keywords: Vexflow API, Vexflow rendering, Vexflow modifiers Staves

1. Introduction

Multilevel supporting system for special forms of learning (in brief "MSFL") is being developed in the first instance as solution for teaching music theory in modern way. MSFL has features such as front-end layout for teaching and backend system for administration. First part of this article is focused on detailed description how we use Vexflow API in music learning management system, how many features are implemented in core generator and what features are planned to implement. There will be shown some code examples with screenshots of working generated staves.

Second part of this article will show how to use MSFL, how to register and how to study with this portal. There will be shown some screenshots with short description how it all works. MSFL portal is not based on any learning management system such as Moodle, eTutor, Spirit and more. At its first instance system is designed and optimized especially for music purposes.

2. Vexflow API

2.1 Work With Vexflow API

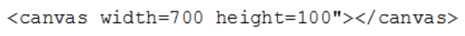
VexFlow is an open-source web-based music notation rendering JavaScript API that can generate any type of staves. This is an API that was designed for HTML 5 Canvas.

It is important to note that VexFlow is a low-level rendering API. Most applications will want to use something like VexTab which is a higher-level language for rendering guitar tablature and music notation.

VexFlow is written completely in JavaScript and when using it with HTML5 Canvas, requires no external libraries or dependencies. For SVG support, you will need to include the Raphael JavaScript library into your sources. That said, this tutorial also makes use of the jQuery library to select and manipulate DOM elements.

1) The basics for the creation of staves

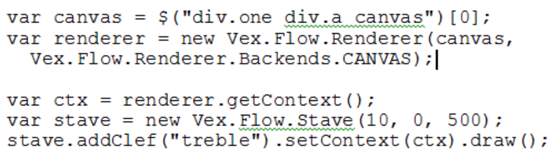
The absolute basis is tag canvas with specified width and height.



```
<canvas width=700 height=100"></canvas>
```

Fig. 1. Basic canvas tag

In order to create a blank stave, it need the following JavaScript code:



```
var canvas = $("div.one div.a canvas")[0];
var renderer = new Vex.Flow.Renderer(canvas,
  Vex.Flow.Renderer.Backends.CANVAS);

var ctx = renderer.getContext();
var stave = new Vex.Flow.Stave(10, 0, 500);
stave.addClef("treble").setContext(ctx).draw();
```

Fig. 2. Blank stave code

The first line calls the canvas object using jQuery function and store in a variable of the model canvas. You also need to generate your own background music using the function Vex.Flow.Renderer() with the parameters of the object variable canvas. The third important variable is the variable ctx which calls the getContext(), which creates environment for us stave. The fourth variable is the state that using Vex.Flow.Stave() creates a long staves with the positions of the first note from the left and top of your specified number of pixels. The shift is important for the higher notes, which may appear just behind the key. All are rendered using a addClef("trouble") will result in an empty stave with treble clef.

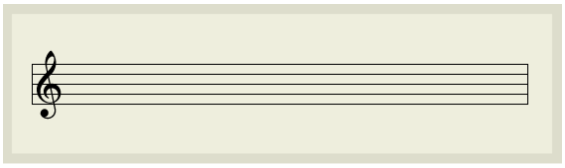


Fig. 3. Blank stave on front-end

In the above code first create a rendering context from the canvas element. This rendering context gives VexFlow a consistent 2D drawing interface, which is modeled on HTML5 Canvas. We then create a new Vex.Flow.Stave positioned at 0, 0 with a width of 500 pixels. We pass the context to the stave and call draw, which renders the stave on the context. Notice that the stave is not exactly drawn in position 0, 0. This is because it reserves some head-room for higher notes.

2) *Inserting notes*

Adding notes to the stave is a slightly more involved process. To understand the code, you need to understand the data model of the renderer. A StaveNote is a set of notes that belong on a stem. It can be a single note, or a chord. Its stem can be up, or down. All StaveNote instances have an associated duration. These StaveNotes are grouped into a Voice. Voices have a time signature, and the set of notes in the voice (including the rests) must utilize all beats in the voice. So, a 4/4 voice with only three quarter-notes is invalid - you'll need to add another quarter note or rest.
Voices are grouped into VoiceGroups. This is particularly useful when you have multi-voice music. Upon rendering, the notes in each voice of the group aligned on the stave. A VoiceGroup must contain at least one voice.
Finally, you have a Formatter, which takes a voice group and justifies the voices based on configurable rules, so that all the voices in the group look pretty on the stave. In the code below, we create a voice with two notes and a chord, and render it on the stave.

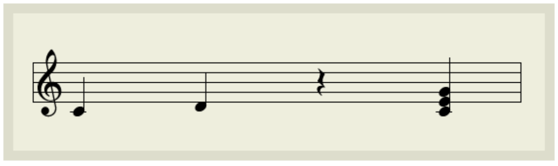
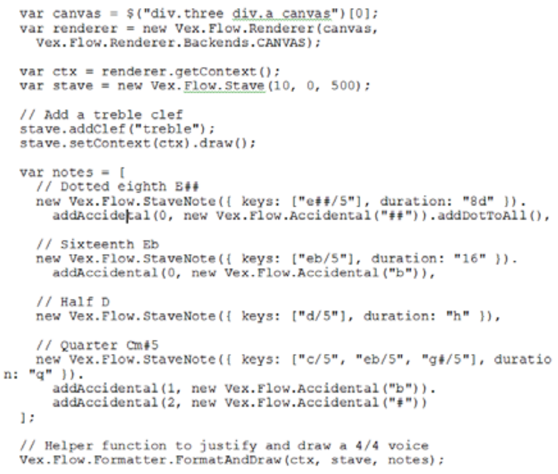


Fig. 4. Inserting note example

3) *Modifiers*

Modifiers are essentially decorators that are attached to notes. They include Accidentals, Vibratos, Dots, Annotations etc. Modifiers for a single group of notes live inside a ModifierContext. This allows them to intelligently juxtapose themselves based on other modifiers in the context. For example, a chord consisting of two close-by notes, each having accidentals, needs to position the accidentals such that they don't clash.
Code below shows what it looks like this situation:



```
  var canvas = $("div.three div.a canvas")[0];
  var renderer = new Vex.Flow.Renderer(canvas,
    Vex.Flow.Renderer.Backends.CANVAS);

  var ctx = renderer.getContext();
  var stave = new Vex.Flow.Stave(10, 0, 500);

  // Add a treble clef
  stave.addClef("treble");
  stave.setContext(ctx).draw();

  var notes = [
    // Dotted eighth E##
    new Vex.Flow.StaveNote({ keys: ["e##/5"], duration: "8d" }).
      addAccidental(0, new Vex.Flow.Accidental("##")).addDotToAll(),

    // Sixteenth Eb
    new Vex.Flow.StaveNote({ keys: ["eb/5"], duration: "16" }).
      addAccidental(0, new Vex.Flow.Accidental("b")),

    // Half D
    new Vex.Flow.StaveNote({ keys: ["d/5"], duration: "h" }),

    // Quarter Cm#5
    new Vex.Flow.StaveNote({ keys: ["c/5", "eb/5", "g#/5"], duratio
n: "q" }).
      addAccidental(1, new Vex.Flow.Accidental("b")).
      addAccidental(2, new Vex.Flow.Accidental("#"))
  ];

  // Helper function to justify and draw a 4/4 voice
  Vex.Flow.Formatter.FormatAndDraw(ctx, stave, notes);
```

Fig. 5. Example modifier code

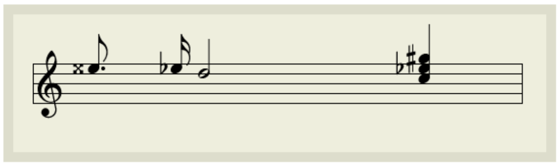
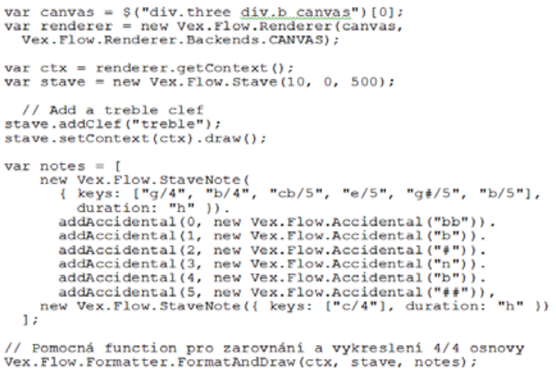


Fig. 6. Example modifier stave

In the above example, note that even though we set the note names and durations correctly, we explicitly request the rendering of accidentals and dots.
This is by design, and exists for two reasons:
We don't want to couple rendering logic with notational logic. The API user has the final say for what should and should not be displayed. This enables higher level tools and libraries (such as VexTab) to make rendering decisions based on their own notational semantics. Also notice that is used the FormatAndDraw helper function to create a 4/4 voice out of the notes, justify it to the stave, and render all of it. Lets add a few more modifiers and see how they position themselves.



```
var canvas = $("div.three div.b canvas")[0];
var renderer = new Vex.Flow.Renderer(canvas,
  Vex.Flow.Renderer.Backends.CANVAS);

var ctx = renderer.getContext();
var stave = new Vex.Flow.Stave(10, 0, 500);

  // Add a treble clef
stave.addClef("treble");
stave.setContext(ctx).draw();

var notes = [
    new Vex.Flow.StaveNote(
      { keys: ["g/4", "b/4", "cb/5", "e/5", "g#/5", "b/5"],
        duration: "h" }).
       addAccidental(0, new Vex.Flow.Accidental("bb")).
       addAccidental(1, new Vex.Flow.Accidental("b")).
       addAccidental(2, new Vex.Flow.Accidental("#")).
       addAccidental(3, new Vex.Flow.Accidental("n")).
       addAccidental(4, new Vex.Flow.Accidental("b")).
       addAccidental(5, new Vex.Flow.Accidental("##")),
    new Vex.Flow.StaveNote({ keys: ["c/4"], duration: "h" })
  ];

// Pomocná function pro zarovnání a vykreslení 4/4 osnovy
Vex.Flow.Formatter.FormatAndDraw(ctx, stave, notes);
```

Fig. 7. Another modifier code

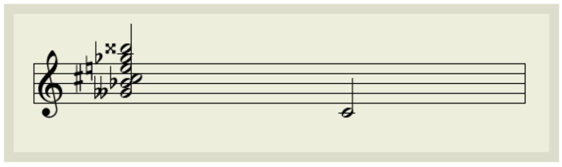
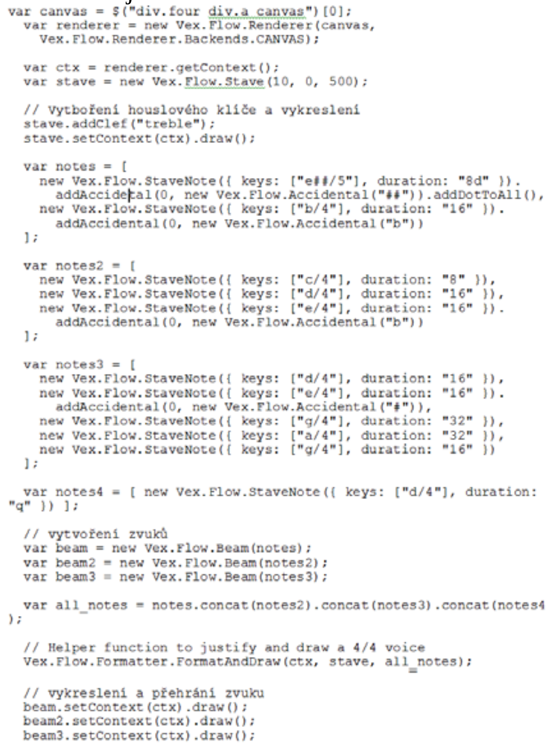


Fig. 8. Another modifier example

Above, the accidentals are positioned such that they don't overlap but maintain their X-relation to the associated note.

2.2 Sounds and Notations

VexFlow can play music, but only provided that the class reads beam.js. Creation of such a code is shown below.



```
var canvas = $("div.four div.a canvas")[0];
  var renderer = new Vex.Flow.Renderer(canvas,
    Vex.Flow.Renderer.Backends.CANVAS);

  var ctx = renderer.getContext();
  var stave = new Vex.Flow.Stave(10, 0, 500);

  // Vytboření houslového klíče a vykreslení
  stave.addClef("treble");
  stave.setContext(ctx).draw();

  var notes = [
    new Vex.Flow.StaveNote({ keys: ["e##/5"], duration: "8d" }).
      addAccidental(0, new Vex.Flow.Accidental("##")).addDotToAll(),
    new Vex.Flow.StaveNote({ keys: ["b/4"], duration: "16" }).
      addAccidental(0, new Vex.Flow.Accidental("b"))
  ];

  var notes2 = [
    new Vex.Flow.StaveNote({ keys: ["c/4"], duration: "8" }),
    new Vex.Flow.StaveNote({ keys: ["d/4"], duration: "16" }),
    new Vex.Flow.StaveNote({ keys: ["e/4"], duration: "16" }).
      addAccidental(0, new Vex.Flow.Accidental("b"))
  ];

  var notes3 = [
    new Vex.Flow.StaveNote({ keys: ["d/4"], duration: "16" }),
    new Vex.Flow.StaveNote({ keys: ["e/4"], duration: "16" }).
      addAccidental(0, new Vex.Flow.Accidental("#")),
    new Vex.Flow.StaveNote({ keys: ["g/4"], duration: "32" }),
    new Vex.Flow.StaveNote({ keys: ["a/4"], duration: "32" }),
    new Vex.Flow.StaveNote({ keys: ["g/4"], duration: "16" })
  ];

  var notes4 = [ new Vex.Flow.StaveNote({ keys: ["d/4"], duration:
"q" }) ];

  // vytvoření zvuků
  var beam = new Vex.Flow.Beam(notes);
  var beam2 = new Vex.Flow.Beam(notes2);
  var beam3 = new Vex.Flow.Beam(notes3);

  var all_notes = notes.concat(notes2).concat(notes3).concat(notes4
);

  // Helper function to justify and draw a 4/4 voice
  Vex.Flow.Formatter.FormatAndDraw(ctx, stave, all_notes);

  // vykreslení a přehrání zvuku
  beam.setContext(ctx).draw();
  beam2.setContext(ctx).draw();
  beam3.setContext(ctx).draw();
```

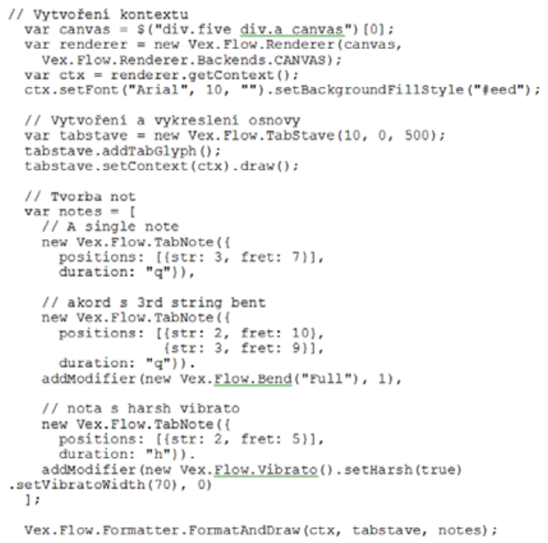
Fig. 9. Sound example code

In the above example, we created three groups of notes with beams. The slope of the beams is a function of the direction of the music, and the number of beams for each group is dependent on the duration of the notes underneath.

Tieing notes involves a similar set of operations. To render a tie, you create a StaveTie instance, and pass it the two StaveNotes to tie together. Since each StaveNote can be a chord, it is also need to pass in the indices of the specific notes we want to tie.

2.3 Guitar Outline

VexFlow guitar can also draw an outline. Mechanisms for the outline view are very similar to classical stave. However, it is necessary to use other classes to generate curriculum and notes.



```
// Vytvoření kontextu
  var canvas = $("div.five div.a canvas")[0];
  var renderer = new Vex.Flow.Renderer(canvas,
    Vex.Flow.Renderer.Backends.CANVAS);
  var ctx = renderer.getContext();
  ctx.setFont("Arial", 10, "").setBackgroundFillStyle("#eed");

  // Vytvoření a vykreslení osnovy
  var tabstave = new Vex.Flow.TabStave(10, 0, 500);
  tabstave.addTabGlyph();
  tabstave.setContext(ctx).draw();

  // Tvorba not
  var notes = [
    // A single note
    new Vex.Flow.TabNote({
      positions: [{str: 3, fret: 7}],
      duration: "q"}),

    // akord s 3rd string bent
    new Vex.Flow.TabNote({
      positions: [{str: 2, fret: 10},
                  {str: 3, fret: 9}],
      duration: "q"}).
    addModifier(new Vex.Flow.Bend("Full"), 1),

    // nota s harsh vibrato
    new Vex.Flow.TabNote({
      positions: [{str: 2, fret: 5}],
      duration: "h"}).
    addModifier(new Vex.Flow.Vibrato().setHarsh(true)
.setVibratoWidth(70), 0)
  ];

  Vex.Flow.Formatter.FormatAndDraw(ctx, tabstave, notes);
```

Fig. 10. Guitar outline code

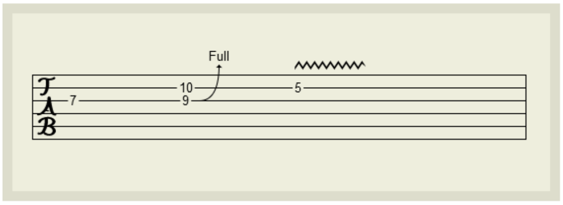


Fig. 11. Guitar outline

Above, we replaced Stave with TabStave and StaveNote with TabNote. There are also added some bend and vibrato modifiers.

There are two things we have to manually specify here the font style, and the background fill color. The former is used to display fret numbers, annotations, and other text. The latter is only required for the SVG backend (although using it with Canvas is harmless), and is used to create an internal implementation of clearRect. A custom clearRect is required to clear sections of the canvas. This is not supported by SVG because it has no "clear" semantics.

3. Stave Module and Quiz Module

3.1 Sheet of Music Generator

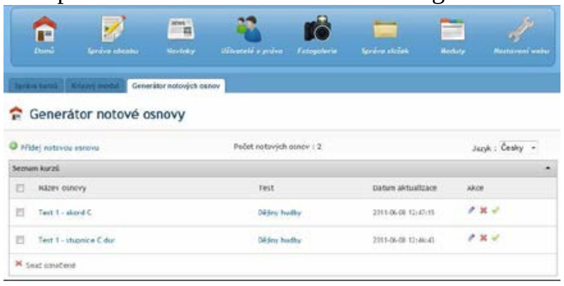
Module for generating staves is served as the simplified interface for creating musical staves for the quiz module and for the presentation of given expressions on the web. The main features of the module include the name of the stave, connection with the test and stave status.

From the database perspective the module contains a data section, which is generated by module and stored in a database as a complete code ready for presentation.

1) Creation of musical staves

After click on “Pridej notovou osnovu” (Add sheet of music) you can see common information about sheet of music. You

must create unique information about your sheet of music. This item is required and you must fill it. Now it has created tag<canvas> with displayed javascript. If you create two note scores with same identifier. You don’t see anything note score. Then it is necessary to name your score and set it as a active.
Elementary setting of your score is on the right side of form. It is useful to edit these parameters. For example width of score with pixels value determines width of tag <canvas>.



Generátor notové osnovy

Přidej notovou osnovu — Počet notových osnov : 2 — Jazyk : Česky

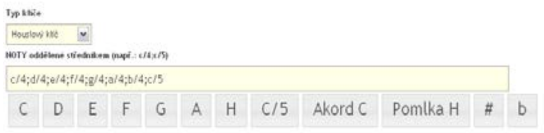
Seznam kurzů

Název osnovy	Test	Datum aktualizace	Akce
Test 1 - akord C	Dějiny hudby	2011-06-08 12:47:15	
Test 1 - stupnice C dur	Dějiny hudby	2011-06-08 12:46:41	

Smazat označené

Fig. 12. Sheet of music generator

Other is number of lines. You would be determines the number of lines. Position of x and y determines only view of first character against note score. You need not set this property. Last argument is score longitude in pixel’s value. This value determines exact place for view of notes.


Typ klíče: Houslový klíč

NOTY oddělené středníkem (např.: c/4;c/5)

c/4;d/4;e/4;f/4;g/4;a/4;b/4;c/5

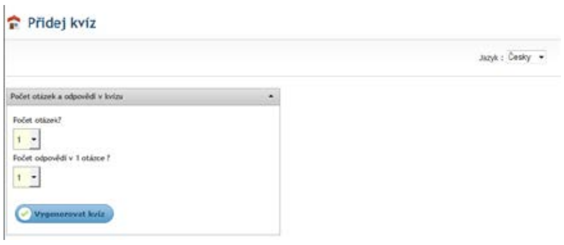
Fig. 13. Stave generator module preview

3.2 Quiz Module

This is the module that generates any types of quiz or test. The basic properties of this module are the name of the quiz results, categories, and the basic operations such as editing, adding and deleting quizzes.

1) *New quiz creation*

Creation of the quiz is divided into two steps. The first step is the selection of the number of questions and the number of responses. Here is a good to decide the corresponding number of questions and answers, because if you change your mind the number of questions or the number of answers you have to regenerate the form. In this situation is the best solution to save the quiz and then edit it because editing will not lost the actual data.


Přidej kvíz

Jazyk : Česky

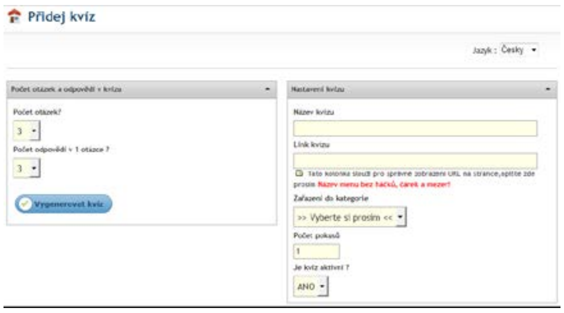
Počet otázek a odpovědí v kvízu

Počet otázek? 1

Počet odpovědí v 1 otázce ? 1

Fig. 14. Quiz creation – first step

In the second step is "Quiz generation." The module will now expand to the right side. Here is necessary to fill in the quiz name, category, set number of attempts and select status of the quiz. When you generate the quiz, the interface will appear box for adding questions with answers.


Přidej kvíz

Jazyk : Česky

Počet otázek a odpovědí v kvízu

Počet otázek? 3

Počet odpovědí v 1 otázce ? 3

Nastavení kvízu

Název kvízu

Link kvízu

Tato kolonka slouží pro správné zobrazení URL na stránce, vyplňte zde prosím Název menu bez háčků, čárek a mezer!

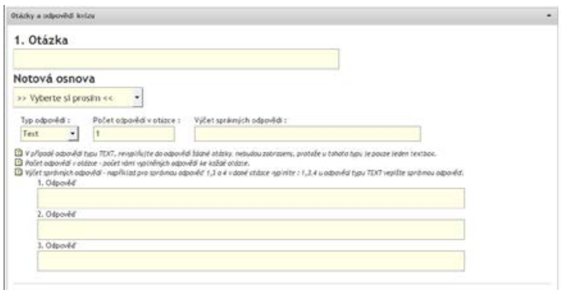
Zařazení do kategorie: >> Vyberte si prosím <<

Počet pokusů: 1

Je kvíz aktivní ? ANO

Fig. 15. Quiz creation – second step

You can simply fill in your questions with answers. Because the questions and answers are over dimensioned, it is necessary to modify a few items. The first item is the type of answers. This offers the text type, radio and the checkbox. Changing any type will display the form of answers. Selecting musical stave from the select box will display chosen stave. Please refer to the fact that each question has to have different stave (or stave with another identifier).


Otázky a odpovědi kvízu

1. Otázka

Notová osnova: >> Vyberte si prosím <<

Typ odpovědi : Text — Počet odpovědí v otázce : 1 — Výčet správných odpovědí :

V případě odpovědi typu TEXT, nevyplňujte do odpovědí žádné otázky, nebudou zobrazeny, protože u tohoto typu je pouze jeden textbox.
Počet odpovědí v otázce - počet všech vyplněných odpovědí ke každé otázce.
Výčet správných odpovědí - například pro správnou odpověď 1,3 a 4 v dané otázce vyplníte : 1,3,4 u odpovědí typu TEXT vepište správnou odpověď.

1. Odpověď

2. Odpověď

3. Odpověď

Fig. 16. Quiz creation – questions and answers

4. Problems and Solutions

MSFL version 1.0 allows storing only a few types of notes in staves. The main problem is only with various types of combinations which have to be implemented by the programmers.
Next problem is connected with communication of basic hardware piano pads and computers. We have plan to invent USB interface which will communicate through the java applet between browser and hardware and serve the maximal comfort for the MSFL users.

5. Conclusion

Vexflow API converts user inputs in music learning management system to the canvas HTML tag which display output on the page. We made the MSFL version 1.0 which allows to use only a few user inputs in the form of notes to the system.

Our plan is to improve the converting core to allow users to insert all types notes and connect a special convertor between computer and basic piano pad to increase efficiency of using MSFL.

The main mission of this project is to spread the music learning management system to the whole world. We have some analysis from the conservatory that teacher need some instrument how to easily teach and share their knowledge with the students. This article was the last part and now we are prepared to finish the customizations and release the first version.

References

[1] Henk C.A : van TILBORG (2000). *Fundamentals of cryptology*, Kluwer Academic, ISBN 0-7923-8675-2, Norwell.

[2] Schlossnagle, Georgie (2004). *PHP 5 advanced programming*, Computer Press, ISBN 80-86815-14-5, Brno.

[3] Composite authors (2007), *PHP 5 mastery*, Computer Press, ISBN 978-80-251-1519-0, Brno.

[4] Resig, John (2007). *Javascript a AJAX : modern programming of web applications*, Computer Press, ISBN 978-80-251-1824-5, Brno.

[5] Zelinka, Ivan. (1999). Applied informatics, Editorial center, UTB, . ISBN 80-214-1423-5, Zlin

[6] Castro, Elizabeth (2007). *HTML : XHTML a CSS: web site creating*. Computer Press, ISBN 978-80-251-1531-2, Brno.

[7] RESIG, John. (2007) Javascript a AJAX : *Moderní programování webových aplikací*. Computer Press, ISBN 978-80-251-1824-5. Brno.

[8] Latal, P. *Web portal for teaching music theory: Tomas Bata University*, Faculty of Applied Informatics, Department of Computer and Communication Systems, 2011, 87 p. Thesis supervisor Ing. Dalibor Slovak.

[9] Ullman, Larry (2004). *HTML : PHP a MySQL*, Computer Press, ISBN 80-251-0063-4, Brno.

[10] Composite authors (2007), *PHP 5*, Computer Press, ISBN 978-80-251-1519-0, Brno.

[11] Wexflow [available online]:
<http://vexflow.com/docs/tutorial.html>

[12] Cufonlibrary[available online]:
<https://github.com/sorccu/cufon/wiki/About>

[13] *Raphael library* [available online]: *Cufon* [available online]:
<http://raphaeljs.com/>