FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Nástroj pro konverzi a nasazení e-learningových materiálů
Student:	Bc. Petr Pejša
Vedoucí:	Ing. Miroslav Balík, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

Cílem této diplomové práce je zajistit přesun všech výukových materiálů ze serveru e-learning.fit.cvut.cz do Moodlu nasazeného na FIT ČVUT v Praze.
Server e-learning.fit.cvut.cz je klon serveru EDUX upravený pro podporu e-learningu na FIT ČVUT v Praze.
Pro cílovou platformu Moodle vytvořte automatizovaný přenos předmětů BI-PA1, BI-PA2 a BI-UOS.
Analyzujte možné formy transformace kurzu tak, aby výsledný kurz na platformě Moodle byl co možná nejvíce totožný se stávajícím (výukové materiály, testy apod.) a zároveň se zachovaly přechody mezi jednotlivými výukovými materiály a testy. Pokuste se zajistit převod tak, aby zajistil dostatečné editační možnosti bez nutnosti použití prostého HTML.
Na základě analýzy transformujte data ze stávajícího kurzu do stavu, který bude vhodný pro import do Moodlu.
Naimportujte data a multimediální obsah (obrázky, animace, atd.) do Moodlu.
Transformujte přechody mezi jednotlivými materiály.
Zkontrolujte výsledný stav.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 2. února 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Nástroj pro konverzi a nasazení e-learningových materiálů

Bc. Petr Pejša

Katedra softwarového inženýrství
Vedoucí práce: Ing. Miroslav Balík, Ph.D.

8. května 2018

Poděkování

Chtěl bych zde poděkovat panu Ing. Miroslavu Balíkovi, Ph.D., za jeho dobré rady, odborné vedení a vstřícný přístup. Dále také za pomoc při gramatické kontrole práce. Rovněž bych chtěl poděkovat panu Ing. Ladislavu Vagnerovi, Ph.D., za vstřícnost a pomoc při získání potřebných informací a podkladů. Mé poděkování patří též mé rodině a blízkým přátelům za pomoc a podporu během studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 8. května 2018 .....

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Pejša, Petr. *Nástroj pro konverzi a nasazení e-learningových materiálů.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Tato práce se zabývá přesunem dat studijních materiálů ze serveru e-learning.cvut.cz do Moodlu. Hlavním cílem bylo data z e-learningu transformovat do struktury odpovídající struktuře Moodle a jednotlivé studijní materiály přetransformovat do syntaxe, kterou podporuje Moodle. Velký důraz byl kladen na transformaci dat a to tak, že bylo požadováno, aby byly zajištěny editační možnosti. V příloze lze nalézt dokumentaci a zdrojové kódy.

Klíčová slova Transformace dat, parsování textu, e-learningové kurzy, studijní materiály, Moodle, DokuWiki, Python, PHP, DOT

Abstract

This work is focused on transfering study materials data from e-learning.fit.cvut server to Moodle. The main objective of this thesis was to transform the e-learning data to the structure which is compatible with Moodle and each study materials transform to syntax which is supported by Moodle. Great emphasis was put on the data transformation to secure editable possibilities. Documentation and source codes are listed in annexe.

Keywords Data transformation, parse text, e-learning course, study materials, Moodle, DokuWiki, Python, PHP, DOT

Obsah

Seznam obrázků

Seznam tabulek

Úvod

E-learningový portál e-learning.fit.cvut.cz obsahuje výukové kurzy, které slouží k výuce na FIT ČVUT v Praze. Na portálu e-learning.fit.cvut.cz jsou výukové materiály ke kurzům Programování a algoritmizace 1, 2 a Programování v shellu. E-learningové kurzy se skládají z výukových materiálů strukturovaných do kapitol. Kapitola obsahuje výukové lekce a testy. Ve výukových lekcích je vysvětlena a popisována probíraná látka. Testy slouží k ověření nově nabytých znalostí z předchozích lekcí.

Všechny výukové materiály jsou uspořádány tak, aby šlo kurzem projít co nejjednodušeji a nejsrozumitelněji. Výukové materiály jsou uspořádány tak, aby na sebe navazovaly. Toto pořadí je definováno grafem struktury. Graf struktury kurzu je ve formátu DOT. Uzly grafu představují studijní materiály a hrany definují následující výukové materiály. Graf nám tedy definuje pořadí jednotlivých studijních materiálů.

Server e-learning.fit.cvut.cz je klon serveru EDUX, na kterém jsou dostupné studijní materiály v rámci vyučovaných předmětů na FIT ČVUT v Praze. Server e-learning.fit.cvut.cz je upravený pro podporu e-learningu. Na serveru je použitý systém DokuWiki pro editaci výukových materiálů. DokuWiki je forma wiki zaměřená na vytváření dokumentace. Systém není úplně vyhovující pro potřeby e-learningového portálu. Systém nepoužívá databázi a všechna data jsou uložena v textových souborech na disku.

Každá výuková lekce je samostatná stránka, strukturovaná pomocí víceúrovňových nadpisů a dalších textových a multimediálních prvků. Testy systém DokuWiki nepodporuje, je do něj doimplementována podpora, která odesílá požadavky na zadání a vyhodnocení testů na progtest.fit.cvut.cz

Moodle je open-source e-learningová platforma, která je vyvíjena a podporována komunitou. Moodle podporuje širokou škálu studijních, výukových a dalších podpůrných materiálů, případně lze další přidat pomocí zásuvných modulů (pluginů). Moodle podporuje tvorbu testů, které mohou podmiňovat přístup k dalším materiálům. Moodle podporuje velký počet různých druhů

otázek v testech.

Cílem této práce je zajistit přesun všech výukových materiálů pro kurzy Programování a algoritmizace 1 (BI-PA1), Programování a algoritmizace 2 (BI-PA2) a Programování v shellu 1 (BI-PS1) ze serveru e-learning.fit.cvut.cz do Moodlu, nasazeného na FIT ČVUT v Praze. Transformace kurzu by měla proběhnout tak, aby výsledný kurz byl co možná nejvíce podobný se stávajícím. Důležité je zachování pořadí výukových lekcí a testů v kapitolách. Zároveň je potřeba zajistit podobné editační možnosti.

Kapitola 1

Analýza e-learningových kurzů

Než začneme kopírovat data do Moodlu, musíme nejprve zjistit, která data se budou transformovat, jak je budeme transformovat, v jakém jsou data formátu, do jakého formátu je potřebujeme dostat, a mnoho dalšího.

Proto nejprve začneme analýzou stávajícího systému na serveru e-learning.fit.cvut.cz. Server e-learning.fit.cvut.cz je klon serveru EDUX upravený pro podporu e-learningu. Systém, na kterém beží e-learningový portál, se jmenuje DokuWiki. „DokuWiki je standardy dodržující, jednoduše použitelná forma wiki, zaměřená především na vytváření dokumentace všeho druhu“ - jak autoři sami píší na webu [1]. „Je cílena na vývojové týmy, pracovní skupiny a malé firmy.“ - uvádí dále.

Systém není příliš vhodný pro e-learningové kurzy. DokuWiki cílí spíše na vytváření a správu dokumentace. Tím bychom mohli vytvořit pouze výukové lekce, ale není možné vytvořit testy pro studenty ani nic podobného. Oproti tomu Moodle [2] je robustní open-source vzdělávací platforma. Moodle podporuje širokou škálu studijních a podpůrných materiálů a mnoho dalších lze přidat pomocí pluginů. Moodle je celosvětově využívaný vzdělávací systém.

Moodle je tedy přesně takový software, který potřebujeme pro e-learningový kurz. Nové e-learningové kurzy lze v Moodlu vytvořit pomocí interního textového editoru, který poskytuje mnoho formátovacích prvků. Na starém systému již jsou e-learningové kurzy zpracované, proto chceme tyto kurzy dostat do Moodle.

V této kapitole si rozebereme e-learningové kurzy na jednotlivé prvky výukových materiálů, které se v kurzu vyskytují a které budeme chtít následně transformovat a přenést.

Současné e-learningové kurzy na e-learning.fit.cvut.cz obsahují pouze dvě kategorie výukových materiálů. První kategorií jsou výukové lekce a druhou jsou testy.

1.1 Výukové lekce na DokuWiki

Výukové lekce jsou uloženy v textovém souboru a jsou popsány jednoduchým formátovacím jazykem. Výuková lekce má podobu jedné stránky, která je formátovaná pomocí víceúrovňových nadpisů a dalších textových a multimediálních prvků. Tyto prvky si teď jednotlivě rozebereme.

Budou zde uvedeny pouze prvky syntaxe DokuWiki, které jsou použity na e-learningovém portálu e-learning.fit.cvut.cz, ostatní můžeme ignorovat.

1.1.1 Základní formátování textu

Mezi základní formátování textu patří:

1.1.1.1 Tučné písmo

Tučné písmo je základním prvkem formátování. V DokuWiki syntaxi je tučné písmo uvozeno znaky ** a zakončeno také **. To znamená, že cokoli mezi znaky ** a ** bude tučně.

```
**Takto se zapíše tučné písmo**
```

1.1.1.2 Kurzíva

Kurzíva je písmo vyznačující se mírným sklonem doprava a u některých písmen drobnými změnami, které písmo přibližují psacímu.

Kurzíva se v DokuWiki syntaxi zapíše mezi dvě // a je tedy uvozena dvojicí znaků //a zakončena dvojicí znaků //.

```
//Takto se zapíše kurzíva//
```

1.1.1.3 Podtržené písmo

Podtržené písmo se zapíše opět podobně. Podtržené písmo je uvozeno znaky __ a zakončeno znaky __.

```
__Takto se zapíše podtržené písmo__
```

1.1.1.4 Neproporciální písmo

Neproporciální písmo je druh písma, kde mají všechny znaky stejnou šířku. Například písmena *I* a *E* nevyžadují stejné množství prostoru, takže v písmech proporcionálních bude první z nich užší než to druhé. V písmu neproporcionálním by však každé z nich zabíralo stejný prostor.

V DokuWiki je neproporciální písmo uvozeno znaky '' a zakončeno znaky ''.

```
''Takto se zapíše neproporciální písmo''
```

1.1.1.5 Horní a dolní index

Horní a dolní index se hojně využívá nejenom v matematice.

V DokuWiki se horní a dolní index zapíší stejně jako v jazyce HTML, mezi párové tagy <sup>, resp. <sub>. Horní index se tedy zapíše mezi tagy ^a a dolní index se zapíše mezi tagy _a:

```
<sup>Takto se zapíše horní index</sup>
<sub>Takto se zapíše dolní index</sub>
```

1.1.1.6 Přeškrtnutý text

Pro přeškrtnutý text využívá DokuWiki opět syntaxe HTML. Přeškrtnutý text se zapíše mezi tagy <del> a </del>.

```
<del>Takto se zapíše přeškrtnutý text</del>
```

1.1.1.7 Odstavce a zalomení řádku

Odstavce se oddělují pomocí dvou prázdných řádků.

Zalomení řádku bez nového odstavce lze udělat pomocí \\ následované bílými znaky nebo koncem řádku.

1.1.2 Odkazy

DokuWiki podporuje dva způsoby, jak vytvářet odkazy - interní a externí.

1.1.2.1 Externí odkazy

Externí odkazy jsou rozpoznávány automaticky, když je odkaz napsaný v celé formě včetně protokolu http://. DokuWiki dále podporuje možnost si nastavit vlastní text k odkazu místo URL.

1.1.2.2 Interní odkazy

Interní odkazy se vytváří pomocí dvojitých hranatých závorek. Interní odkaz je uvozen znaky [[a je ukončen znaky]]. Znak | mezi závorkami odděluje URL od textu. Pokud odkaz neodsahuje text, použije se jako text URL odkazu. Syntaxe vypadá následovně:

```
[[www.cvut.cz]]
[[Text odkazu|www.cvut.cz]]
```

1.1.3 Nadpisy

Pro rozumné strukturování textu je poskytnuto pět úrovní nadpisů. V DokuWiki syntaxi se je nadpis uvozen znaky == pro nadpis nejnižší úrovně a ====== pro nadpis první úrovně a zakončené jsou == resp. ======.

```
======= Nadpis 1. úrovně =======
```

1.1.4 Oddělovací vodorovná čára

Pomocí čtyř, nebo více pomlček vytvoříte oddělovací vodorovnou čáru.

1.1.5 Obrázky a jiné soubory

DokuWiki poskytuje možnost nahrát do dokumentu obrázky a jiné soubory. Soubory, podobně jako odkazy, se vkládají pomocí složených závorek {{URL}}.

Při vkládání obrázku je možnost dalšího nastavení. Velikost obrázku můžeme definovat zápisem velikosti za URL souboru. URL a rozměry obrázku jsou oddělené otazníkem.

```
{{URL?500}}
```

```
{{URL?450x150}}
```

Titulek k obrázku připojíme za velikost souboru, pokud je nastavena. Titulek je oddělen svislým znakem.

```
{{URL|Titulek}}
```

```
{{URL?450x150|Titulek}}
```

Zarovnání obrázku je nastaveno pomocí mezer mezi úvodními a koncovými závorkami. Následovně:

```
{{  URL?450x150}} - zarovnání vpravo
```

```
{{URL?450x150  }} - zarovnání vpravo
```

```
{{  URL?450x150  }} - vycentrování
```

1.1.6 Seznamy

DokuWiki podporuje číslované i nečíslované seznamy. Položku seznamu vytvoříme odsazením textu o dvě mezery a použitím hvězdičky * pro nečíslované seznamy, resp. pomlčky - pro seznamy číslované.

Další úroveň seznamu vytvoříme odsazením o další dvě mezery, jak je vidět na příkladu níže.

- Číslovaný seznam
  - 2. úroveň seznamu
    - 3. úroveň seznamu
- Číslovaný seznam

1.1.7 Tabulky

DokuWiki podporuje jednoduchou syntaxi pro vytváření a formátování tabulek. Normální řádky tabulky musí začínat a končit svislou čárou |, řádky v hlavičce stříškou ^. Jednotlivé buňky v řádku tabulky jsou opět uvozeny svislou čárou |, resp. stříškou ^.

Příklad zápisu tabulky v DokuWiki:

```
^ Nadpis 1     ^ Nadpis 2             ^ Nadpis 3           ^
^ Nadpis 4     | řádek 1 sloupec 2    | řádek 1 sloupec 3  |
^ Nadpis 5     | nespojené sloupce    |                    |
^ Nadpis 6     | řádek 2 sloupec 2    | řádek 2 sloupec 3  |
```

Spojení dvou buněk dohromady je docíleno ponecháním prázdného místa mezi | a | v buňce. Vedlejší buňka se roztáhne přes prostor této buňky (spojí se dohromady).

Obsah buněk se může také zarovnat. Přidáním alespoň dvou bílých znaků na druhý konec textu: Přidejte dvě mezery nalevo pro zarovnání doprava, dvě mezery napravo pro zarovnání doleva nebo aspoň dvě mezery na oba konce pro zarovnání na střed.

Ukázka formátované tabulky v DokuWiki:

```
^           Tabulka se zarovnáním          ^^^
|       doprava|   na střed   |doleva        |
|doleva        |       doprava|   na střed   |
| xxxxxxxxxxxx | spojené sloupce            ||
```

1.1.8 Předformátovaný text

Pokud je potřeba zobrazit text přesně tak, jak je napsaný (bez formátování), uzavře se daná oblast mezi tagy <nowiki> a </nowiki>, nebo dokonce jednodušeji do znaků dvojitých procent %%.

```
<nowiki>Tento text se nebude upravovat</nowiki>
%% Tento text se nebude upravovat %%
```

1.1.9 Bloky kódu

Do stránek lze zahrnout i bloky, které nebudou interpretovány, když je odsadíme o alespoň dvě mezery od začátku řádku, nebo použitím tagů <code>.

Bloky kódu v DokuWiki jsou velmi hojně využívané na e-learningovém portálu e-learning.fit.cvut.cz. Tvůrci kurzů bloky kódu využívali k různým účelům, díky svému specifickému vzhledu (viz obr. 1.1) je jejich použití rozsáhlé.

Na e-learningovém portálu e-learning.fit.cvut.cz se formátování kódu používá převážně k ukázce výstupů různých programů.

```
code    code
```

```
code    code
```

```
code
code
code
code
```

Obrázek 1.1: DokuWiki code

1.1.9.1 Zvýrazňování syntaxe

DokuWiki umí zvýrazňovat a obarvovat zdrojové kódy (viz obr. 1.2), aby se daly lépe číst. Používá přitom GeSHi (Generic Syntax Highlighter). Díky tomu DokuWiki podporuje všechny jazyky, které umí zvýrazňovat GeSHi.

Syntaxe je stejná jako v bloku kódu v minulé sekci, jen se tentokrát dovnitř tagu vloží název jazyka. Např. <code java>.

```
1: #include <stdio.h>
2: int main()
3: {
4:     // printf() displays the string inside quotation
5:     printf("Hello, World!");
6:     return 0;
7: }
```

Obrázek 1.2: Ukázka obarvení kódu

1.1.10 Vnořování prvků

Systém DokuWiki sice nikde v dokumentaci nedefinuje pravidla pro vnořování prvků do sebe, ale dodržuje striktní pravidla, která jsme zjistili pomocí testování. Dovoluje jednonásobné vnoření skoro každého prvku. Vícenásobné vnořování stejného prvku není možné.

V následující tabulce 1.1 si ukážeme, jaké prvky lze vnořovat do sebe.

Tabulka 1.1: Vnořování prvků

Prvek DokuWiki	**Mohu do prvku vnořit**	**Může být vnořen**
Tučné písmo	Ano	Ano
Kurzíva	Ano	Ano
Podtržené písmo	Ano	Ano
Neproporciální písmo	Ano	Ano
Horní a dolní index	Ano	Ano
Přeškrtnutý text	Ano	Ano
Odkazy	Ne	Ano
Nadpisy	Ne	Ne
Obrázky a soubory	Ne	Ano
Seznamy	Ano	Ne
Tabulky	Ano	Ne
Předformátovaný text	Ne	Ano
Bloky kódu	Ne	Ano

1.2 Testy na DokuWiki

Testy na DokuWiki řeší API, které přes XML zprávy zasílá požadavky. E-learning zasílá požadavky a následně vyhodnocuje odpovědi od progtestu.

Popis funkce je následovný:

1. E-learning pošle požadavek na zaslání otázky. Součástí požadavku je identifikace otázky a studenta.

2. Progtest náhodně vygeneruje otázku podle zadaných parametrů.

3. Progtest odešle otázku uloženou v XML dokumentu zpět na e-learning.

4. E-learning zobrazí otázku a vyčká na vyplnění a odeslání.

5. E-learning pošle na progtest odpověď. Součástí odpovědi je identifikace otázky a studenta.

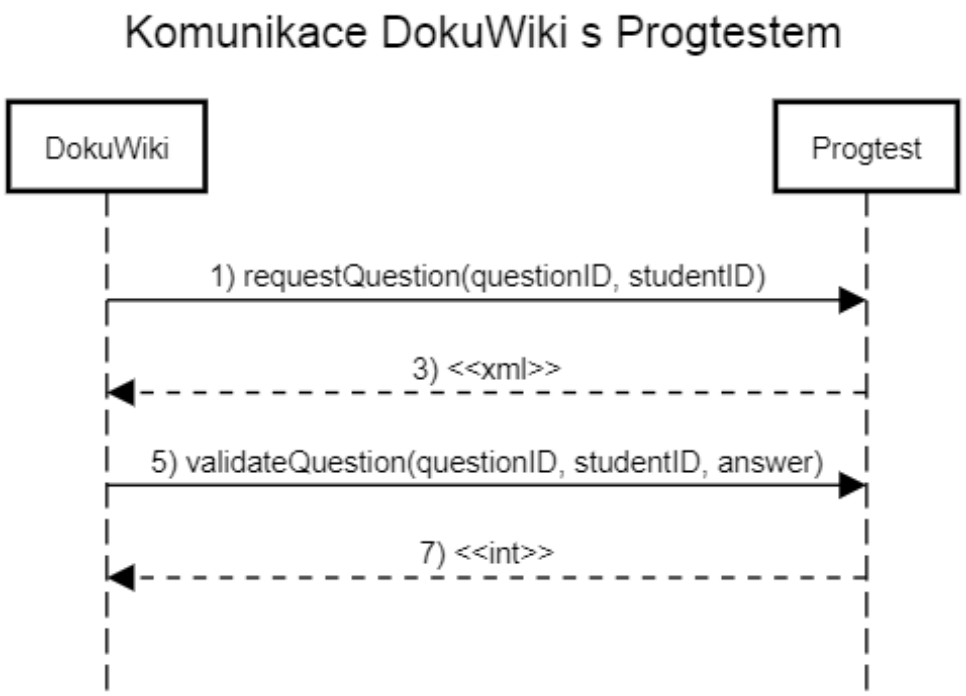
6. Progtest ohodnotí odpověd studenta podle vygenerovaných parametrů.

7. Progtest zpátky odešle bodování.

Zasílání zpráv je znázorněno na obrázku 1.3.

Z bezpečnostních důvodů je celá komunikace progtestu s e-learningovým portálem zajištěna na fixní adresu e-learningového portálu e-learning.fit.cvut.cz.

Na vypracování testu není omezen počet pokusů a výsledek testu není nijak závazný. Pro pokračování v kurzu není nutné splnit test.



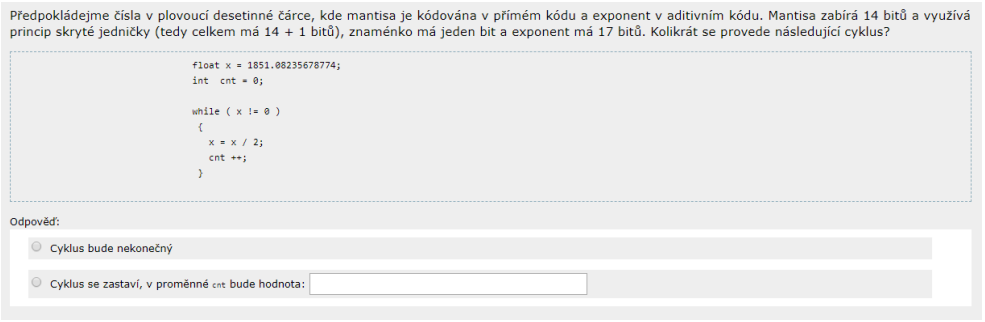
Obrázek 1.3: Ukázka komunikace s progtestemu

1.2.1 Typy otázek

E-learningový portál disponuje těmito typy otázek:

- Výběr z možností
  - Jedna správná odpověď
  - Více správných odpovědí
- Doplňovací otázka (text)
- Doplňovací otázka (číslo)
- Kombinace výběru a doplnění

Doplňovací otázky a otázky s výběrem jsou obecně známé a nepotřebují další komentář. Otázka typu *Kombinace výběru a doplnění* je zobrazena na obrázku 1.4. Otázka se vyplní nejdříve jako výběr ze dvou možností, přičemž jedna možnost je vždy, že neexistuje výsledná hodnota a druhá možnost je, že výsledná hodnota existuje. Když hodnota existuje, je potřeba ještě výslednou hodnotu doplnit.



Obrázek 1.4: Ukázka otázky kombinace výběru a doplnění

1.3 Pluginy na DokuWiki

Na e-learningovém portálu je použit pouze jeden plugin, který vytváří výrazné poznámky a upozornění v textu.

1.3.1 Plugin: Note

Tento plugin umožňuje vytvářet informační poznámky a upozornění v textu.

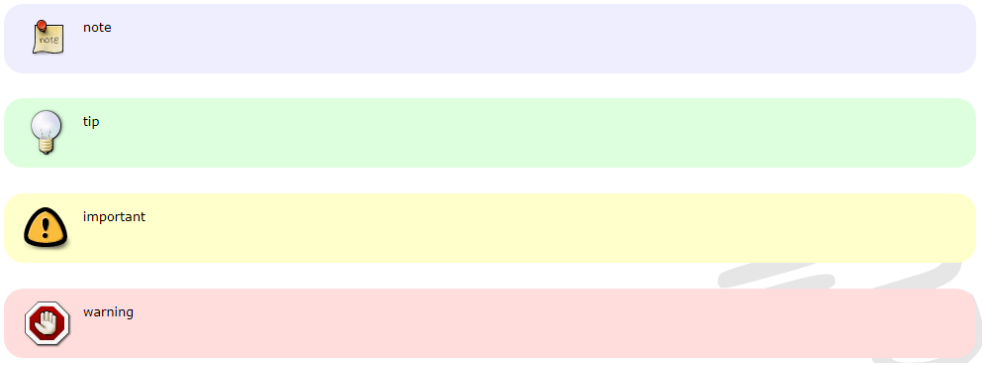
Plugin má 4 typy poznámek:

- informační,
- potvrzovací,
- upozorňovací,
- varovná.

Poznámky se v DokuWiki zapíší uvnitř tagů <note>. Pokud je tag <note> bez atributu, vytvoří se informační poznámka.

Atributy pro vytvoření poznámek:

- informační - <note>,
- potvrzovací - <note tip>,
- upozorňovací - <note important>,
- varovná - <note warning>.



Obrázek 1.5: DokuWiki poznámky

1.4 Analýza struktury

Nyní se podívíme na to, jakým způsobem jsou jednotlivé výukové materiály v kurzu provázány a jak na sebe navazují.

Popis struktury kurzu je uložen pro každý kurz v souborech typu DOT. DOT [3] je jazyk popisující grafy, podporuje jak orientované, tak neorientované grafy. K uzlům a hranám je možné přidat popisky.

V našem případě uzly představují výukový materiál (lekci, nebo test) a hrany přechody mezi nimi.

1.5 Syntaxe DOTu

Jazyk DOT je popsán jednoduchou gramatikou. Gramatika začíná neterminálem *graph*. Terminály jsou psány velkými písmeny a neterminály malými. Hranaté závorky značí nepovinné položky. Terminální znaky jsou mezi uvozovkami.

Gramatika pro jazyk DOT:

```
graph       :  [ STRICT ] (GRAPH | DIGRAPH) [ id ] '{' stmt_list '}'
stmt_list   :  [ stmt [ ';' ] stmt_list ]
stmt        :  node_stmt
            |  edge_stmt
            |  attr_stmt
            |  id '=' id
            |  subgraph
attr_stmt   :  (GRAPH | NODE | EDGE) attr_list
attr_list   :  '[' [ a_list ] ']' [ attr_list ]
a_list      :  id '=' id [ (';' | ',') ] [ a_list ]
edge_stmt   :  (node_id | subgraph) edgeRHS [ attr_list ]
```

```
edgeRHS     :  edgeop (node_id | subgraph) [ edgeRHS ]
node_stmt   :  node_id [ attr_list ]
node_id     :  id [ port ]
port        :  ':' ID [ ':' compass_pt ]
            |  ':' compass_pt
subgraph    :  [ SUBGRAPH [ id ] ] '{' stmt_list '}'
compass_pt  :  (N | NE | E | SE | S | SW | W | NW | C | _)
```

Tento popis je robustní a důkladný, určený i pro velké grafy. My si vystačíme s omezenou gramatikou, odebráním prvků, které nejsou v našem grafu obsaženy.

```
graph       :  DIGRAPH '{' stmt_list '}'
stmt_list   :  [ stmt [ ';' ] stmt_list ]
stmt        :  node_stmt | edge_stmt | attr_stmt
attr_stmt   :  (GRAPH | NODE | EDGE) attr_list
attr_list   :  '[' [ a_list ] ']' [ attr_list ]
edge_stmt   :  node_id edgeop node_id [ attr_list ]
node_stmt   :  node_id [ attr_list ]
```

Naše grafy jsou pouze orientované a obsahují jen definice hran a uzlů s jejich názvy, URL a formátovacími daty.

1.6 Grafy kurzů

Grafy obsahují dva typy hran. Hrana, která je označena jako *gray*, představuje link prvního výukového materiálu odkazující na druhý výukový materiál. Druhý typ hran označený jako *blue* definuje následující kapitolu.

Ukázka je na obrázku 5.1.

Uzly máme také dvou typů. První typ nám představuje výukové lekce a je označen prefixem *edux* a druhý typ představuje testy označené prefixem *progtest*.

Graf kurzu je nesouvislý graf, kde každý souvislý podgraf je kapitola v kurzu. Pro kurzy PA1 a PA2 je k dispozici ještě graf přechodů mezi kapitolami. Pro kurz PS1 bude muset být takový graf vytvořen.

Kapitola 2

Editační možnosti Moodlu

V této kapitole se zaměříme na to, jaké editační možnosti má Moodle a co všechno nám může poskytnout v podobě různých pluginů. Moodle pro uložení textových formátovaných dat používá HTML. Protože by úprava prostého HTML byla velice nepohodlná, tak Moodle poskytuje interaktivní textový editor, který dokáže vytvářet různá formátování bez použití HTML.

Moodle v základní verzi poskytuje 2 textové editory, jsou to *Atto HTML editor* a *TinyMCE HTML editor*. Další textové editory se dají přidat pomocí zásuvných modulů.

Zaměříme se na textový editor *Atto HTML editor*, který Moodle preferuje a je nastaven jako defaultní. My tedy budeme pracovat s ním.

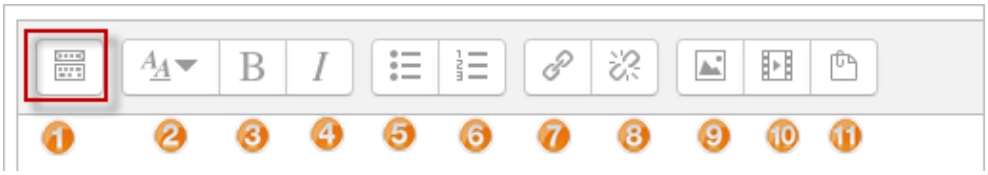
2.1 Atto HTML editor

Textový editor Atto má mnoho funkcí pro editaci uživatelského obsahu. S většinou funkcí je obeznámen každý, kdo používá nějaké textové editory.

Na následujícím obrázku 2.1 je ukázán každý editační prvek, který Atto v jednoduché formě poskytuje. Atto se přepne do rozšířené formy pomocí prvního tlačítka zleva.

Pro jednoduchou formu ikony znázorňují:

1. rozšíření (obrázek 2.2),
2. styly,
3. tučné písmo,
4. kurzíva,
5. nečíslovaný seznam,
6. číslovaný seznam,
7. přidání odkazu,
8. odebrání odkazu,
9. přidání obrázku,
10. přidání videa,
11. správa souborů.



Obrázek 2.1: Textový editor Atto

Pro rozšířenou formu (viz obr. 2.2) ikony znázorňují:

1. podtržené písmo,
2. přeškrtnuté písmo,
3. dolní index,
4. horní index,
5. zarovnání doleva,
6. zarovnání na střed,
7. zarovnání doprava,
8. zvýšit odsazení,
9. snížit odsazení,
10. editor rovnic,
11. speciální znaky,
12. editor tabulek,
13. odstranit veškeré formátování,
14. zpět (krok editace),
15. vpřed (krok editace),
16. kontrola dostupnosti,
17. čtení obsahu,
18. zobrazení HTML stránky.

Kontrola dostupnosti a *čtení obsahu* jsou pomocné nástroje pro handicapované.

Obrázek 2.2: Textový editor Atto rozšířený

2.2 Pluginy do Moodlu

Uvedu zde dva pluginy, které považuji za důležité pro udržení stejné funkcionality, jakou má současný e-learningový portál a kterou neposkytuje Atto HTML editor.

2.2.1 GeSHi Filter

GeSHi (Generic Syntax Highlighter) Filter je plugin, který zajišťuje obarvení syntaxe kódu.

Moodle pracuje s pluginem jako s textovým filtrem a při vykreslení rozpozná tag , tento plugin je zpracuje a vrátí Moodlu upravenou stránku s obarveným kódem.

Práce s pluginem je poměrně jednoduchá. Pro zvýraznění syntaxe je potřeba v textovém editoru Atto otevřít *rozšířené možnosti* a kliknout na *Zobrazení HTML stránky*.

```
1. #include <stdio.h>
2. int main()
3. {
4.     // printf() displays the string inside quotation
5.     printf("Hello, World!");
6.     return 0;
7. }
```

Obrázek 2.3: Ukázka obarvení kódu

Plugin GeGHi nám nabízí výběr jazyka, podle kterého chceme kód obarvit. Dále si můžeme vybrat, pokud chceme mít očíslované řádky u kódu.

Ukázka práce s pluginem:

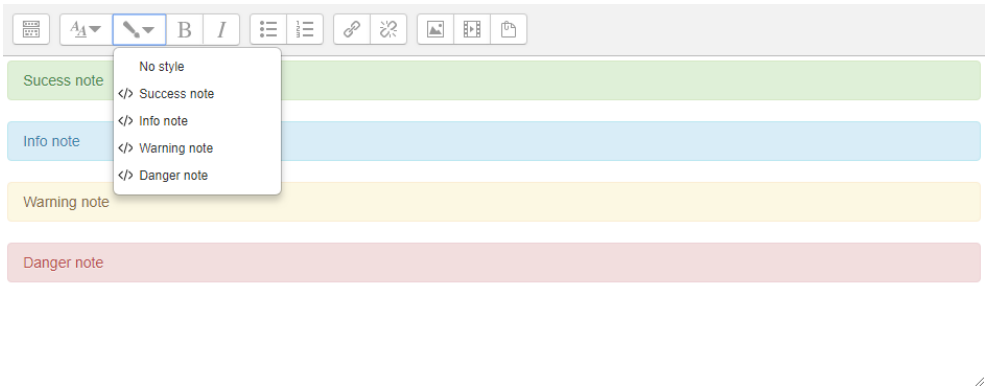
```
<span syntax=”code”>Zde se obarví kód</span>

<span syntax=”code” linenumbers=”yes”>
Zde se obarví kód
A řádky budou číslovány
</span>
```

2.2.2 Atto HTML editor: Styles

Dalším zajímavým pluginem je Styles přímo pro textový editor Atto. Tento plugin nás bude zajímat v souvislosti s pluginem do DokuWiki. Je to jeho Moodlovský ekvivalent.

Obrázek 2.4: Přidání tlačítka na vzhledy



Obrázek 2.5: Vzhled Atto Styles

Do textového editoru Atto přidá tento plugin další tlačítko pro výběr stylu. Práce se styly je intuitivní.

Kapitola 3

Návrh transformace DokuWiki

V této kapitole se budeme věnovat tranformaci dat do formátu, který podporuje Moodle. Data uložená na e-learningovém portálu e-learning.fit.cvut.cz jsou ve formátu DokuWiki, který jsme si představili v minulé kapitole. Moodle používá pro uložení a formátování dat HTML.

Naším cílem bude pro každý prvek formátování z DokuWiki nalézt ekvivalent v HTML a zároveň způsob, jak daný prvek transformovat na HTML. Nesmíme opomenout budoucí úpravy, proto každý prvek formátování, který nahrajeme do Moodlu, musíme být schopni dále upravovat a měnit.

Teoreticky můžeme každý prvek měnit pomocí funkce Atto HTML editoru, *zobrazení HTML stránky*, ale to je pouze krajní varianta, protože úpravy přes prosté HTML nejsou uživatelsky příjemné.

Začneme tedy s návrhem transformace základních DokuWiki prvků.

3.1 Základní formátování textu

Základní formátování bude velice jednoduché. Textový editor Atto podporuje všechny prvky základního formátování.

3.1.1 Tučné písmo

Tučné písmo můžeme v HTML zapsat buď pomocí párových tagů <b>, nebo pomocí párových tagů . Textový editor Atto používá na tučné písmo , proto ho použijeme na transformaci.

DokuWiki syntaxe:

```
**Takto se zapíše tučné písmo**
```

HTML syntaxe:

```
<strong>Takto se zapíše tučné písmo</strong>
```

3.1.2 Kurzíva

Kurzívu můžeme opět zapsat více způsoby. V HTML zapíšeme kurzívu pomocí párových tagů <i>, nebo pomocí párových tagů . Atto používá , proto je nasnadě, že to použijeme také při transformaci.

DokuWiki syntaxe:

```
//Takto se zapíše kurzíva//
```

HTML syntaxe:

```
<em>Takto se zapíše kurzíva</em>
```

3.1.3 Podtržené písmo

Podtržené písmo se zapíše v HTML pomocí párových tagů <u>.

DokuWiki syntaxe:

```
__Takto se zapíše podtržené písmo__
```

HTML syntaxe:

```
<u>Takto se zapíše podtržené písmo</u>
```

3.1.4 Neproporcionální písmo

Neproporciální písmo v HTML zapíšeme pomocí párových tagů <tt>. Neproporciální písmo bez problému transformujeme do HTML, ale problém nastane, když budeme chtít přidat další, nebo upravit nějaké stávající neproporciální písmo.

Textový editor Atto bohužel nemá žádnou funkci na neproporciální písmo a ani nedisponuje žádným pluginem, který by to uměl. Tato funkcionalita bude dostupná pouze přes *zobrazení HTML stránky.*

DokuWiki syntaxe:

```
"Takto se zapíše neproporciální písmo"
```

HTML syntaxe:

```
<tt>Takto se zapíše neproporciální písmo</tt>
```

3.1.5 Horní a dolní index

Syntaxe horního a dolního indexu pro DokuWiki a HTML se shoduje. To znamená, že tento prvek vůbec nemusíme transformovat a necháme ho v původní podobě.

3.1.6 Přeškrtnutý text

Syntaxe pro přeškrtnutý text je opět stejná pro HTML i DokuWiki. Tudíž nebudeme přeškrtnutý text transformovat.

3.1.7 Zalomení řádku

HTML používá tag
 pro zalomení řádku.

DokuWiki syntaxe:

\\

HTML syntaxe:

3.1.8 Odstavce

Odstavce v HTML definují párové tagy . V DokuWiki ale na rozdělení textu do odstavců párové tagy nepoužíváme, nýbrž jen dvojité odřádkování. Budeme z toho muset vytvořit párový tag .

První tag zapíšeme na začátek každého textu. Uzavírací tag zapíšeme na konci každého textu. Pokud uprostřed textu narazíme na dvojité odřádkování, tedy odstavec v syntaxi DokuWiki, zapíšeme oba tagy, zavírací i otevírací. Nejprve zapíšeme a poté .

```
DokuWiki syntaxe:                Moodle syntaxe:

                                 <p>
Lorem ipsum dolor                Lorem ipsum dolor
sit amet, consectetur            sit amet, consectetur
                         =>      </p><p>
adipiscing elit.                 adipiscing elit.
                                 </p>
```

3.2 Odkazy

Odkazy v HTML vytváříme pomocí párových tagů <a>. Pomocí atributu href= nastavíme cíl odkazu.

3.2.1 Externí odkazy

Pro externí odkazy se chová Moodle stejně jako DokuWiki. Externí odkazy jsou rozpoznávány automaticky, když je odkaz napsaný v celé formě včetně *http://*.

Externí odkazy tudíž nemusíme vůbec upravovat a můžeme je nechat ve stavu, v jakém jsou a Moodle se postará o zbytek.

3.2.2 Interní odkazy

U interních odkazů narazíme na první větší komplikace. Pokud bychom pouze doplnili data do tagu <a>, odkaz by vedl na starý e-learningový portál e-learning.fit.cvut.cz, nebo pokud by byl zapsán relativní cestou, tak by nám odkaz vedl na neexistující stránku na Moodlu. To nechceme.

Naším cílem bude URL směřující na e-learning.fit.cvut.cz přetransformovat na URL směřující na Moodle. Problém je, že v tuto chvíli nevíme, jaké URL budou mít stránky, na které chceme odkázat, protože ještě neexistují.

Řešení bude v podstatě jednoduché. Nejprve si vytvoříme prázdné stránky na Moodlu podle struktury v DOTu, tím získáme URL těchto prázdných stránek. Vytvoříme si převodní tabulku starých URL na nové URL. Teprve pak můžeme začít s transformací výukových lekcí.

DokuWiki syntaxe:

```
[[text odkazu|OLD URL]]
```

HTML syntaxe:

```
<a href="NEW URL">text odkazu</a>
```

3.3 Nadpisy

Nadpisy jsou v HTML vytvářeny pomocí párových tagů <h1> až <h6>. Pro naše účely bude stačit nadpis 5. úrovně, tedy <h5> (DokuWiki více nemá).

DokuWiki syntaxe:

```
===== Nadpis 1. úrovně =====
```

HTML syntaxe:

```
<h1>Nadpis 1. úrovně</h1>
```

3.4 Sekce

V HTML vytvoříme sekce pomocí tagu <hr>.

Textový editor Atto nezná sekce, ani žádný plugin nám v tomto případě nepomůže. Budeme se muset spokojit s tím, že úpravy sekcí budeme provádět v *zobrazení HTML stránky.*

DokuWiki syntaxe:

```
----
```

HTML syntaxe:

```
<hr>
```

3.5 Obrázky a jiné soubory

Textový editor Atto poskytuje pro práci se soubory rozsáhlou funkcionalitu včetně nahrání obrázku/souboru, zobrazení obrázku na stránce a další. Pro naše účely budou stačit první dvě funkcionality.

Nastává zde opět stejný problém jako u odkazů. Nicméně tento problém bude mít jiné řešení, protože součástí nahrávání obrázku do Moodlu je i jeho identifikátor (jméno obrázku). Víme tedy předem, jakou URL bude mít obrázek (bude to jeho jméno) a nemusíme tedy obrázky nahrávat předem a stačí je nahrát až společně s výukovými lekcemi.

Obrázky jsou v HTML vykresleny pomocí tagu <img>. Tag <img> obsahuje atributy *src*, kterým se nastaví URL obrázku, *alt*, alternativa, pokud se obrázek nepodaří načíst, *title*, kterým se nastaví popisek obrázku, *height* a *width*, pomocí nichž nastavíme rozměry obrázku a další, ale ty už nebudeme potřebovat.

DokuWiki poskytuje ještě možnost zarovnání obrázku. Na to využijeme tříd Atto *atto_image_button_left*, *atto_image_button_middle* a *atto_image_button_right*. Pro hezčí vykreslení obrázku rovněž využijeme Bootstrapové třídy *img-responsive* (Moodle sám využívá Bootstrap, proto není potřeba ho nějakým způsobem importovat).

DokuWiki syntaxe:

```
{{URL?450x150|Titulek}}
```

HTML syntaxe:

```
<img src="NEW URL" height="450" width="150" alt="Titulek"
title="Titulek" class="img-responsive>
```

3.6 Seznamy

Textový editor Atto umí jednoduše vytvářet seznamy a práce s nimi je intuitivní.

V HTML se seznamy zapíší pomocí párových tagů <ul>, resp. pro nečíslované, resp. číslované seznamy. Jednotlivé položky seznamů se už zapíší jednotně a to pomocí tagů <li>.

DokuWiki syntaxe:

```
  - Číslovaný seznam
    - Číslovaný seznam
      - Číslovaný seznam
  - Číslovaný seznam
```

HTML syntaxe:

```
<ol>
```

```
    <li>Číslovaný seznam</li>
    <ol>
        <li>Číslovaný seznam</li>
        <ol>
            <li>Číslovaný seznam</li>
        </ol>
    </ol>
    <li>Číslovaný seznam</li>
</ol>
```

3.7 Tabulky

Textový editor Atto obsahuje editor tabulek, takže práce s nimi je opravdu jednoduchá.

V HTML se tabulky zapíší pomocí tagů <table>. Uvnitř tabulky zapíšeme řádek pomocí tagů <tr>. Uvnitř řádku zapíšeme buňku tabuky pomocí <th>, resp. <td> pro hlavičkové buňky, resp. obyčejné buňky. Spojení dvou buněk zajistíme pomocí atributu buňky colspan, který nastavíme na hodnotu rovnající se počtu buněk, které nám zastupuje aktuální buňka.

Zarovnání textu v buňce docílíme pomocí atributu buňky aling, který nastavíme na hodnoty *right*, *center*, nebo defaultně *left*.

DokuWiki syntaxe:

```
^ Nadpis 1     ^ Nadpis 2             ^
^ Nadpis 4     | řádek 1 sloupec 2    |
^ Nadpis 5     |    řádek 2 sloupec 2 |
^ spojené sloupce                     |
^   Nadpis 6   | řádek 4 sloupec 2    |
```

HTML syntaxe:

```
<table>
    <tr>
        <th>Nadpis 1</th> <th>Nadpis 2</th>
        <th>Nadpis 4</th> <td>řádek 1 sloupec 2</td>
        <th>Nadpis 5</th> <td align="left">řádek 2 sloupec 2 </td>
        <th colspan=2>spojené sloupce</td>
        <th align="center">Nadpis 6</th><td>řádek 4 sloupec 2</td>
    </tr>
</table>
```

3.8 Předformátovaný text

Předformátovaný text nás vůbec nemusí trápit. Tato funkce byla v DokuWiki pouze pro zobrazení znaků, které mají jinak nějakou funkcionalitu. Pro nás to znamená, že s tímto textem nemusíme vůbec nic dělat a pouze ho nechat tak, jak je.

3.9 Bloky kódu

Jak už bylo řečeno, bloky kódu v DokuWiki jsou velmi hojně využívané k různým účelům díky svému specifickému vzhledu (viz obr. 1.1).

Moodle disponuje velice podobnou komponentou. Moodle prezentuje tuto funkci jako *pre-formated text* (viz obr. 3.1)

Obrázek 3.1: Moodle pre-formated text

3.9.1 Zvýrazňování syntaxe

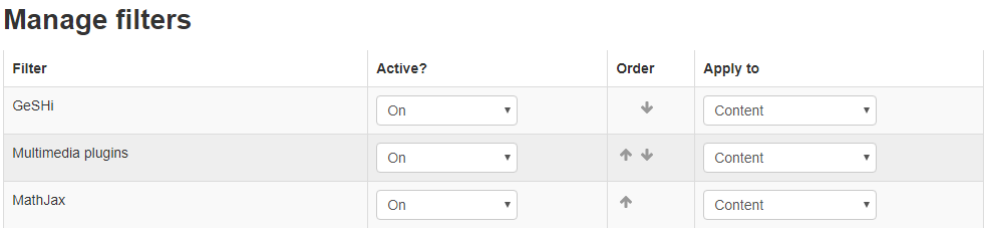
Zvýrazňování syntaxe kódu zajišťuje plugin GeSHi do Atto HTML editor.

Moodle pracuje s pluginem jako s textovým filtrem a při vykreslení rozpozná tag , tento plugin je zpracuje a vrátí Moodlu upravenou stránku s obarveným kódem.

Protože DokuWiki vykresluje LaTeX [4] i v kódech, měli bychom udělat totéž, aby se zachovalo stejné vykreslení. V Moodle nastavení *Site administration* vybereme v menu *Plugins* a dále *Filters/Manage filters*.

Zde nastavíme pořadí textových filtrů tak, aby se LaTeX vykreslil až po GeSHi. V našem případě postačí na LaTeX defaultní plugin MathJax [5]. V pořadí musí být tedy MathJax až za GeSHi.

Ukázka správného pořadí filtrů (Multimedia plugins vykresluje animace ve formátu swf, které obsahuje kurz PA1):

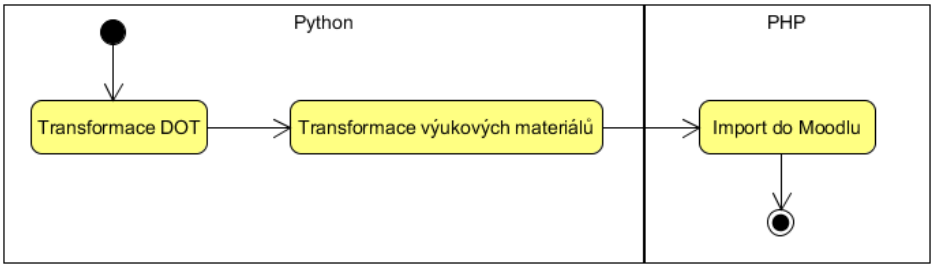


Obrázek 3.2: Pořadí filtrů

3.10 Důsledek transformace

Na první pohled bylo pořadí operací jasné a přímočaré. Nejprve transformujeme strukturu a výukové materiály. Následně do Moodlu naimportujeme výukové materiály podle definované struktury.

Pořadí operací je znázorněno v grafu na obrázku 3.3.

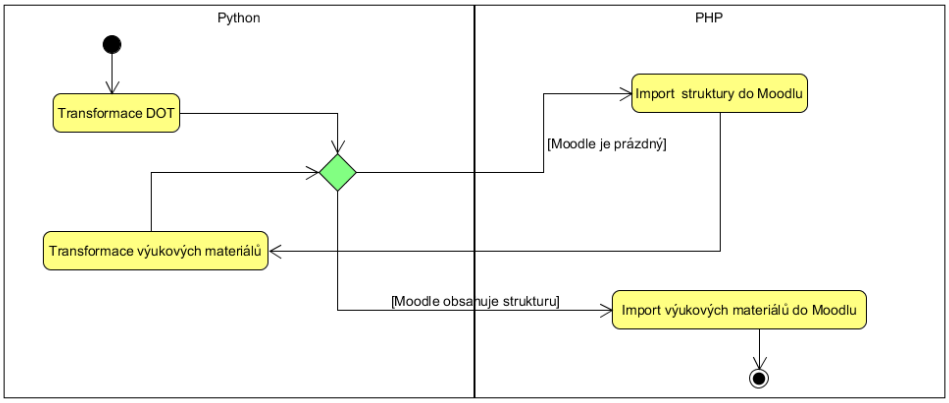


Obrázek 3.3: Pořadí operací

Kvůli transformaci odkazů se změnilo pořadí importování a transformace výukových materiálů, aby se vytvořily nejprve nové URL pro odkazy.

Z toho vyplývá, že budeme muset data importovat nadvakrát. Poprvé nahrajeme prázdné výukové materiály v pořadí, které nám určuje struktura. Napodruhé už naimportujeme transformované výukové materiály.

Nový sled operací znázorňuje graf na obrázku 3.4.



Obrázek 3.4: Nové pořadí operací

Kapitola 4

Návrh transformace testů

Z analýzy testů na DokuWiki jsme se dozvěděli, že se testy na serveru e-learning.fit.cvut.cz nevytvářejí, ale zasílají se požadavky na progtest.fit.cvut.cz. Testy vytvořené na progtestu se odešlou zpět na e-learning, kde je student vyplní a odešle k vyhodnocení zpět na progtest. Celá komunikace je znázorněna na obrázku 1.3. Z bezpečnostních důvodů je celá komunikace progtestu s e-learningovým portálem zajištěna na fixní adresu e-learning.fit.cvut.cz.

Nabízí se varianta, že vytvoříme stejné přesměrování otázek, jako je mezi e-learningovým portálem a Progtestem. Moodle v tomto směru nabízí plugin [6], který pracuje se vzdáleným vyhodnocováním testů. Bohužel tento plugin je specializován pro komunikaci s jinými testovacími prostředími. Především jde o STACK[7] a OpenMark[8]. Kdybychom chtěli plugin použít, musel by se upravit, a to je nad rámec této práce.

Jak jsem již popisoval v části týkající se testů, testy jsou generované a neexistuje tedy žádný balík otázek, který bychom mohli naimportovat do Moodlu. Proto tento balík potřebujeme vytvořit. Pan Ing. Ladislav Vagner, Ph.D., který spravuje server progtest.fit.cvut.cz, vygeneroval XML balík otázek z progtestu. V balíku otázek je ke každé otázce vygenerováno dvacet jejích náhodných variant.

Ukázka syntaxe XML exportu z progtestu pro otázku typu *MulStrict*:

```
<Question
  id="586"
  seed="200081"
  type="MulStrict">
  <QuestionText>text</QuestionText>
  <Choice isCorrect="Yes" inputField="No">text</Choice>
  <Choice isCorrect="No" inputField="No">text</Choice>
  <Choice isCorrect="Yes" inputField="No">text</Choice>
  <Choice isCorrect="Yes" inputField="No">text</Choice>
  <Choice isCorrect="No" inputField="No">text</Choice>
</Question>
```

Moodle podporuje import otázek v mnoha formátech. Pro nás nejzajíma-

vější formát bude *Moodle XML*. Tento formát je mnohem komplexnější, než v jakém, máme aktuálně uložené otázky, ale všechna klíčová data obsahuje. Naším cílem bude naše XML rozšířit do požadované syntaxe *Moodle XML*.

Ukázka syntaxe *Moodle XML* pro otázku typu *Multi choice*:

```
<question type="multichoice">
  <name>
    <text>opRelational</text>
  </name>
  <questiontext format="html">
    <text>text</text>
  </questiontext>
  <generalfeedback format="html">
    <text></text>
  </generalfeedback>
  <defaultgrade>1.0000000</defaultgrade>
  <penalty>0</penalty>
  <hidden>0</hidden>
  <single>true</single>
  <shuffleanswers>true</shuffleanswers>
  <answernumbering>123</answernumbering>
  <correctfeedback format="html">
    <text>Your answer is correct.</text>
  </correctfeedback>
  <partiallycorrectfeedback format="html">
    <text>Your answer is partially correct.</text>
  </partiallycorrectfeedback>
  <incorrectfeedback format="html">
    <text>Your answer is incorrect.</text>
  </incorrectfeedback>
  <shownumcorrect/>
  <answer fraction="0" format="html">
    <text>text</text>
    <feedback format="html">
      <text></text>
    </feedback>
  </answer>
  <answer fraction="0" format="html">
    <text>text</text>
    <feedback format="html">
      <text></text>
    </feedback>
  </answer>
  <answer fraction="100" format="html">
    <text>text</text>
    <feedback format="html">
      <text></text>
    </feedback>
  </answer>
</question>
```

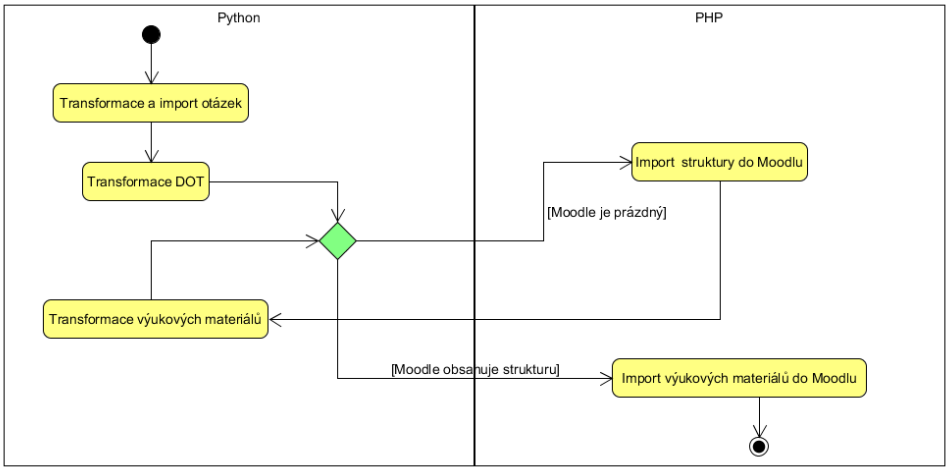
Všechny zde zobrazené prvky XML jsou povinné. Pokud by importované otázky některé z položek neobsahovaly, skončí import chybou.

Součástí exportu XML od pana Ing. Ladislava Vagnera, Ph.D., jsou i data o testech. Je v nich definováno, jaké otázky patří do jakého testu. Moodle bohužel neumí importovat celé testy, pouze otázky. Testy tedy budeme muset naimportovat stejně jako ostatní studijní materiály přes low-level API (podrobnosti v kapitole o importu do Moodlu).

Otázky budeme strukturovat do kategorií a podkategorií. Hlavní kateogrií pro nás bude *Default for BI PA1*, resp. *Default for BI PA2*, resp. *Default for BI PS1*. Jsou to defaultní kategorie, které existují v každém kurzu. Do této kategorie budeme vkládat podkategorie odpovídající kapitole v kurzu. Pro tyto podkategorie vytvoříme další podkategorie odpovídající danému testu v kapitole.

Import otázek pochopitelně musí předcházet importu testů. Naopak to být nemůže, protože nelze vytvořit test bez otázek. Znovu musíme upravit pořadí vykonávaných operací.

Nové pořadí operací znázorňuje graf na obrázku 4.1.



Obrázek 4.1: Nové pořadí operací s testy

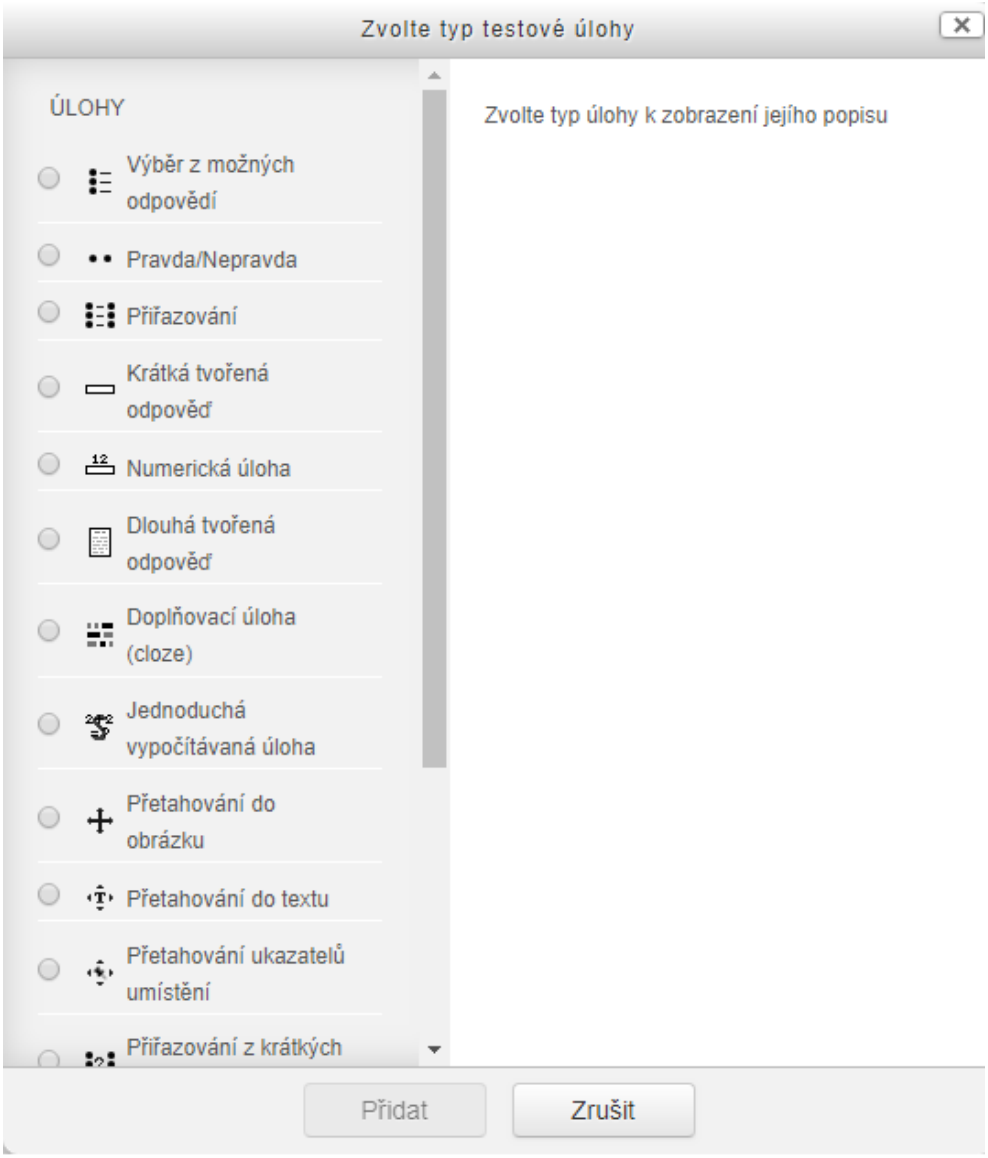
4.1 Typy otázek

Moodle podporuje mnoho druhů otázek. Pro naše účely budou stačit pouze otázky typu *Multiple choice*, *Short answer* a *Numerical*. Jediný typ otázky, kterým disponuje starý systém a Moodle neobsahuje, je *Kombinace výběru a doplnění*. Ostatní typy mají své ekvivalenty v Moodlu.

Typ otázky *Kombinace výběru a doplnění* budeme muset upravit do podoby Moodlovských otázek. Z otázky zachováme pouze doplňovací část. Znamená to, že pokud na otázku existuje odpověď, tak se otázka změní v prostou

doplňovací otázku. Pokud na otázku neexistuje odpověď, otázka se nepoužije do balíku úloh.

Ukázka typů otázek na obrázku 4.2:



Obrázek 4.2: Ukázka typů otázek

Kapitola 5

Parsování grafu struktury kurzu

Nyní už víme, jak vypadají jednotlivé prvky syntaxe jazyka DokuWiki. Také víme, jaké prvky HTML nám nabízí Atto HTML editor a známe alternativu pro každý prvek jazyka DokuWiki k HTML. Máme také strukturu kurzů v grafech.

Grafy obsahují dva typy hran. Hrana, která je označena jako *gray*, představuje link prvního výukového materiálu odkazující na druhý výukový materiál. Druhý typ hran označený jako *blue* definuje následující kapitolu.

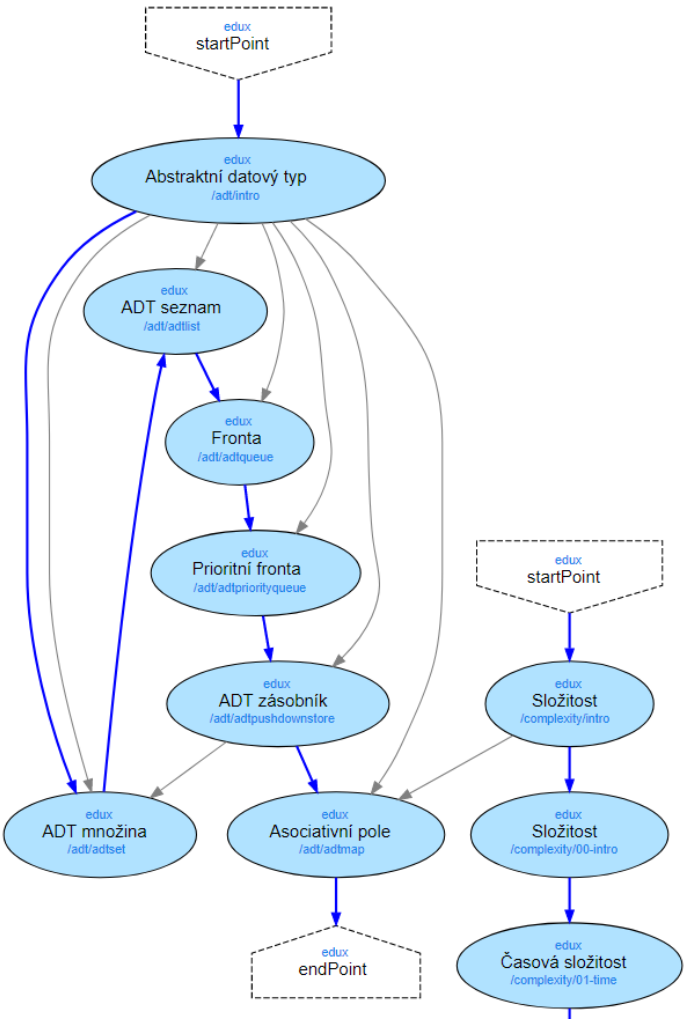
Uzly máme také dvou typů. První typ nám představuje výukové lekce, je označen prefixem *edux* a druhý typ představuje testy označené prefixem *progtest*.

Graf kurzu je nesouvislý graf, kde každý souvislý podgraf je kapitola v kurzu. Pro kurzy PA1 a PA2 je k dispozici ještě graf přechodů mezi kapitolami. Pro kurz PS1 budeme muset takový graf vytvořit.

V těchto grafech nyní budeme hledat průchod kurzem tak, abychom prošli celým kurzem a zároveň neprošli žádné téma dvakrát. Tím získáme pořadí kapitol, jak jdou po sobě a takto je poté nahrajeme do Moodlu.

Grafy máme uložené v textových souborech ve formátu DOT. Z grafu extrahujeme pouze hlavní cestu kurzem, ostatní informace nás nebudou zajímat. Na obrázku 5.1 je znázorněn průchod kapitolou *Abstraktní datový typ*.

Extrahování hlavní cesty provedeme ve skriptovacím jazyce Python, protože je na práci s textem ideální. Python [9] je vysokoúrovňový skriptovací programovací jazyk, nabízí nám mnoho funkcí pro práci s textem včetně možnosti využít regurální výrazy.



Obrázek 5.1: Ukázka grafu kurzu

Průchod grafem, jako výstup z našeho programu, bude soubor ve formátu JSON.

S tímto výstupem budeme pracovat při nahrávání prázdných stránek do Moodlu.

5.1 Hledání prvků grafu

V následujících ukázkách se pokusím vysvětlit funkci parseru DOTu. Následující kód je zjednodušen a jsou vynechány implementační detaily. Kód ani nemusí být plně funkční. Cílem ukázek je pouze zobrazit myšlenku algoritmu. Přesná implementace je ve zdrojových kódech na přiloženém CD.

Postupujeme přesně podle definice gramatiky DOT uvedené v předchozích kapitolách.

Soubor s grafem si načteme do proměnné *dot_lang*. Necháme *dot_lang* rozpadnout na jednotlivé *statements* oddělené pomocí středníku a uložíme si neprázdné prvky do proměnné *stmt_list*. Potom v cyklu pro každý *statement* zavoláme funkci *parse_stmt(stmt)*. Podle navrácené hodnoty buď přidáme uzel, nebo hranu.

```
dot_lang =  open("PA1.dot", mode="r").read()
stmt_list = [x.strip() for x in dot_lang.split(";") if x.strip()]
for stmt in stmt_list:
  res = parse_stmt(stmt)
  if "node" in res:
    nodes.add(res)
  elif "edge" in res:
    edges.add(res)
```

Funkce *parse_stmt(stmt)* rozpozná (kód je zjednodušen), zdali je *statement* hrana, nebo uzel a zavolá příslušnou funkci, která *statement* dále zpracuje.

```
def parse_stmt(stmt):
  if stmt.find("node") != -1:
    return "node", node_stmt(stmt)
  elif stmt.find("edge") != -1:
    return "edge", edge_stmt(stmt)
  else:
    return None
```

5.2 Uzly

Funkce *node_stmt(stmt)* rozdělí název uzlu a jeho atributy. Funkce vrátí jméno uzlu (odpovídá jménu výukového materiálu) a jeho atributy.

```
def node_stmt(stmt):  # DOT syntax - node_id [attr_list]
  node, attributes = stmt.split(" ")
  attributes = attributes.strip("[]").split(",")
  return node, attributes
```

5.3 Hrany

Funkce *edge_stmt(stmt)* rozdělí vstupní řádek na uzly a atributy. Uzly rozdělí na *node_from* a *node_to*. Pokud je v atributech, že je hrana modrá (následující lekce), vrátíme oba uzly, jinak nevrátíme nic.

```
def edge_stmt(stmt):  # DOT syntax - node_id -> node_id [attr_list]
  nodes, attributes = stmt.split("[")
  node_from, node_to = nodes.split("->")
  if "blue" in attributes:  # modra hrana je primarni
    return node_from, node_to
```

5.4 Nová struktura

Nyní načtená data transformujeme do jednodušší struktury, která nám poslouží při nahrávání dat do Moodlu jako pořadník importovaných studijních materiálů.

Funkce *create_new_structure(nodes, edges, starting_node)* projde přes všechny uzly (studijní materiály) kurzu a zaznamená procházené pořadí. Nastavíme si proměnnou *actual_node* = *starting_node*, kde *starting_node* je první lekce kurzu. Tuto proměnnou nastavíme znovu na konci cyklu while na *actual_node* = *edges[actual_node]*, kde *edges[actual_node]* je následující uzel pro *actual_node*. Cyklus while končí, když neexituje následující uzel, to nám zajistí kompletní průchod kurzem.

Uvnitř cyklu while upravíme data a uložíme je do proměnné *topics*. Proměnná *topic* je prefix *id*, které původně značilo cestu k souboru, nyní toho využijeme k rozdělení do kapitol. Pokud *topic* už existuje v *topics*, pouze ho rozšíříme.

```
def create_new_structure(nodes, edges, starting_node):
  topics = list()
  actual_node = starting_node
  while actual_node in edges:
    if actual_node.startswith("edux"):
      site = "edux"
      id = actual_node[5:].lower()
    else:
      site = "progtest"
      id = actual_node[9:].lower()
    topic = id.split("/")[0]
    url = nodes[actual_node]["url"]
    if topic not in topics:
      topics.append("edux", topic, [site, id, url])
    else:
      topics[topic][2].append([site, id, url)
    actual_node = edges[actual_node]
  return topics
```

Nad nově vytvořenou strukturou je provedeno ještě pár úpravných operací, které nám zaručí, že struktura je kompletní a obsahuje i stránky, které nejsou v hlavním průchodu kurzu (je na ně pouze odkazováno z jiných kapitol).

Tuto strukturu už jen uložíme do souboru.

```
output_file = open("structure.json", mode="w")
print(json.dumps(topics, indent=2), file=output_file)
```

Nová struktura bude ve formátu JSON s následující definicí:

```
{
  "Topics": [
    {
      "Type": "topic",
      "Name": "string",
      "Study materials": [
        {
          "Type": "lecture/test",
          "Name": "string",
          "URL": "url",
        }
      ]
    }
  ]
}
```

Podle této struktury budeme nahrávat kapitoly v daném pořadí do kurzu. Kapitoly obsahují studijní materiály. Podle typu studijního materiálu se vytvoří buď test, nebo lekce.

Kapitola 6

Import do Moodlu

Nyní, když už máme transformovanou strukturu kurzu a přesně víme, jak budeme transformovat výukové lekce a importovat je do Moodlu, můžeme začít podle diagramu na obrázku 4.1 importem prázdného kurzu do Moodlu. Jediné, co je u kurzu důležité, je vytvoření struktury podle výstupu z předchozího programu.

Moodle podporuje širokou škálu studijních a podpůrných materiálů (viz obr. 6.1) pro vytvoření kurzu. V našem případě využijeme pouze testy a stránky.

Import do Moodlu proběhne ve formě low-level API pluginem, jak je uvedeno v dokumentaci [10]. V této formě jsou všechny databázové tabulky zpřístupněny skrz low-level API, které zná všechny typy, vztahy a validační pravidla pro data v tabulkách. Na této úrovni není prováděna kontrola oprávnění z důvodů výkonu a složitosti.

Čtení z databáze budeme provádět pomocí *Moodle Data manipulation API* [11]. Zde budeme využívat funkcí *$DB→get_record($table,array())* a *$DB→get_records($table,array())*.

Zápisy do databáze nebudeme provádět přes *Moodle Data manipulation API*, ale přes knihovní funkce Moodlu, které za nás provedou všechny validační kontroly dat. Tyto funkce jsou: *course_create_sections_if_missing($course_id, $indexes)*, *course_update_section($course_id, $course_section_id, $record)*, *create_module($moduleinfo)* a *quiz_add_random_questions($quiz,0, $category→id, $count, true)*. Činnost těchto funkcí je zřejmá už z jejich názvu, u funkce *create_module($moduleinfo)* je výukovým modulem stránka, nebo test.

V následujících ukázkách se pokusím vysvětlit funkci programu. Následující kód je zjednodušen a jsou vynechány implementační detaily. Přesná implementace je ve zdrojových kódech na přiloženém CD.

Nejprve tedy načteme strukturu kurzu v JSON. Spočítáme počet kapitol a pokud je v Moodlu méně kapitol (kurz v Moodlu vždy po vytvoření obsahuje pár prázdných kapitol), přidáme chybějící kapitoly pomocí funkce *course_create_sections_if_missing($course_id, $indexes)*.

```
$input_file = fopen($PATH, "r");
$input_json = json_decode(
                fread($input_file, filesize($PATH)), true
              );
$course_sections = $DB->get_records("course_sections",
                    array("course" => $COURSE_ID));
if (count($course_sections) < count($input_json) ){
  $indexes = array();
  $cnt_section = count($course_sections)
  for ($i = $cnt_section; $i < count($input_json); $i++)
    array_push($indexes, $i);
    course_create_sections_if_missing($COURSE_ID, $indexes);
  }
}
```

V další části pro každou kapitolu (*$course_section*) nastavíme jméno podle starého identifikátoru (*$topic→name*). V kapitole budeme pro každou výukovou lekci, resp. test, vytvářet stránku, resp. test podle starého identifikátoru. Po nahrání výukového materiálu pomocí funkce *create_module($moduleinfo)* Moodle vrátí id prvku. V posledním kroku na staré identifikátory namapujeme nové id vrácené Moodlem.

```
foreach ($input_json as $topic) {
  $course_section = $DB->get_record("course_sections",
    array("course" => $COURSE_ID,"section" => $section_id++));
  $record = new stdClass();
  $record->name = $topic->name;
  course_update_section($COURSE_ID, $course_section, $record);
  foreach ($topic->study_materials as $study_material) {
    if ($study_material->type == "edux") {
      $moduleinfo = new stdClass();
      $moduleinfo->modulename = 'page';
      $moduleinfo->course = $COURSE_ID;
      $moduleinfo->name = $study_material->name;
      $page = create_module($moduleinfo);
      $URLS[$study_material->id] = $page->id;
    }
    else if ($study_material->type == "progtest") {
      $moduleinfo = new stdClass();
      $moduleinfo->modulename = 'quiz';
      $moduleinfo->course = $COURSE_ID;
      $moduleinfo->name = $study_material->name;
      $quiz = create_module($moduleinfo);
      $URLS[$study_material->id] = $quiz->id;
    }
  }
}
```

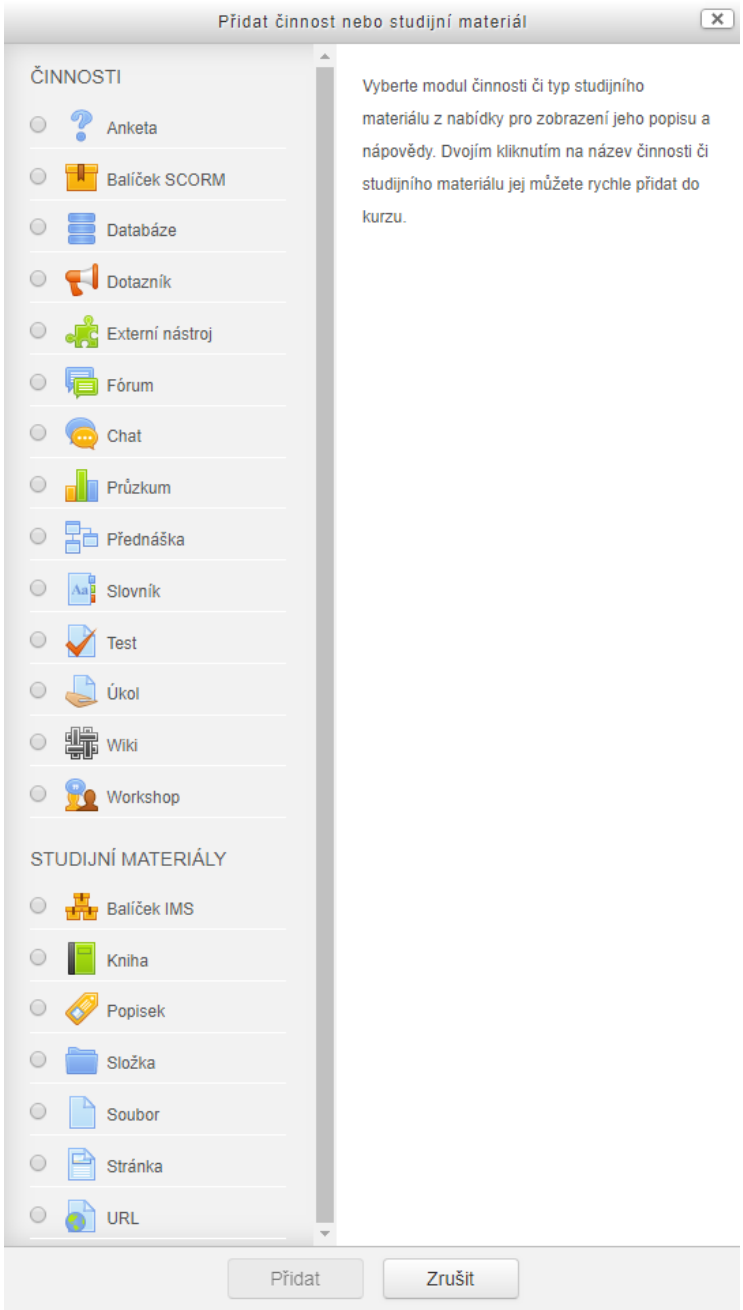
Nyní máme v Moodlu vytvořenou základní kostru kurzu. Obsahuje všechny výukové lekce i testy, ale prozatím jsou bez obsahu.

Namapované staré identifikátory a k nim odpovídající nové id z Moodlu převedeme do JSON struktury a uložíme na disk. Další operace s namapovanými identifikátory budeme provádět až při transformaci odkazů.

```
$fp = fopen($PATH . '/links.json', 'w') ;
fwrite($fp, json_encode($URLS,JSON_UNESCAPED_SLASHES));
fclose($fp);
```

Ukázka struktury JSONu s namapovanými identifikátory:

```
{
  "URLS": [
    {
      "oldID": "newID"
    }
  ]
}
```



Obrázek 6.1: Ukázka studijních materiálů

Kapitola 7

Transformace výukových lekcí

Transformace výukových lekcí bude nejrozsáhlejší z našich programů. Transformaci musíme naimplementovat pro každý prvek syntaxe jazyka DokuWiki. Pro každý prvek budeme potřebovat funkci, která nám daný prvek v textu vyhledá, a funkci, která nalezený prvek transformuje na HTML.

K transformaci budeme opět využívat Python, díky jeho funkcím pro práci s textem a podporou regulárních výrazů nám ušetří spoustu práce.

Nejdříve si nadefinujeme rodičovskou třídu *DokuWikiElement()*. Od této třídy budou dědit třídy, které představují jednotlivé prvky DokuWiki. Třída obsahuje statickou metodu *find_element*, která hledá výskyt daného prvku v textu. Druhá statická metoda *parse_element* zpracuje nalezený element a vrátí instanci třídy společně s číselnou hodnotou *end*, která značí konec zpracovávaného prvku.

```
class DokuWikiElement(object):
  @staticmethod
  def find_element(text, start, end, nested = False):
  """Returns first appearance of element in text[start:end]"""
  return -1

  @staticmethod
  def parse_element(text, start, end):
  """Returns instance of DokuWikiElement in text[start:end] and
                                    integer contains end"""
  default_element = ElementText(text=text[start:end])
  return default_element, end
```

Definice všech tříd je dostupná v přiložené dokumentaci a zde nebude uvedena. Budou zde uvedeny pouze ukázky z klíčových algoritmů programu.

Nadefinujme si proměnnou *DOKUWIKI_ELEMENTS*, která obsahuje pro každý prvek syntaxe DokuWiki odpovídající třídu.

```
  DOKUWIKI_ELEMENTS = [ElementLink, ElementBold, ElementHeading,
  ElementItalic, ElementUnderline, ElementMonospaced,
  ElementNewline, ElementParagraph, ElementHorizontalLine,
  ElementUnorderedList, ElementOrderedList, ElementMediaFile,
  ElementTable, ElementCode, ElementNote, ElementNoWiki,
  ElementSuperScript, ElementSubScript, ElementComment]
```

Každá třída je potomkem třídy *DokuWikiElement()* a přepíše metody *find_element()* a *parse_element*. Všechny třídy také budou definovat metodu *__str__(self)* pro výpis daného prvku v syntaxi HTML.

Díky dědičnosti a přepsání metod *find_element()* a *parse_element()* můžeme k objektům v *DOKUWIKI_ELEMENTS* přistupovat jako k "black-box".

Program začíná načtením namapovaných odkazů (výstup z předchozího programu) a načtením souborů k transformaci. Z adresáře, kde jsou uloženy stránky, načte všechny soubory a ty bude transformovat. Celý obsah souboru se zpracuje funkcí *parse_dokuWiki()*.

```
new_urls = json.load(open(PATH_TO_LINKS, mode="r"))
files = [f for f in listdir(PATH_TO_PAGES) if isfile(join(
                                    PATH_TO_PAGES, f))]
for file in files:
  text = open(file, mode="r").read()
  parsed_doc = parse_dokuWiki(text=text, start=0, end=len(text))
```

Funkce *parse_dokuWiki(text, start, end, nested=False)* běží v cyklu while, dokud zbývá nějaký text k parsování (*start*<*end*). Vždy pro každou třídu z proměnné *DOKUWIKI_ELEMENTS* najde, funkcí *find_element()*, první výskyt prvku jazyka DokuWiki.

První výskyt ze všech tříd se zpracuje funkcí *parse_element()*. Funkce *parse_element()* vrátí zpracovaný prvek a pozici konce tohoto prvku.

Další hledání bude probíhat až za tímto koncem, tj. nastavíme *start* = *end_element*. Pokračuje se znovu, dokud je nějaký text ke zpracování.

Po zpracování celého textu funkce vrací všechny prvky v chronologickém pořadí.

```
def parse_dokuWiki(text, start, end, nested=False):
  res_elements = list()
  while start < end:
    element = None
    first = -1
    for elem in DOKUWIKI_ELEMENTS:
      position = elem.find_element(text, start, end, nested)
      if (position < first or first == -1) and position != -1:
        first = position
        element = elem
    item, end_element = element.parse_element(text, first, end)
    start = end_element
    res_elements.append(item)
  return res_elements
```

7.1 Objektový návrh

Na obrázku 7.1 je znázorněn diagram tříd, včetně metod a proměnných. Všechny třídy dědí od třídy *DokuWikiElement*. Každá třída implementuje statické funkce *find_element()* a *parse_element()*. Každá třída dále implementuje funkci *__str(self)__* a statickou proměnnou *first_occurence*. Tato proměnná zajišťuje, že při opakovaném hledání vrátí předchozí nalezenou hodnotu a neopakuje zbytečně hledání, přesnější popis je níže.

7.1.1 Funkce find_element()

Funkce *find_element(text, start, end, nested)* najde pozici nejbližšího výskytu prvku v *text*u od pozice *start* do pozice *end*, jinak vrátí -1.

Tato funkce při hledání může využít *first_occurence*, pokud volání funkce není vnořené v jiném prvku (*nested*) a je hodnota *first_occurence* mezi *start* a *end*.

Pokud tato podmínka neplatí, vyhledá se nejbližší prvek pomocí funkce *str.find(str)*. Pokud nejde o vnořené volaní, tak se výsledek uloží do *first_occurence*.

Ukážeme si to na příkladu. Ukázka funkce *find_element()* pro *ElementBold* (tučné písmo):

```
@staticmethod
def find_element(text, start, end, nested=False):
  if start <= ElementBold.first_occurrence <= end and not nested:
    return ElementBold.first_occurrence
  position = text.find("**", start, end)
  if not nested:
    ElementBold.first_occurrence = position
  return position
```

7.1.2 Funkce parse_element()

Po nalezení nejbližšího prvku syntaxe se tento první prvek zpracuje. Funkce na zpracování prvku je *parse_element(text, start, end)*. Parametr *start* ukazuje vždy na začátek parsovaného prvku.

Funkce najde druhý párový koncový symbol, pokud je, a rekurzivně zavolá *parse_dokuWiki* tentokrát s parametrem *nested* nastaveným na *True*.

Rekurzivní volání se provádí jen pro prvky, které to dovolují, viz tabulka 1.1. Z textu mezi *start* a nalezeným koncovým symbolem se vytvoří instance dané třídy. Funkce vrací instanci třídy daného prvku a pozici konce tohoto prvku v *text*u.

Ukážeme si to na příkladu. Ukázka funkce *parse_element()* pro *ElementBold* (tučné písmo):

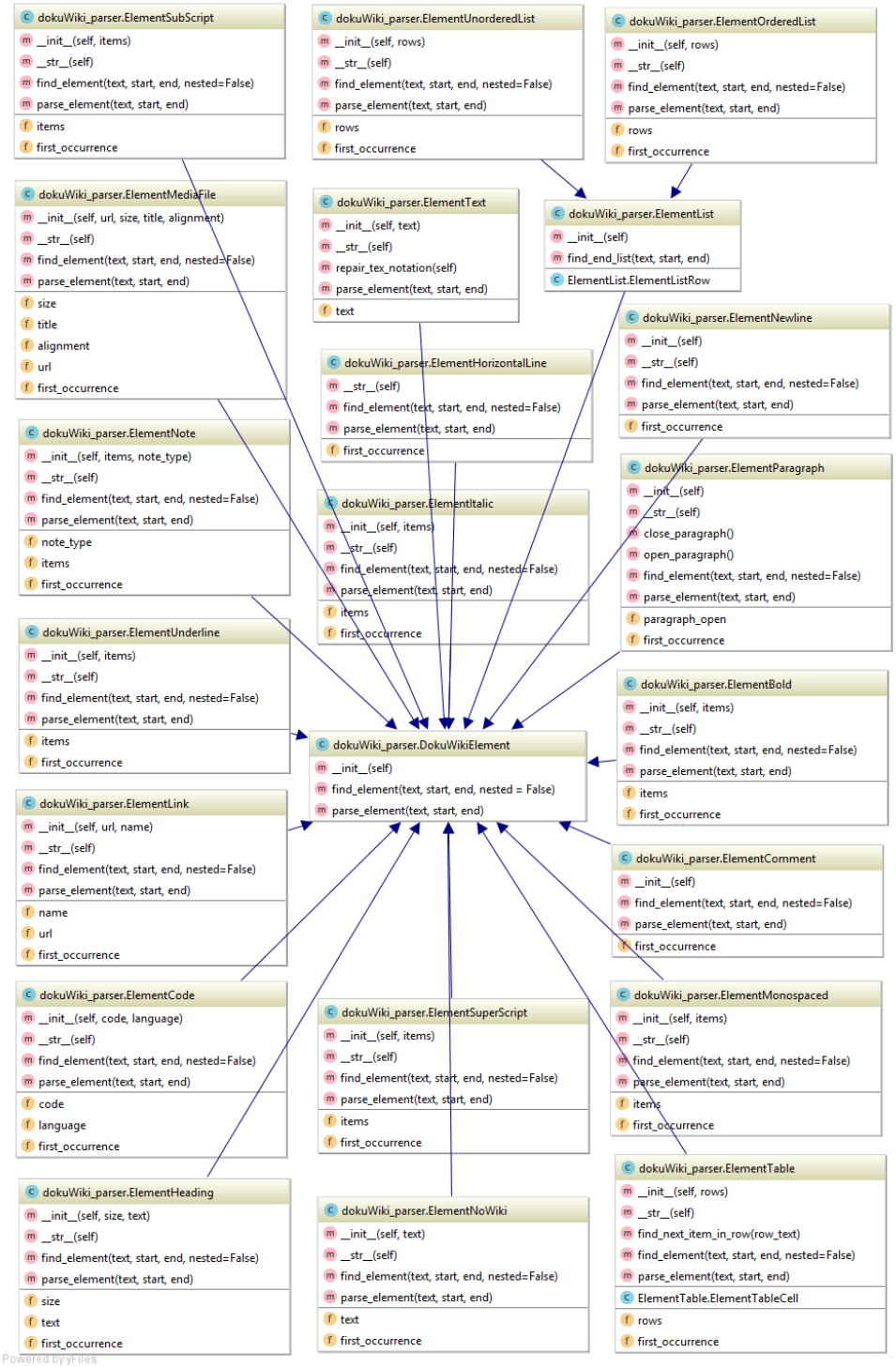
```
@staticmethod
def parse_element(text, start, end):
  end_bold = text.find("**", start + 2, end)
  if end_bold == -1:
    return ElementText(text=text[start:start + 2]), start + 2
  items = parse_dokuWiki(text, start + 2, end_bold, nested=True)
  bold = ElementBold(items=items)
  return bold, end_bold + 2
```

7.1.3 Funkce __str__()

Funkce __str__() vrátí string s daným prvkem v syntaxi HTML. V předchozích funkcích byly odstraněny symboly syntaxe DokuWiki a nyní jsou nahrazeny HTML symboly.

Ukážeme si to opět na příkladu. Ukázka funkce *__str__()* pro *ElementBold* (tučné písmo):

```
def __str__(self):
  bold_str = "<strong>"
  for item in self.items:
  bold_str += item.__str__()
  bold_str += "</strong>"
  return bold_str
```



Obrázek 7.1: Diagram tříd

Kapitola 8

Update dat na Moodlu

V této kapitole zakončíme nahrávání dat do Moodlu. V předchozích kapitolách jsme do Moodlu už nahráli strukturu výukových materiálů. Tato struktura ale neobsahovala žádná data, to znamená, že jsme do Moodlu výukové materiály nahráli, ale neobsahovala žádná data.

Tato data jsme vytvořili až následně, jak je popsáno v předchozí kapitole a zobrazeno na obrázku 4.1.

Nyní tedy zakončíme nahrávání tím, že nahrajeme transformované výukové materiály do Moodlu.

Na nahrávání do Moodlu použijeme opět low-level API, které jsme použili při nahrání prázdných výukových materiálů. Low-level API nám bude kontrolovat validační pravidla pro data v tabulkách. Na této úrovni není prováděna kontrola oprávnění z důvodů výkonu a složitosti.

Čtení z databáze budeme provádět pomocí *Moodle Data manipulation API* [11]. Zde budeme využívat funkcí *$DB→get_record($table,array())* a *$DB→get_records($table,array())*.

Zápisy do databáze nebudeme provádět přes *Moodle Data manipulation API*, ale přes knihovní funkce Moodlu, které za nás provedou všechny validační kontroly dat. Mezi tyto funckce patří *create_file_from_pathname($filerecord, $file)* a *course_update_section($course_id, $course_section_id, $record)*. Činnost těchto funkcí je zřejmá.

V následujících zjednodušených ukázkách se pokusím vysvětlit hlavní části programu. Cílem ukázek je pouze ukázat použité funkce. Přesná implementace je uvedena ve zdrojových kódech na přiloženém CD.

Nejprve tedy načteme strukturu kurzu v JSON a naši tabulku s převodem starých odkazů na nová URL.

```
$input_file = fopen($PATH, "r");
$structure = json_decode(
                fread($input_file, filesize($PATH)) , true
             );
$urls_file = fopen($PATH_TO_LINKS, "r")
$URLS = json_decode(
           fread($urls_file, filesize($PATH_TO_LINKS)), true
        );
```

V cyklu pro každou kapitolu, ze vstupního souboru se strukturou načteme transformovanou stránku kapitoly. Pro ni najdeme odpovídající kapitolu v databázi na Moodlu a změníme jméno kapitoly (*course_update_section($COURSE_ID, $course_section, $record)*). Původní jméno bylo id, které platilo na DokuWiki, nové jméno je definováno hlavním nadpisem v kapitole.

Dále načteme soubory, které souvisejí s danou kapitolou (jsou ve složce kapitoly), budeme je používat dále. Soubory se načtou pomocí funkce *get_files($directory)* popsanou níže.

Cyklus pokračuje dále.

```
foreach ($structure as $topic) {
  $page_file = fopen($topic["path"], "r");
  $page = json_decode(
            fread($page_file,
            filesize($topic["path"])),
            true
          );
  $course_section = $DB->get_record("course_sections",
      array("course" => $COURSE_ID, "section" => $cnt++));
  $record = new stdClass();
  $record->name = $page["name"];
  course_update_section($COURSE_ID, $course_section, $record);
  if (file_exists($PATH_TO_MEDIA . "/" . $topic["name"]))
    $files = get_files($PATH_TO_MEDIA . "/" . $topic["name"]);
```

Funkce *get_files($directory)* načte všechny soubory ze složky a všech jejích podsložek. Soubory z podsložky mají jako prefix jméno podsložky.

```
function get_files($directory){
  $items = scandir($directory);
  $files = array();
  foreach ($items as $item) {
    if ($item == "." or $item == ".." or $item == "Thumbs.db")
      continue;
    if (is_dir($directory . "/" . $item)) {
      $files_sub = get_files($directory . "/" . $item);
      $files = array_merge($files, $files_sub);
    } else
```

```
        $files [] = $directory . "/" . $item;
    }
    return $files;
}
```

Pokračujeme uvnitř předchozího cyklu, tudíž uvnitř kapitoly a nyní budeme nahrávat jednotlivé studijní materiály z kapitoly.

Načteme soubor s transformovanou stránkou a načteme z databáze v Moodlu odpovídající záznam.

Změníme jméno stránky a nastavíme obsah podle transformovaných dat. Pomocí funkce *update_module($page)* nahrajeme upravená data do Moodle databáze. V ukázce chybí další nastavení, která jsou důležitá pro Moodle, ale pro ukázku jsou zbytečná.

Cyklus pokračuje dále.

```
    foreach ($topic["study_materials"] as $study_material) {
      $page_file = fopen($study_material["path"], "r");
      $page_json = json_decode(
                    fread($page_file,
                    filesize($study_material["path"])),
                    true
                   );
      $new_url = substr($path, $LEN_OF_PATH, -4);
      $module = $DB->get_record("course_modules",
                  array("id" => $URLS[$new_url]));
      $page = $DB->get_record("page",
                array("id" => $module->instance));
      $page->name = $page_json["name"];
      $page->page = array("itemid" => $page->id,
                          "text" => $page_json["text"],
                          "format" => "1");
      update_module($page);
```

Pokračujeme uvnitř předchozího cyklu, tedy uvnitř výukového materiálu a nyní budeme nahrávat soubory (obrázky) pro studijní materiály. Soubory už jsme načetli dříve, protože jsou soubory uložené vždy společně pro jednu kapitolu.

Nyní se pro každý soubor z kapitoly pokusíme najít výskyt URL souboru v textu stránky. Pokud výskyt najdeme, znamená to, že se obrázek na dané stránce vyskytuje a my ho musíme přidat do Moodlu.

To uděláme pomocí funkce *create_file_from_pathname($filerecord, $file);*. Argumenty funkce jsou zobrazeny v ukázce.

```
      foreach ($files as $file) {
        if (strpos($page_json["text"],basename($file))!==false){
          $source = new stdClass();
          $source->source = basename($file);
          $filerecord = array(
```

```
            'contextid' => $page->id,
            'component' => 'mod_page',
            'filearea' => 'content',
            'source' => basename($file),
            'filepath' => "/",
            'filename' => basename($file)
          );
          $file_instance = $fs->create_file_from_pathname
                                        ($filerecord, $file);
      }
    }
  }
}
```

Tím končí hlavní smyčka programu.

Kapitola 9

Technické detaily zvolených technologií

Na závěr bych uvedl technické specifikace použitých systémů a technologií, aby nevznikl žádný problém s kompatibilitou. Technologie byly zvoleny na základě mých zkušeností s nimi a jejich použitelnosti pro naše potřeby. Dalším kritériem samozřejmě bylo, aby byly technologie kompatibilní s cílovou platformou Moodle.

9.1 Moodle

Verze Moodlu nasazeného na FIT ČVUT v Praze je 3.1.6., je proto jasné, že budeme optimalizovat náš program pro tuto verzi.

Další verzí Moodlu, která nás bude zajímat, je verze 3.4., nejnovější stabilní verze Moodlu. Moodle 3.4. přidává do Moodlu velice zajímavou funkcionalitu.

Jsou to linky na sousední výukové lekce v kapitole. Tato funkcionalita záměrně nebyla přenesena z původního systému. Důvod je prostý. Vyřeší to za nás v budoucnu aktualizace Moodlu. Naopak pokud bychom tuto funkcionalitu přenášeli, byl by zde problém při aktualizaci s dvojími linky.

9.2 Python

Python byl použit především pro jeho výbornou podporu funkcí pro práci s textem. V Pythonu jsme transformovali všechny výukové materiály z e-learningového portálu.

Všechny programy napsané v Pythonu jsou odladěné na verzi Python 3.6.

9.3 PHP

PHP jsme využili při nahrávání transformovaných dat do Moodlu a jejich následný update.

Při nahrávání dat bylo použito PHP 7.0.23.

9.4 MySQL

Moodle pro uložení dat využívá SQL databázi. Při naší práci byla použita databáze MySQL.

Verze MySQL použita na našem Moodlu je 5.7.19.

9.5 GIT

Jako verzovací systém byl použit GIT. V našem projektu jsme více verzí ani nemuseli použít, protože jsem na projektu pracoval sám. GIT nám spíše posloužil jako vzdálená záloha našich dat.

Celý projekt je dostupný na webovém repozitáři GitLab [12].

Závěr

Cílem této práce bylo zajistit přesun všech výukových materiálů ze serveru e-learning.fit.cvut.cz do Moodlu, nasazeného na FIT ČVUT v Praze. Transformace kurzu měla proběhnout tak, aby nový kurz byl co možná nejvíce totožný se starým. Důležité bylo zachování pořadí výukových lekcí a testů v kapitolách. Zároveň bylo potřeba zajistit totožné editační možnosti.

Práci jsem začal analýzou e-learningového portálu nasazeného na e-learning.fit.cvut.cz. Systém, na kterém běží e-learningový portál, je klon serveru EDUX upravený pro podporu e-learningu a jmenuje se DokuWiki.

Systém nebyl příliš vhodný pro e-learningové kurzy. Systém se spíše specializoval na vyváření dokumentací. Kurzy na e-learningovém portálu obsahovaly pouze dva druhy výukových materiálů. První byly výukové lekce a druhé byly testy.

Struktura kurzu na e-learningovém portálu byla popsána orientovanými grafy přechodů ve formátu DOT. Výukové lekce byly popsány značkovacím jazykem DokuWiki. Testy se vytvářely na serveru progtest.fit.cvut.cz a zasílaly se do e-learningového portálu přes XML zprávy.

Poté jsem analyzoval systém Moodle, jaké nám poskytne funkce, jaké má editační možnosti, jak se do něj zapisují data a v jakém jsou formátu. Moodle je e-learningová platforma, specializovaná na vytváření a běh e-learningových kurzů. Moodle všechna data ukládá do SQL databáze. Textová data jsou uložena v syntaxi HTML jazyka. Pro úpravu textových dat na Moodle z webového prohlížeče poskytuje Moodle interaktivní textový editor Atto. Do Moodlu se dají přidat další editační funkce, studijní materiály a další nastavení pomocí pluginů.

Když jsem zjistil, v jakém formátu jsou data na e-learningovém portálu a v jakém se ukládají na Moodlu, mohl jsem zahájit návrh transformace pro každý prvek syntaxe DokuWiki. Pro transformaci URL v odkazech, mezi výukovými lekcemi, jsem musel vygenerovat nová URL pro Moodle. Pro testy, které se generují a vyhodnocují na Progtestu, mi vygeneroval pan Ing. Ladislav Vagner, Ph.D., správce Progtestu, sadu otázek, které se mohou střídat

a mohou tak simulovat náhodně generovaná data.

Z grafu přechodů mezi studijní materiály jsem izoloval pouze jednu cestu. Tato cesta prochází každým uzlem grafu a představuje průchod kurzem, jak na sebe studijní materiály navazují.

Z testů uložených v XML se použily otázky pouze rozšířením o další atributy, které Moodle vyžaduje, a byly nahrány v XML formátu do banky otázek na Moodlu. Testy budou vytvořeny z těchto otázek při nahrávání dalších výukových materiálů.

Před transformací výukových lekcí jsem musel do Moodlu nahrát strukturu kurzu, aby bylo možné transformovat odkazy mezi výukovými lekcemi. Bylo potřeba, aby se vytvořily nové URL, na které budou odkazy ukazovat. Poté jsem transformoval všechny výukové materiály do syntaxe HTML jazyka.

Nakonec jsem všechny transformované výukové materiály nahrál do Moodlu. Struktura kurzu už byla vytvořena, proto jsem jen přidal upravená data přes low-level API.

Výsledný stav byl zkontrolován a disponuje stejnou funkcionalitou, jakou měl starý systém. Navíc obsahuje další možnosti úprav a přidání dalších výukových materiálů.

Stanovené cíle v rámci diplomové práce se podařilo splnit v plném rozsahu.

Literatura

[1] *dokuwiki [DokuWiki]*. [cit. 2018-04-22]. Dostupné z: https://www.dokuwiki.org/start?id=cs:dokuwiki

[2] Moodle.org: *Moodle - Open-source learning platform*. [cit. 2018-04-22]. Dostupné z: https://moodle.org/

[3] *The DOT Language*. [cit. 2018-04-22]. Dostupné z: https://www.graphviz.org/doc/info/lang.html

[4] LaTeX: *LaTeX Documentation*. [cit. 2018-05-07]. Dostupné z: https://www.latex-project.org/help/documentation/

[5] Moodle.org: *MathJax filter - MoodleDocs*. [cit. 2018-04-22]. Dostupné z: https://docs.moodle.org/27/en/MathJax_filter

[6] Moodle.org: *Opaque question type - MoodleDocs*. [cit. 2018-04-28]. Dostupné z: https://docs.moodle.org/34/en/Opaque_question_type

[7] Weidner, J.: *StudyStack*. [cit. 2018-04-28]. Dostupné z: https://www.studystack.com/

[8] University, T. O.: *Distance Learning Courses and Adult Education - The Open University*. [cit. 2018-04-28]. Dostupné z: http://www.open.ac.uk/

[9] [US], P. S. F.: *Welcome to Python.org*. [cit. 2018-04-22]. Dostupné z: https://www.python.org/

[10] Moodle.org: *Communication Between Components - MoodleDocs*. [cit. 2018-04-23]. Dostupné z: https://docs.moodle.org/dev/Communication_Between_Components

[11] Moodle.org: *Data manipulation API - MoodleDocs*. [cit. 2018-04-23]. Dostupné z: https://docs.moodle.org/dev/Data_manipulation_API

[12] GitLab: *Petr Pejsa / Diplomka - GitLab*. [cit. 2018-05-07]. Dostupné z: https://gitlab.fit.cvut.cz/pejsape1/diplomka

Příloha **A**

Seznam použitých zkratek

DOT Graph description language

HTML HyperText Markup Language

XML Extensible Markup Language

JSON JavaScript Object Notation

URL Uniform Resource Locator

PHP Hypertext Preprocessor

API Application Programming Interface

PA1 Programování a algoritmizace 1

PA2 Programování a algoritmizace 2

PS1 Programování v shellu 1

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `latex` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Pejša_Petr_2018.pdf` ..... text práce ve formátu PDF