**Имена:**  Димитър Добринов Кюртов **фн:**  *81797***Начална година:** *2022* **Програма:** бакалавър, (КН) **Курс: 4**   
**Тема: 14.1  
Дата: 22-13-01 Предмет: w15prj\_KN \_final имейл: dimitarkiurtov@gmail.com**

**преподавател:** доц. д-р Милен Петров

**Предаване:** Задачата се предава в архив с попълнен настоящия документ, проекта/проектите с кодовете, README.txt файл, който описва съдържанието на архива; папка с допълнителни компоненти и използваниресурси**, архива да се казва 9999\_project\_final.zip. (Успех!). (Редактирайте маркираните зони в жълто с коректната информация)**

# ТЕМА: xml2emmet и emmet2xml

## 1. Условие

Да се създаде онлайн платформа, която да позволява генерирането на xml код по даден emmet и обратното. Да има възможност за запазване на заявките към конкретен потребител и преглеждането им.

## 2. Въведение

...

## 3. Теория

За дизайна е избран MVC модела. Централният модел в приложението е синтактичното дърво на кода, който трябва да се преобразува. И двата типа се преобразуват до този модел и след това само начина на изпечатване е различен. Подхода е избран, понеже двата езика или частите от тях, които поддържаме са взаимно заменими и е логично да се свеждат до еднаква структура. Изгледите ни се генерират динамично на база данните в модела, извлечени от базата данни или от текущото преобразувание. Контролер модула ни отговаря за подготовка на суровите данни да бъдат сведени до синтактичното им дърво както и за връзката и заявките към базата. Използвани са сесии за коректната аутентикция през различните страници. Всички заявки към базата или за превеждането на кода се изпълняват с ajax заявки към сървъра, за да не се налага презараждане и за по-добро преживяване на потребителя.

## 4. Използвани технологии

Html, css, js, php, mysql

## 5. Инсталация и настройки

Във controllers/credentials.php се избират съответните данни за връзката с базата. После от браузъра отиваме на адреса на install.php, например: <http://localhost/emmet_translator/install.php> , ако хоста е localhost. Базата се създава автоматично, ако не съществува и се пълни с тестови записи от sql файла, който също се намира в root папката.

## 6. Кратко ръководство на потребителя

http://localhost/emmet\_translator/views/home\_view.php:

Graphical user interface, application

Description automatically generated

В Еmmet to xml блока въвеждате скрипт в emmet формат и с tab бутона се генерира xml кода в полето под него. Аналогично и за десният блок.Поддържаниете emmet атрибути са: Child: >, Sibling: +, Grouping: (), Multiplication: \*, ID and CLASS attributes, Custom attributes, Text: {}. Техните значения могат да бъдат намерени на <https://docs.emmet.io/cheat-sheet/>. За xml се поддържат id, class, custom\_attribute, content. Няма валидация на входа. При коректни входни данни се очаква коректен изход. При некоректни входни данни, поведението е недефинирано. Ако има клас/id/атрибут, който съдържа „?“ ще изведе грешен резултат. Конфигурационният json се прилага и за двете преобразувания. При зареждане на страницата се генерират всички възможни атрибути на конфигурацията. Ако някой от ключовете липсва в конфигурацията, се приема че има стойност “false”. “history”=”true” е възможно само за логнати потребители, иначе се променя на false. Конфигурацията указва кои атрибути да се визуализират, но всички се парсват и запазват в базата!!! Стандартният първи ред за xml и html документ се игнорира. Attribute свойството на конфигурацията оказва да се генерират всички такива атрибути с техните пътища до тях. Засега се поддържа само id и class. Подържат се коментари, като може да се окаже началният и крайният разделител, разделени със спейс от comment\_delim свойството. Коментарите могат да се включват и изключват от comment атрибута. Кода на заявката и конфигурацията могат да се заредят от url адрес, чрез config\_url и data\_url атрибутите. Чрез ouput\_result може да се избере дали да се генерира резултата на самата страница или да се изпрати на адрес, чрез get заявка. callback\_url оказва на кой адрес да се изпрати съответната заявка. Login и register бутона ни пращат в съответните форми. Има клиентска и сървърна валидация с подходящи съобщения при неизпълнението им. При успешно влизане се появява възможност за запазване на заявките с бутон. Една заявка, със сответната конфигурация може да бъде запазена веднъж от даден потребител. При опит за повторен запис се показва подходящо съобщение:

Graphical user interface, application

Description automatically generated

Страницата history ни позволява да гледаме заявките, които сме запазили като укажем колко заявки искаме да видим (сортирани по времето им на запазване) както и от кой тип (xml или emmet). Ако не укажем лимит се генерират всички направени от нас заявки. Имаме и опция да заредим стара заявка от load бутона на съответният ред. За тестване използвайте потребител “lmao” “Lmao123”, който има много запазени заявки. При опит за нелогнат потребител да достъпи тази страница се праща на логин формата с подходящо съобщение за грешка.

Graphical user interface, application, table

Description automatically generated

## 7. Примерни данни

lmao:div>(header#hea[z="5" d="5"].p1{click}.p2[j="12"][x="3" y="4"]>ll:ul>li[x=3]\*2>a)+{lmap}+dd:footer>pp{33}

<lmao:div>

<header id="hea" class="p1 p2" z="5" d="5" j="12" x="3" y="4">

click

<ll:ul>

<li x=3>

<a></a>

</li>

<li x=3>

<a></a>

</li>

</ll:ul>

</header>

lmap

<dd:footer>

<pp>33</pp>

</dd:footer>

</lmao:div>

Тези два примера тестват всички свойства на системата. В базата има още доста, които могат да се проверят и лесно да се заредят от history страницата.

## 8. Описание на програмния код

## w15prj\_KN\_REQ\_final - информация за проекта, как да се ползва и какво представлява.

## queries.sql - файла, който install.php използва, за да зареди базата

## install.php - скрипт, който зарежда тестова база с данните от /controllers/credentials.php

## LICENCSE - лиценза на приложението

## views - съдържа всички изгледи, динамично генерирани с php. Съдържа стилизирането им в views/css. Съдържа анимации и асинхронните заявки към сървъра в views/jss.

## models/tree.php - централният модел за синтактичен node, до който се свеждат изразите от emmet и xml.

## controllers

## credentials.php - данни за базата

## db.php - връзка към базата

## emmet\_to\_xml.php и xml\_to\_emmet.php - основният контролен поток за трансформацията от израз в синтактично дърво

## utility.php - съдържа всички функции за обхождане и обработване от стринг към дърво и обратното

## history.php - заявките и динамичното генериране на страницата с историята

## login.php - логиката за логване на потребител

## register.php - логиката за регистриране на портебител

## logout.php - логиката за излизане на потребител

## persist\_query.php - заявките към базата за запазване на фрагментите код от home страницита9. Приноси на студента, ограничения и възможности за бъдещо разширение

Проекта може да се разшири да поддържа повече елемнти от синтаксиса на двата езика. Това ще стане лесно, просто трябва да се добавят подобни на вече съществуващите функции за парсване на конкретните елементи и запазването и в същата структура. При голямо потребление, може да се на направи изкуствен интелект, който спрямо данните от базата да дава персонални подсказки на всеки потребител, докато пише код.

## 10. Какво научих

Как да ползвам еммет, за да си пиша по бързо xml-a.

Как да ползвам ajax-и и как работят

Че не искам да пиша на php

## 11. Използвани източници

...

Предал (подпис): 81797, Димитър Кюртов, КН, 1

/*фн, имена, спец., група*/

Приел (подпис): ………………………….