Интеграция на LESS интерпретатор и SharePoint сървър

Мирослав Михайлов, Цветелин Павлов

Integration of LESS interpreter in SharePoint server: *Because of the rising adoption of HTML 5, web applications became more and more demanding. Languages that are building blocks of the web became more complicated and sophisticated. Since the main focus in HTML 5 markup is to be semantically correct, more and more responsibilities are delegated to CSS, especially in complex environments like SharePoint. Although CSS is very powerful it lacks basic language characteristics like inheritance, variables, functions etc. That is why non-standard languages like LESS and Sass became more popular. This paper describes the possibilities for integration of LESS preprocessor with SharePoint server without losing any SharePoint or LESS capabilities.*

Key words: *interpreter, CSS, integration, SharePoint server, LESS, preprocessor, HTML5.*

ВЪВЕДЕНИЕ

Cascading Style Sheets (CSS) е декларативен описателен език, основна съставна част на съвременната уеб мрежа. Използва се заедно с HTML и служи за визуално стилизиране на обектите (таговете) декларирани в HTML документите [1]. CSS е приеман предимно като статичен актив. Веднъж написани, CSS документите обикновено не подлежат на допълнителна обработка, а само се доставят от уеб сървъра до уеб агента (браузера) [2].

С развитието на мобилните платформи и навлизането на HTML 5 спецификацията, изискванията към стилизирането на уеб страниците стават все по-големи и разработката на CSS все по-трудна, тъй като възможностите на езика се оказват недостатъчни. Затова, заедно с нарастващата популярност на програмните рамки за HTML и CSS, започват да навлизат и езици от по-високо ниво, които се интерпретират до CSS преди доставянето до браузера или вътре в самата HTML страница с помощта на JavaScript. Често тези езици са наричани “динамичен CSS” или „метаезици“, а интерпретаторите се определят като препроцесори, тъй като интерпретацията се свежда до превод от един език на друг и се извършва преди истинската интерпретация на CSS от уеб агента [2, 3, 4].

Примери за такива езици са Sass и LESS. Първият е сравнително по-популярен, тъй като е добре внедрен в платформите за разработка на уеб приложения на Ruby, но е по-ограничен като възможности, докато LESS е по-либерална спецификация, която притежава няколко версии за различни среди за разработка и повече гъвкавост [5, 6]. По същата аналогия съществуват и препроцесори за JavaScript, най-популярният от които е CoffeeScript, а наскоро Microsoft пуснаха свой вариант, наречен TypeScript [7, 8].

КРАТКО ОПИСАНИЕ НА LESS:

LESS е динамичен език за стилизиране, който често е реферирan като LESS CSS. Той е надмножество на CSS. С други думи – всеки CSS е валиден LESS, но обратното не е винаги вярно.

LESS надгражда CSS като добавя възможности за използване на променливи, миксове (mixins), влагане и наследяване, функции и операции, области на наименования и пакети.

Променливите са нетипизирани, могат само да бъдат декларирани и използвани. За да се смени стойността на една променлива, тя се предекларира.

Таблица 1. Променливи в LESS.

LESS код	Изходящ CSS код
`@myColor: #6c94be;` `#header { color: @myColor; }`	`#header { color: #6c94be; }`

Миксовете се приемат като механизъм за наследяване в LESS езика и позволяват всички декларирани правила в един селектор да бъдат включени в друг. Това, заедно с променливите, е най-използваната възможност на езика и най-честата причина да се прибягва до употребата му. Разликата между миксовете и нормалното наследяване е, че в LESS бащиният селектор може да се параметризира, което прави миксовете нещо средно между наследяване и функции.

Таблица 2. Mixins в LESS.

LESS код	Изходящ CSS код

`.underlined(@border-width: 2px) {`
`  color: black;`
`  border-bottom: @border-width solid red;`
`}`
`.default {`
`  .underlined;`
`}`
`.thicker {`
`  .underlined(5px);`
`}`
	`.default {`
	`  color: black;`
	`  border: 2px solid red;`
	`}`
	
	`.thicker {`
	`  color: black;`
	`  border: 5px solid red;`
	`}`

Миксът в случая е селектора .underlined, който има един параметър – ширината на границата, която по подразбиране е 2 пиксела.

Селекторът .default наследява .underlined и получава граница с дебелина 2 пиксела, а селекторът .thicker наследява .underlined, но указва стойност на параметъра и получава граница от 5 пиксела.

Миксовете в LESS езика включват още възможности като филтриране по шаблонен аргумент, предефиниране с различна сигнатура (overloading) и условно наследяване (guard expressions).

LESS позволява селекторите да се влагат един в друг, което изключително много допринася за прегледността на изходния код. Чрез влагане се образуват и т.нар. пакети (bundles), които са аналог на пространствата от наименования (namespaces) от другите езици и също допринасят за прегледността на изходния код, но и подпомагат преизползването му.

Таблица 3. Влагане на селектори и използване на пакет.

LESS код	Изходящ CSS код

`.bordered {`
`  &.float { float: left; }`
`  .top { margin: 5px; }`
`}`
`#smallBundle {`
`  .make-red { color: red; }`
`  .make-blue { color: blue; }`
`}`
`H1 {`
`  #smallBundle > .make-red;`
`}`
	`.bordered.float {`
	`  float: left;`
	`}`
	
	`.bordered .top {`
	`  margin: 5px;`
	`}`
	
	`H1 {`
	`  color: red;`
	`}`

Езикът позволява още интерполация на символни низове, използване на JavaScript функции, както и набор от стандартни, често използвани функции и импортиране на други LESS файлове. Всички тези възможности правят езика изключително подходящ за създаване на CSS платформи като например Twitter Bootstrap, която е използвана в социалната мрежа Twitter [9].

МОТИВАЦИЯ И ПРЕДИЗВИКАТЕЛСТВА ПРИ ИНТЕГРАЦИЯТА

В контекста на системата за управление на онлайн съдържанието на сайтовете на университета, използването на приложна рамка за стилизиране като Twitter Bootstrap е желателно, защото скъсява времето за разработка, прави кода преизползваем и управлението му - улеснено. Рамката подпомага разработката, като свежда до минимум усилията за уеднаквяване на потребителския интерфейс в

различните браузери. В допълнение позволява да се използват готови стилове за разполагане, включително отзивчиво разполагане (responsive layout), както и множество готови стилове за формуляри, контроли и други елементи на потребителския интерфейс.

Предизвикателства:

1. **SharePoint.** SharePoint включва множество компоненти, които спомагат за улеснена редакция и контрол на съдържанието (лента с инструменти, валидация, проверка на правописа, теми). Неправилното използване на външни компоненти може да доведе до влошена практическа работа на потребителите или до невъзможност за употреба на заложените функционалности.
2. **Теми.** SharePoint включва т. нар. машина за теми (theme engine), която позволява да се променят изцяло цветовете и шрифтовете на сайтовете. Необходимо е машината да функционира след интеграцията с LESS интерпретатора. Машината за теми работи чрез специални коментари в съдържанието на CSS файловете и при превключване на тема създава нова версия на същия файл. Включването му в изходния код на страницата става чрез специален сървърен контрол, което на практика означава, че по време на разработка не е явно къде ще се намира файла и как ще се казва.
3. **Физически и виртуални файлове.** SharePoint позволява редакция на CSS файлове. По подразбиране повечето CSS файлове се намират на файловата система на сървъра, но ако един файл бъде променен, той се виртуализира (ghosting) и се разполага в базата данни. Когато такъв файл бъде заявен, SharePoint разбира къде е разположен и ако е в базата данни, го извлича оттам и го доставя на браузера прозрачно за уеб сървъра. Необходимо е тази функционалност да се запази, за да не се губи гъвкавостта на платформата.
4. **Архитектурата на SharePoint.** Интеграцията трябва да стане чрез „официалните“ инструменти и според добрите практики за разработка на решения за SharePoint. При сървърна ферма с няколко сървъра, SharePoint се грижи да синхронизира промените по уеб приложенията на различните сървъри, затова е необходимо интеграцията да бъде извършена чрез SharePoint сървъра, а не на по-ниско ниво (примерно директно върху лежащия отдолу IIS).

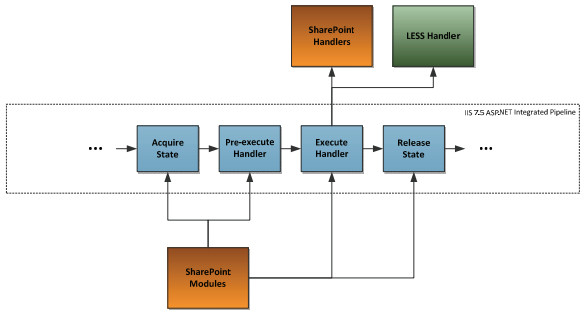
ВЪЗМОЖНОСТИ ЗА ИНТЕГРАЦИЯ

Интеграция в браузера чрез JavaScript. Официалният препроцесор на LESS е разработен на JavaScript и позволява LESS файловете да се интерпретират направо в уеб страницата, без да е необходима някаква промяна по сървърите [6].

Този метод е гъвкав, бърз и устойчив на промени. В много уеб приложения това е предпочитаният начин за интеграция на LESS интерпретатор. Основните му недостатъци са, че затруднява отстраняването на грешки (няма debug възможности) и натоварва браузерите при рендиране на страницата. Подходящ е при леки сайтове, малка скорост на трафика или слаб сървър и при невъзможност за контрол на уеб сървъра (например при облачен хостинг). При SharePoint този метод не е подходящ, тъй като SharePoint сървърът включва приложен интерфейс за работа с обектния му модел през JavaScript, който може да изпадне в конфликт с други заредени JavaScript скриптове, а и сам по себе си е достатъчно тежък. Допълнителното натоварване на браузера увеличава значително времето за рендиране и драстично влошава практическата работа на потребителите.

Интеграция в уеб сървъра чрез обработчик на HTTP заявки. Освен официалният JavaScript препроцесор, съществуват и няколко интерпретатора на

LESS, предназначени за различни среди. За .NET може да се използва DotLess [10] библиотеката, която включва и HTTP обработчик, готов за интегриране с IIS сървъра. Обработчиците на HTTP заявки са начин за разширение на интегрирания pipeline на IIS, чието основно приложение е да отговарят за обработката на определен ресурс, заявен през уеб сървъра. Точно такъв е случаят при LESS файловете.



Фиг. 1 – DotLess интерпретатор като обработчик на HTTP заявки.

Предимствата на този метод са:

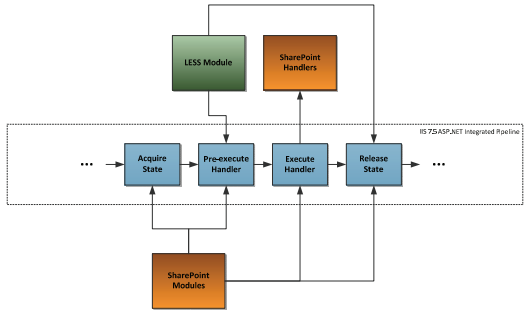
- Интеграцията става прозрачно за SharePoint, тъй като се използва „естествения“ механизъм за разширяване на функционалностите на уеб сървъра;
- SharePoint разполага с приложен механизъм за инсталиране на HTTP обработчици на всичките сървъри, участващи във фермата;
- Може да се използва готов проект с отворен код, който включва обработчик и интерпретатор. Проектът има силна общност и развитието му продължава.

Като недостатък може да се изтъкне, че обработчикът се приема до голяма степен като „черна кутия“, няма контрол върху входа на интерпретатора или върху изхода и процесът на интерпретация остава скрит. Затова ще е трудно да се използват функционалностите за теми в SharePoint. Въпреки, че машината за теми може да работи с LESS файловете, то няма как стойността на даден цвят или шрифт да се присвои на променлива от LESS файла, което обикновено е желаният начин да се работи с теми.

Важно е да се отбележи за този метод на интеграция, че е необходимо да се смени класът, отговарящ за намиране на файловете, които да бъдат доставени от уеб сървъра. Това е т.нар. path provider и също е точка за разширяване на IIS уеб сървъра. По подразбиране се използва path provider, които работи с файлове на диска. При SharePoint това не е задължително, може даден файл да се намира и в базата данни, затова трябва да се използва VirtualPathProvider класът в ролята на path provider.

Интеграция в уеб сървъра чрез HTTP модул. Този метод е сходен с предходния, но вместо обработчик се използва HTTP модул. Разликата е, че модулът има мощта да се прикачи към всяка една стъпка от обработката на заявка през интегрирания pipeline. Изборът на момент, в който да се включи модулът е изключително важен и трябва да се подбере внимателно, в зависимост от желания резултат. Един възможен момент за включване на модул за интерпретация в pipeline-а е при настъпване на събитието „Pre-execute Handler“, точно преди заявката да бъде подадена на HTTP обработчиците. Така ще може да се използват всички предимства на сървъра, включително автентикация, оторизация и кеширане, но

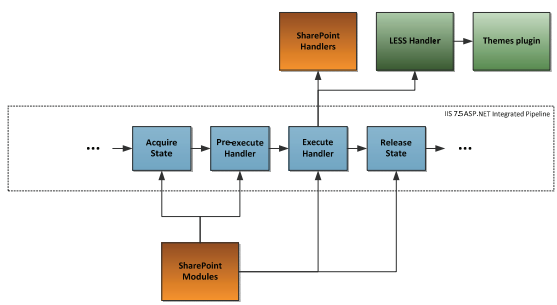
няма да може да се използва виртуализацията на SharePoint, тъй като обработчикът на SharePoint все още няма да е сработил. Друг възможен момент е събитието “Release State”, когато обработчиците вече са си свършили работа и освобождават заетите ресурси. Тогава обаче няма възможност да се използва изходния поток, който се изпраща обратно към браузера. Работата в „Execute Handler“ фазата не се препоръчва, тъй като тази част от pipeline-а отговаря за изпълнението на всички конфигурирани обработчици и към нея има специални изисквания.



Фиг. 2 – DotLess интерпретатор като HTTP модул.

Възможността да се контролира входа и изхода на LESS интерпретатора позволява да се присвоят стойности от темата, избрана в SharePoint на определени променливи от LESS файла. Освен това, методът притежава всички предимства на предходния. Като недостатък може да се изтъкне, че за да се промени входът или изходът на интерпретатора и да се инжектират съответните променливи, е необходимо да се работи със символни низове с голям размер. Това позволява генерираният CSS да бъде повреден при некоректна обработка от страна на модула. В допълнение, инжектирането на символни низове не е добра практика. Към това може да се добави, че модулът може да работи само с физически файлове, разположени по файловата система на сървъра. За да може да се работи и с виртуализирани файлове, модулът трябва да се прикачи към няколко събития от pipeline-а, което го прави сложен, по-труден за разработка, отстраняване на грешки и поддръжка.

Притурка за интеграция с машината за теми на SharePoint. Интерпретаторът DotLess има вградени възможности за разширяване на функционалностите чрез своята plugin архитектура. Интерпретаторът позволява да се напише притурка, която да се включи в процеса на интерпретация и да го промени. Това е най-подходящият начин за внедряване на SharePoint темите в LESS файлове.



Фиг. 3 – Използване на притурка за DotLess интерпретатора.

Притурката трябва да наследява абстрактния клас VisitorPlugin, който е част от библиотеките на DotLess интерпретатора. Всички притурки изпълняват шаблона за проектиране „Visitor" [11] и се извикват последователно върху всеки елемент от LESS файла, който е разпознат от интерпретатора. Така, когато притурката разпознае променлива с определено име, тя може да и присвои стойност от SharePoint темата.

ЗАКЛЮЧЕНИЕ

Като част от развойната дейност по системата за управление на онлайн съдържанието на Русенски университет бяха разгледани и изследвани различните методи за интеграция на LESS интерпретатор с SharePoint сървър. Беше идентифициран най-подходящия метод за интеграция - HTTP обработчик и притурка за DotLess, който позволява да се използват пълните функционалности както на SharePoint, така и на LESS езика.

ЛИТЕРАТУРА

[1] „CSS," [Онлайн]. Available: http://bg.wikipedia.org/wiki/CSS. [Отваряно на 16 Октомври 2012].

[2] A. Kennedy и I. León, „Dynamic CSS," в Pro CSS for High Traffic Websites, Apress, 2011.

[3] V. Chowdhary, „CSS Preprocessors: The future of CSS," 14 Юни 2011. [Онлайн]. Available: http://php.refulz.com/css-preprocessors-the-future-of-css/. [Отваряно на 16 Октомври 2012].

[4] G. Williams, „Laying the CSS3 Foundations," в Learn HTML5 and JavaScript for Android, Apress, 2012.

[5] „About Sass," [Онлайн]. Available: http://sass-lang.com/about.html. [Отваряно на 16 Октомври 2012].

[6] [Онлайн]. Available: http://lesscss.org/. [Отваряно на 16 Октомври 2012].

[7] [Онлайн]. Available: http://coffeescript.org/. [Отваряно на 16 Октомври 2012].

[8] Microsoft Corporation, [Онлайн]. Available: http://www.typescriptlang.org/. [Отваряно на 16 Октомври 2012].

[9] Twitter, [Онлайн]. Available: http://twitter.github.com/bootstrap/index.html. [Отваряно на 19 Октомври 2012].

[10] [Онлайн]. Available: http://www.dotlesscss.org/. [Отваряно на 19 Октомври 2012].

[11] Е. Гама, Р. Хелм, Р. Джонсън и Д. Влисидес, Шаблони за дизайн: Елементи на обектно-ориентирания софтуер за многократно използване, СофтПрес ООД, 2005, p. 331.

За контакти:

доц. д-р Мирослав Михайлов, директор на ЦИКО, Русенски университет „Ангел Кънчев", тел.: 082 888 782, e-mail: mmihaylov@uni-ruse.bg.

маг. инж. Цветелин Павлов, ЦИКО, Русенски университет „Ангел Кънчев", тел.: 082 888 328, e-mail: cpavlov@uni-ruse.bg

Публикацията е реализирана по проект по ФНИ 2012 РУ-07

Докладът е рецензиран.