OTS 3 Fachvortrag

XSLT 2.0 im Browser mit Saxon-CE

Dr. Thomas Meinike, Hochschule Merseburg

Motivation

XML-Technologien haben sich im Bereich der Technischen Dokumentation vor allem zur Erstellung von Medienprodukten etabliert. Bei clientseitigen Web-Anwendungen beschränkt sich die Nutzung im Wesentlichen auf XHTML (zunehmend durch HTML5 verdrängt) und bestenfalls zusätzliche Formate wie SVG und MathML.

XSLT [1] hat sich als mächtiges Werkzeug zur Transformation von XML-Inhalten nach HTML und XML oder in sonstige textorientierte Zielformate bewährt. Der unmittelbare Einsatz im Browserkontext wird eher in Ausnahmefällen praktiziert. Häufig bereiten unterschiedliche Implementierungen in Desktop- oder Mobil-Browsern Probleme oder werden alternative Datenzugriffsmethoden wie JSON favorisiert. Allerdings wird XSLT durchaus auf der Serverseite zur Produktion von HTML-Ausgaben eingesetzt, z. B. als PHP-Modul. Hier „regiert“ jedoch noch die bereits 1999 spezifizierte XSLT-Version 1.0, deren Möglichkeiten nicht alle Entwicklerwünsche abdecken.

Seit 2007 bietet XSLT 2.0 ein stark erweitertes Spektrum, insbesondere durch Verbesserungen wie Mehrfachausgaben, Gruppierung, strengere Datentypisierung, neue Operatoren und Deklaration eigener Funktionen sowie Erweiterungen der XPath-Funktionsbibliothek [2]. Insofern war es längst an der Zeit, diese Erweiterungen auch für die clientseitige Web-Entwicklung verfügbar zu machen. Da sich Browserhersteller bisher konsequent verweigern, hat das Saxonica-Team [3] um den Herausgeber der W3C-Spezifikation und Saxon-Prozessor-Entwickler Michael Kay bereits 2011 mit Saxon-CE einen eigenen innovativen Ansatz vorgestellt (CE = Client Edition). Die im Februar 2013 als Open-Source-Software veröffentlichte Version 1.1 und ihre praktische Nutzung ist Gegenstand der folgenden Ausführungen.

Voraussetzungen

Die benötigte Basissoftware lässt sich über [4] beziehen. Zur produktiven Nutzung dient das im Archiv Saxon-CE_1.1.zip enthaltene Verzeichnis Saxonce, das zusätzliche Verzeichnis SaxonceDebug entspricht diesem formal und enthält Testcode für die Entwicklungsphase. Die bei GitHub [5] gelagerte Fassung Saxon-CE-master.zip bietet zusätzlichen Beispielcode und die Entwicklerdokumentation. Letztere steht auch online zur Verfügung [6] und wurde selbst mit Saxon-CE umgesetzt. Einen Eindruck vermittelt Abbildung 1.

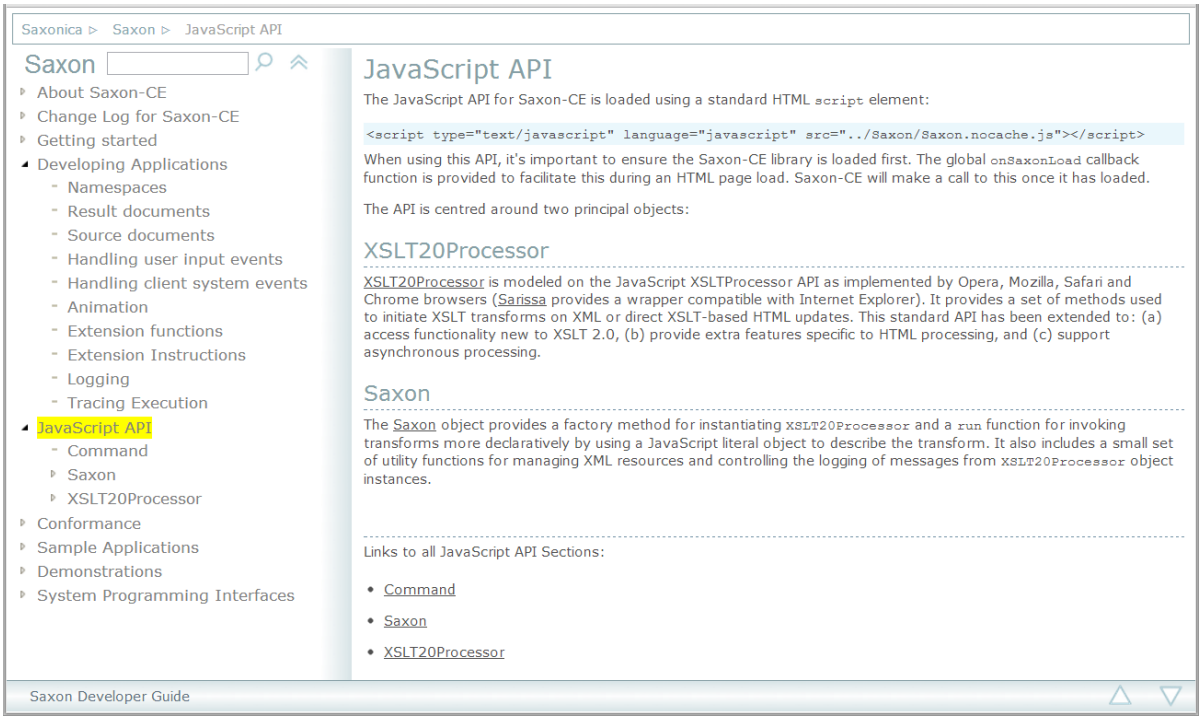


Abb. 1: Entwickler-Dokumentation zu Saxon-CE

Unter Windows steht mit XMLQuire [7] ein spezielles Werkzeug zur CE-Entwicklung bereit. Hier wird ein lokaler Web-Server benötigt und die Darstellung der Ergebnisse erfolgt durch den integrierten systemeigenen Internet Explorer. Der Zugriff auf alle Ressourcen gestaltet sich komfortabel unter einer Oberfläche. Ansonsten kann mit einem für den Umgang mit HTML und XML geeigneten Editor sowie einschlägigen Browsern direkt begonnen werden.

Entwicklung

Saxon-CE wurde mit dem Google Web Toolkit (GWT) aus den Java-Quellen des XSLT-Prozessors Saxon nach JavaScript „cross-kompiliert“. Details dazu liefern die Autoren in einem Balisage-Konferenzartikel [8]. Wesentlich für den Einsatz ist die Einbindung der im Verzeichnis Saxonce enthaltenen JavaScript-Ressource Saxonce.nocache.js im head-Element eines HTML-Dokuments. Dieses initiale Skript lädt browserspezifische *.cache.html-Dateien mit dem eigentlichen CE-Code nach.

Im HTML-Dokument wird auf externe Referenzen zum XSLT-Stylesheet für die Transformation und das (optionale) XML-Datendokument verwiesen. Dazu stehen mehrere Möglichkeiten bereit, besonders praktikabel ist die Verwendung eines weiteren script-Elements mit src- und data-source-Attributen. Da die Anwendungslogik komplett im XSLT-2.0-Dokument liegt, sind im body-Element nur wenige Vorbereitungen zu treffen. Für erste Tests reichen einige verschachtelte div-Elemente aus, die mit IDs versehen werden und später im XSLT-Code über #idname angesprochen werden und somit Ausgaben in Form von HTML-DOM-Fragmenten entstehen. CSS-Dateien für Formatierungen werden wie üblich mittels link-Element im HTML-Dokument eingebunden.

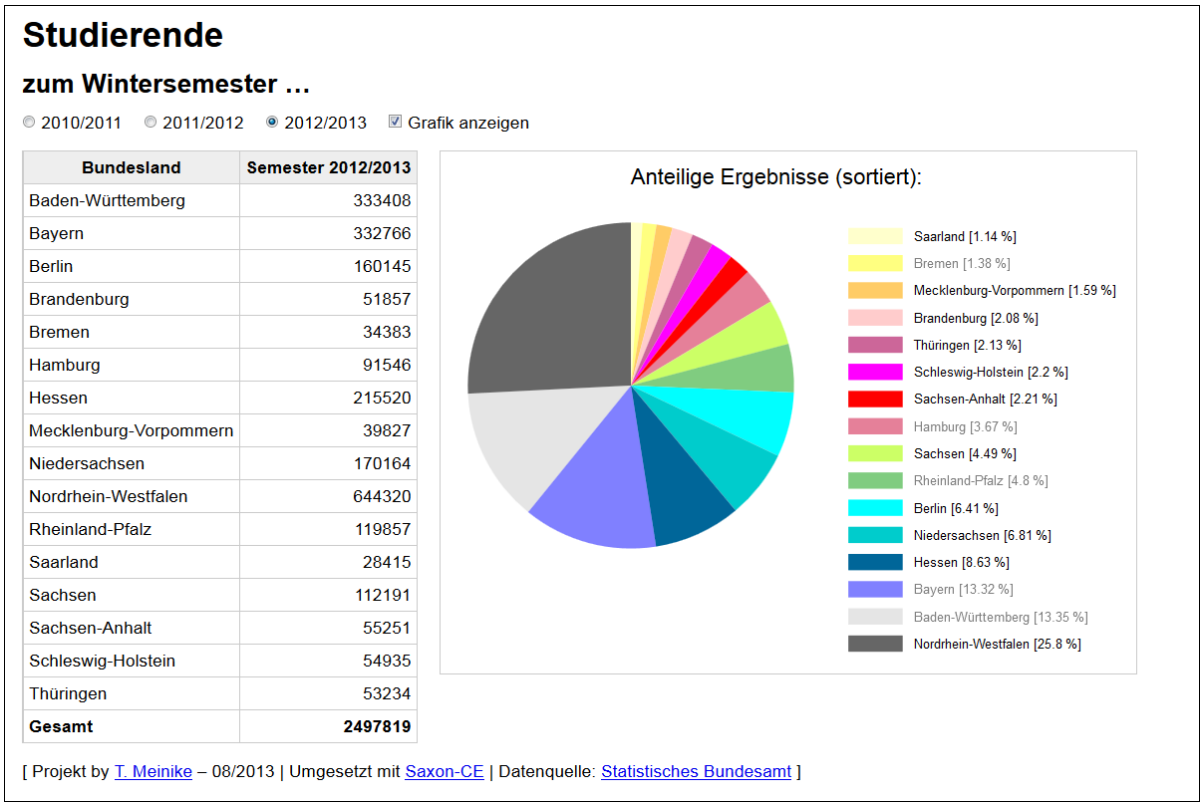
Zur konkreten Entwicklung sind fundierte Kenntnisse im Bereich XSLT und XPath 2.0 nötig. Entsprechend angepasste Templates ermöglichen die Verarbeitung des XML-Dokuments bzw. eine zielgerichtete Steuerung von Ausgaben. Die selektive Ausgabe der transformierten Daten erfolgt mit dem 2.0-Element xsl:result-document. Eigentlich zur Dateiausgabe bestimmt, wird über dessen

href-Attribut auf die genannten #idname-Referenzen in der HTML-Basisstruktur verwiesen. Dabei erlaubt das Attribut method das Anhängen von Inhalten (ixsl:append-content) oder ihre vollständige Ersetzung (ixsl:replace-content).

Der Namensraum ixsl dient ebenfalls zur Nachrüstung weiterer Interaktionsmöglichkeiten. Ruft man ein Template z. B. mit der Angabe mode="ixsl:onclick" auf, wird das zugehörige Nutzerereignis, etwa beim Anklicken eines Buttons, getriggert und die gewünschte DOM-Manipulation ausgeführt. Weitere Namensräume sind style und prop, die für den Zugriff auf Stylesheet-Eigenschaften und JavaScript-Objekteigenschaften bestimmt sind. Obwohl die Entwicklungsarbeit grundsätzlich keine vertieften JavaScript-Kenntnisse voraussetzt, sind diese dennoch nützlich. So lassen sich in Analogie zu vordefinierten XPath-Funktionen mittels js:myfunction(…) aus dem XSLT-Kontext heraus eigene JS-Funktionen aufrufen und somit auch bereits vorhandene, etwa mathematische Funktionen über das Math-Objekt, einbinden.

Darüber hinaus stellt Saxon-CE Erweiterungsfunktionen für den gemeinsamen Zugriff auf XML-Eingabe- und HTML-Ausgabedaten bereit und erlaubt die Interaktion mit JavaScript-eigenen Objekten wie window sowie den Zugriff auf im HTML-DOM präsente Attribute. Ein Logging-Mechanismus bringt entsprechende Ausgaben in den Konsolenbereichen der Browser unter.

Das vom Autor umgesetzte Beispielprojekt [9] stellt statistische Daten in HTML-Tabellenform und als SVG-Kreisdiagramm dar. Dabei lassen sich ausgehend vom nach Jahren organisierten Datenbestand mehrere Teilansichten aktivieren, erzeugte Tabellenspalten per Mausklick auf die Überschrift sortieren und die Vektorgrafik kann durch Mausbewegung erkundet werden, siehe Abbildung 2.



Bundesland	Semester 2012/2013
Baden-Württemberg	333408
Bayern	332766
Berlin	160145
Brandenburg	51857
Bremen	34383
Hamburg	91546
Hessen	215520
Mecklenburg-Vorpommern	39827
Niedersachsen	170164
Nordrhein-Westfalen	644320
Rheinland-Pfalz	119857
Saarland	28415
Sachsen	112191
Sachsen-Anhalt	55251
Schleswig-Holstein	54935
Thüringen	53234
Gesamt	**2497819**

Abb. 2: Mit Saxon-CE umgesetztes Beispielprojekt

Ausblick

Saxon-CE erweist sich als interessante Alternative zur Entwicklung von interaktiven Browser-Anwendungen. Insbesondere datengetriebene Dokumentationsprozesse und webbasierte Onlinehilfen, wie die CE-Dokumentation, können von diesem Konzept profitieren. Im Vortrag werden die genannten und weiteren Techniken mit Quellcode untersetzt und demonstriert.

Literaturangaben und Links

[1] W3C: XSL Transformations (XSLT); http://www.w3.org/TR/xslt

[2] Meinike, T.: XSLT 2.0 und XPath 2.0 für Praktiker – Neuerungen im Überblick. In: tekom, Gesellschaft für technische Kommunikation e. V.; Tagungsband zur Jahrestagung 2007 in Wiesbaden, S. 212–215 (Folien: http://www.tekom.de/upload/2284/INF_114_Meinike_Vortrag.pdf)

[3] Saxonica: XSLT and XQuery Processing; http://saxonica.com/

[4] Saxon-CE: http://saxonica.com/ce/index.xml

[5] GitHub: https://github.com/Saxonica/Saxon-CE

[6] CE-Dokumentation: http://saxonica.com/ce/user-doc/1.1/

[7] XMLQuire Web Edition – A Free XSLT 2.0 Editor for the Web; http://qutoric.com/xmlquire/ce/

[8] Delpratt, O. and Kay, M.: Interactive XSLT in the browser – Balisage: The Markup Conference 2013; http://balisage.net/Proceedings/vol10/html/Delpratt01/BalisageVol10-Delpratt01.html

[9] Meinike, T.: Studierende in Deutschland; http://datenverdrahten.de/xslt2/saxon-ce/studis/

für Rückfragen:
thomas.meinike@hs-merseburg.de