

Urs SIEBER, sieberu@stvsteinach.ch
Tobias REBER, toby@seidenfein.ch

1 Newsprojekt Etappe 3 mit SVG

Das Ziel dieser Etappe ist die Transformation des XML-Files mit JDOM in ein SVG-File. Die Daten im XML-File müssen für den Benutzer grafisch aufgearbeitet sein, damit er schnell einen Überblick über die wichtigsten statistischen Grunddaten erhalten kann.

Zur Etappe 3 ist zu erwähnen, dass wir anfangs mit der Aufgabenstellung ziemlich ratlos waren. Wir kannten bis dahin weder SVG, noch JDOM und wir hatten noch nie mit Java programmiert. Für uns zwei Geographiestudenten war jedoch klar, dass wir eine dynamische Karte mit Statistik erzeugen wollen.

1.1 Die generierten Statistiken

Zuerst wurden anhand des News-Schemas die vier verschiedenen Statistiken ausgewählt. Durch die Aufgabenstellung war die Statistik, die eine geografische Verteilung der Informationen zeigt, bereits gegeben. Die anderen Statistiken wählten wir aus, indem wir uns überlegten, welche Daten für einen Benutzer interessant sind.

Zuerst erstellten wir eine SVG-Vorlage. Dazu wurde die auf dem Internet gefundene Open-Source Applikation Inkscape (<http://inkscape.org>), mit welcher man mit wenig Aufwand eine SVG-Datei zeichnen kann, verwendet. Mit dem Programm erstellten wir das Layout unseres SVG-Dokumentes. Anschliessend wurde der SVG-Code gesäubert und ergänzt, was sich als grosser Vorteil für die spätere Entwicklung mit JDOM erwies. Auch wurde der Code gut kommentiert, so dass die Übersicht im Code gewährleistet blieb.

Der nächste Schritt war das Einrichten der Java-Entwicklungsumgebung Eclipse und erste Gehversuche mit JDOM. Um einen besseren Überblick zu behalten, haben wir zu den verschiedenen graphischen Darstellungen eigene Klassen erstellt, welche mit der Methode `getElement` ausgestattet sind, um eben diese graphischen Elemente zu erstellen:

Für die Geschlechterverteilung `GenderBuilder.java`: Die Methode `getGender` nimmt als Parameter die Anzahl weiblicher und die Anzahl männlicher User in Prozent. Beim Aufrufen der Methode wird die Statistik über die Geschlechterverteilung zurückgegeben.

Für die Medienvolumenverteilung `MediaBuilder.java`: Die Methode `getMedia` nimmt als Parameter die vier Prozentwerte zu Internet, Radio, Fernsehen und Zeitung (Flieskommazahl < 1).

Für die Karte: `MapBuilder.java`: Die Methode `getMap` dieser Klasse ist parameterlos und gibt lediglich ein Element mit der Karte und der Legende zurück.

In Zusammenspiel mit der Karte `CircleBuilder.java` und `LineBuilder.java`:

Diese Klassen zeichnen auf der Karte die Linien und Kreise, welche mit den Methoden `getCircle(X-Koordinate, Y-Koordinate, Anzahl News)` und `getLine(x1, y1, x2, y2)` verlangt werden können. Je mehr Informationen mit derselben Herkunft, desto grösser der Radius des Kreises. Bei mehr als 10 Informationen wird der Radius jedoch nicht weiter vergrössert, aber der Kreis wird animiert dargestellt. Die Linien zu den related Infos werden auch animiert dargestellt.

1.2 Auswahl der Daten mit JDOM

Die Auswahl der nötigen Parameter für die oben erwähnten Statistiken geschieht in der Datei `SVGConstructor.java`. In diesem File werden im ersten Teil alle Daten aus dem XML-File gelesen und in Variablen gespeichert. Im unteren Abschnitt werden die diversen Builder-Objekte kreiert, um die Daten im SVG-Ausgabedokument sichtbar zu machen. Das Vorgehen zur Datensammlung verläuft immer im gleichen Schema: Es wird eine `ArrayList` generiert, in die die mittels X-Path abgefragten Daten Elementweise gespeichert werden. Mit der richtigen Auswertung der `ArrayList` erhält man danach die korrekten Parameter.

Ein spezielles Vorgehen musste gewählt werden für die Daten der `relatedInfos`, damit die Verbindungen der `related Infos` erstellt werden konnten. Es wurden zuerst in eine länderspezifische `ArrayList` die Herkunftsländer der `related Infos` gespeichert. Die Abfrage wurde über die `Info-ID` generiert. Als Beispiel die Abfrage der Herkunftsländer der `related Infos` für Schweizer Infos:

```
ArrayList arrCH = (ArrayList)XPath.newInstance
("//infos/info[infoID = //infos/info[infoOrigin='Suisse']
/relatedInfos/info/@infoID]/infoOrigin").selectNodes(root);
```

Danach wurde mit jeder länderspezifischen `ArrayList` eine `for`-Schleife durchlaufen, die die Linien zu den in den `ListElementen` enthaltenen Ländern zeichnet.

1.3 Persönliche Erfahrungen mit SVG / JDOM

Für die spätere Entwicklung mit JDOM, erwies sich die Erstellung einer SVG Vorlage sehr wichtig. Man braucht zuerst ein mögliches Resultat und die Vorstellung davon. Danach muss man mit JDOM diesen in der Vorlage erstellten Code immer genauer abbilden, bis am Schluss das korrekte Resultat erzielt wird. Als Nachteil von SVG gilt sicher die unkonventionelle Art der Implementierung von grafischen Objekten. Ein `Path-Element` wird mit unzähligen Attributen und Parametern ausgestattet, bis es so aussieht, wie man gerne möchte. Hätten wir den Rat nicht befolgt und anfangs keine Vorlage eines SVG-Dokumentes erstellt, wären wir bei diesem Projekt wohl auf der Strecke geblieben.

Positiv überrascht waren wir, wie man in SVG mit sehr wenig Aufwand die Elemente anhand ihrer Attribute animieren kann. Die Effekte sind nützlich und einfach zu erstellen.

Nach einiger Übung waren wir erstaunt, wie praktisch man mit JDOM Daten aus dem XML-Dokument herauslesen und daraus ein SVG-Dokument erzeugen kann. Am Schluss würden wir sogar behaupten, dass eine Transformation mit JDOM logischer und übersichtlicher ist, als eine umständliche Transformation mit XSL-T.

Nützlich war die Einführung eines `Debug-Bits`, das je nach Wert nützliche Ausgaben bei der Transformation darstellt. Für das Debugging hat sich dies als wertvoll erwiesen.

2 Vorhandene Dateien

Folgende Dateien sind von Reber und Sieber für Etappe 3 vorhanden:

`News_sieber_reber.xml`: XML-File mit allen Daten von User und Infos.

`europa.jpg`: Europakarte, die für den geografischen Statistikblock benötigt wird.

`GEoNews_sieber_reber.svg`: SVG-File mit den Statistiken, das automatisch generiert wird.

`*.java`: Die diversen benötigten Klassen sind als `.java Files` gespeichert.