

Tutorial: Transformationen entwickeln

Version: 1
Autor: Daniel Neumann

Inhaltsverzeichnis

| | | |
|-----|---|----|
| 1 | Anlegen von Transformationen für das "Zitat" | 2 |
| 2 | Literale Methoden..... | 3 |
| 2.1 | Ausgeben von Objektdaten..... | 6 |
| 3 | Binäre Methoden..... | 8 |
| 4 | Anlegen von Transformationen für die Zitatsammlung..... | 10 |
| 4.1 | Anzahl der Zitate..... | 11 |
| 4.2 | Aktuelle Uhrzeit..... | 12 |
| 4.3 | Berechnung des Indexes..... | 12 |
| 4.4 | Ausgabe des jeweiligen Zitats..... | 13 |

Voraussetzungen

Um dieses Tutorial durchzuarbeiten, sollten Sie über **Grundkenntnisse von XSLT** verfügen. Möchten Sie zu diesen Themen zunächst ein Tutorial durcharbeiten, empfehlen wir Ihnen an dieser Stelle das von w3schools.org.

Darüber hinaus muss das **Modul "XSL Renderengine"** im onion.net Editor bereits vorhanden sein.

- > [XSLT Tutorial](#)
- > [Informationsmodell festlegen](#)

Beschreibung

In der zweiten Folge der Reihe wird gezeigt, wie schnell und unkompliziert sich die erfassten Daten über XSL-Transformationen verwerten lassen.

Hinweis: Die Verarbeitungslogik wird im Datenbereich des onion.net Editors erstellt. Eine Transformation wird immer einem Schema zugeordnet. Sie versteht sich dabei als objektorientierte Methode, die – wie in der Programmierung üblich – durch Ableitungen erweitert und Methodensignaturen parametrisiert werden kann.

Übrigens: Die Verarbeitungslogik wird in onion.net als Information erfasst. Dies ist eines von zahlreichen Beispielen, in der onion.net uns bei neuen Lösungen und Erweiterungen unterstützt hat.

Zeichenerklärung



Text, der grün umrandet und mit dem Pfeil-Symbol gekennzeichnet ist, enthält konkrete Anweisungen, was als nächstes zu tun ist.



Texte in solchen Kästchen enthalten Tipps und Tricks.

Quellcode wird in solch blauen Boxen dargestellt.

1 Anlegen von Transformationen für das "Zitat"

Als erstes erstellen wir eine Transformationsgruppe. Dort sollen alle Zitat-Transformationen Platz finden.



Dazu öffnen Sie in der Baumstruktur der Detailansicht den Ordner „Transformations“ und klicken mit rechts auf das Objekt „Onion“. Im Kontextmenü wählen Sie nun „Transformationsgruppe“.



Im rechten Bereich des Editors sehen Sie nun die Detailansicht der Transformationsgruppe.



Vergeben Sie als Titel „Zitatverwaltung“.

Als nächstes definieren wir, für welchen Datentyp die Transformation gelten soll.



Dazu klicken Sie mit rechts auf die eben erstellte Transformationsgruppe „Zitatverwaltung“ und wählen unter „Neu“ den Eintrag „Datentyp“ aus. Als Titel wählen Sie hier „quotation“, da die Transformation für ein Zitat gelten soll.

Allein durch den Titel wird noch keine Verbindung zu dem Schema hergestellt.



Klicken Sie auf „name“, um das Feld anzuzeigen. In dieses Feld muss nun das Schema eingefügt werden.



Tipp: Um Drag & Drop zwischen den beiden Editorfenstern zu nutzen, ziehen Sie das Objekt einfach über das Symbol z.B. von der Datenansicht, ohne die Maustaste loszulassen. Die Ansicht wechselt nun automatisch auf die neue Ansicht. Nun können Sie das Objekt an der Stelle der neuen Ansicht fallenlassen, an der es eingefügt werden soll.



Wechseln Sie in die Schemaverwaltung und ziehen mittels Drag & Drop das Schema „quotations/quotation“ in das Feld „name“ des eben angelegten Datentyps. Geben Sie dieses anschließend zurück.

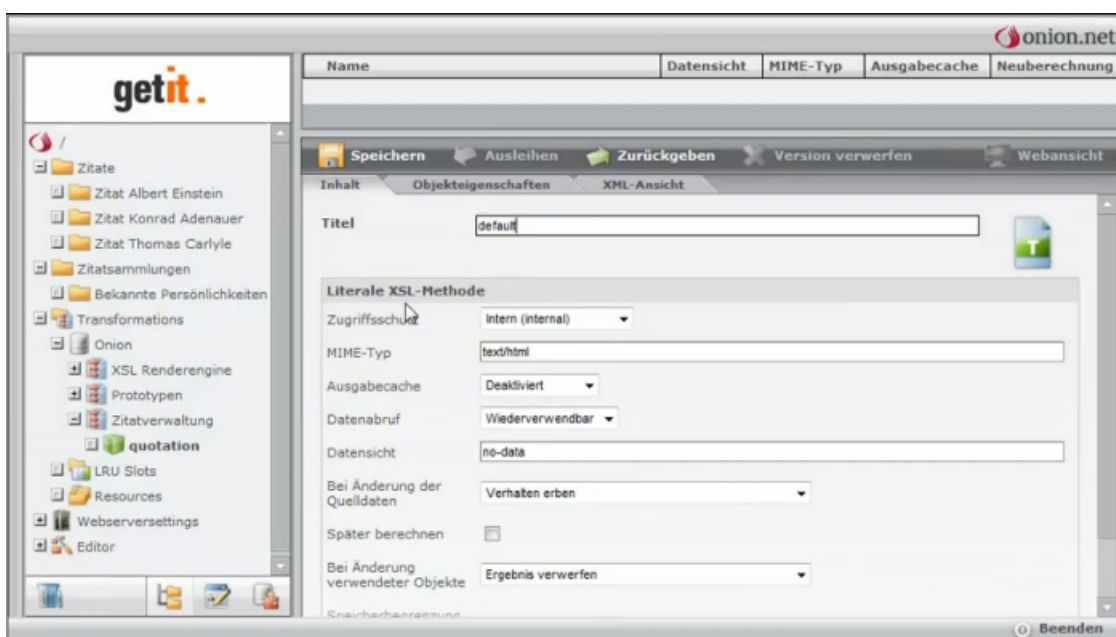
Der Transformation ist nun ein XML-Schema eindeutig zugeordnet.

2 Literale Methoden

Wir erzeugen nun eine literale Methode für unseren Datentyp. Diese sind in der Lage, textbasierte Ausgaben zu erzeugen.



Klicken Sie dazu mit rechts auf den Datentyp „quotation“ und erstellen eine neue Literale Methode aus dem Kontextmenü. Geben Sie dieser Methode den Namen „default“.



„default“ gilt als Standardmethode für das Transformationssystem. Das bedeutet, dass diese Methode aufgerufen wird, wenn ein Objekt von diesem Datentyp angezeigt werden soll.

Unter dem Titel können einige Einstellungen für diese Methode vorgenommen werden. Wir müssen in diesem Fall den Zugriffsschutz ändern. Der Zugriffsschutz steuert, von wo aus die Methode aufgerufen werden darf.

internal von anderen Transformation

protected von .NET Code

public über öffentliche Schnittstellen



Für die default-Methode wählen wir „Öffentlich (public)“ aus.

Ganz unten in der Detailansicht sehen Sie ein Eingabefeld. In dieses wird nun die eigentliche Transformation geschrieben.



Schreiben Sie dort „<xsl:stylesheet“ und drücken Sie STRG+Leertaste.

Über die Tastenkombination wird ein Assistent zur Vervollständigung der XSL Transformation geöffnet.

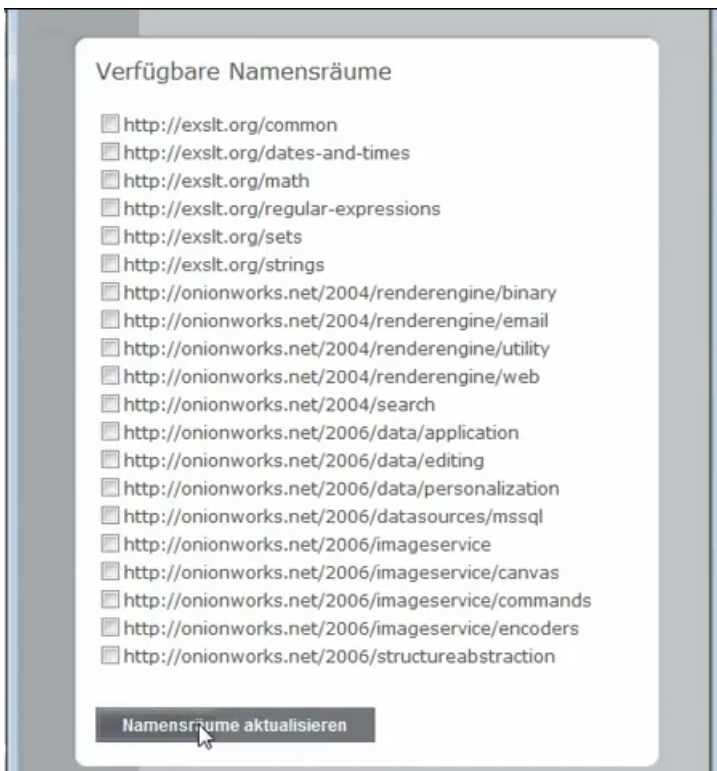


Möglicherweise wird der Assistent durch den Popup-Blocker des Browsers unterdrückt. In diesem Fall stellen Sie Ihren Browser so ein, dass Popups zugelassen werden

XSLT ist eine erweiterbare Transformationssprache. Hier finden sich die standardisierten EXSLT-Erweiterungen sowie nützliche onion.net-Erweiterungen, die z.B. Methoden zur Bildmanipulation oder PDF-Generierung enthalten.



Klicken Sie in dem Assistenten auf „Namensräume aktualisieren“, ohne einen Wert auszuwählen.



Es wird folgende Standard-Transformation angelegt:

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:onion="http://onionworks.net/2004/data"
  version="1.0"
>
  <xsl:output
    method=" "
    omit-xml-declaration="yes"
    indent="no"
  />
  <xsl:template match="/"></xsl:template>
</xsl:stylesheet>
```



Tragen Sie im `<xsl:output>`-Element „xml“ in das „method“-Attribut ein. Innerhalb des `<xsl:template>`-Elements geben Sie die gewünschte Ausgabe ein. Wir begnügen uns zunächst mit einem „Hallo Welt!“. Speichern Sie nun die Transformation.

Komplett sieht die Transformation nun wie folgt aus:

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:onion="http://onionworks.net/2004/data"
  version="1.0"
>
  <xsl:output
    method="xml"
    omit-xml-declaration="yes"
    indent="no"
  />
  <xsl:template match="/"> Hallo Welt! </xsl:template>
</xsl:stylesheet>
```



Um das Template direkt zu testen, klicken Sie mit rechts auf ein beliebiges Zitat, z.B. „Zitat Albert Einstein“.

Da der Editor nun eine „default“-Methode für unser Zitat gefunden hat, bietet er uns die Funktion „Vorschau“ im Kontextmenü an.

Wenn Sie dieses anklicken, öffnet sich ein Browserfenster mit dem Text „Hallo Welt!“, genau wie in der Transformation angegeben.

2.1 Ausgeben von Objektdaten

Nun wird für jedes Objekt vom Typ „quotation“, also jedes Zitat, der Text „Hallo Welt!“ ausgegeben. Jetzt ersetzen wir diesen Text durch tatsächliche Werte aus den jeweiligen Zitaten. Im ersten Schritt soll unsere Transformation den Autor ausgeben.



Wechseln Sie zurück in die literale Methode default des Datentypen „quotation“.



Tipp: anders als bei allen Objekten zuvor erscheint die literale Methode nicht als Kindelement in der Baumstruktur auf der linken Seite, sondern befindet sich in der Listenansicht oben rechts, wenn Sie den entsprechenden Datentypen auswählen. Dies wird aus Gründen der Übersichtlichkeit gemacht, da in der Listendarstellung auf einen Blick noch weitere Informationen über die Methode angezeigt werden.

Jedes Datenobjekt verfügt über verschiedene Datensichten, die jeweils einen Aspekt des Objekts repräsentieren. Diese können Sie in dem Feld „Datensicht“ ändern. Standardmäßig ist „no-data“ vorgegeben. In diesem Fall beziehen wir unsere Informationen aus der Datensicht „content“, da wir auf Inhalte aus dem Content des Objekts zugreifen wollen.



Tragen Sie unter Datensicht den Wert „content“ ein.

Wir verarbeiten nun das Zitat gemäß unseres XML-Schemas. Die Transformation soll auf den Wurzelknoten „quotation“ reagieren und den Textinhalt des Elements „author“ ausgeben.



Tipp: Wenn Sie sich bisher noch nicht mit XPath auseinandergesetzt haben, ist nun ein guter Zeitpunkt, sich ein passendes Tutorial dazu durchzulesen. Ein gutes, allerdings englischsprachiges, Tutorial finden Sie auf den Seiten von w3schools.org: XPath Tutorial (<http://www.w3schools.com/xpath/>)



Ändern Sie im `<xsl:template>`-Element das Attribut „match“ mit dem XPath-Ausdruck, der das Wurzelement „quotation“ selektiert.

```
<xsl:template match="/quotation"></xsl:template>
```

Nun soll von quotation das Feld mit dem Namen „author“ ausgegeben werden.



Tipp: Nun werden einige XSLT-Elemente eingeführt. Eine genauere Beschreibung der einzelnen Elemente gibt es z.B. auch bei w3schools: XSLT Elements Reference (http://www.w3schools.com/xsl/xsl_w3celementref.asp)

Verwenden Sie dazu das „value-of“-Element wie folgt:

```
<xsl:value-of select="author" />
```

Insgesamt sieht der Template-Block in der Methode dann wie folgt aus:

```
<xsl:template match="/quotation">
  <xsl:value-of select="author" />
</xsl:template>
```



Speichern Sie nun die Methode und rufen die Vorschau eines Zitates auf.

Nun wird statt „Hallo Welt!“ der Autor des jeweiligen Zitates ausgegeben.

Im nächsten Schritt geben wir das Zitat aus. Das Zitat ist ein formatierbarer Text auf Basis von XHTML. Diesen können wir samt Formatierungsanweisung in die Ausgabe kopieren.



Ergänzen Sie das Template in der literalen Methode um folgende Zeilen, die Sie nach dem <value-of>-Element einfügen:

```
sagte einmal:  
<xsl:copy-of select="quote/node()" />
```

Wenn Sie nun die Vorschau öffnen, sehen Sie zusätzlich zur Ausgabe des Autoren auch den Text „sagte einmal:“ mit anschließendem Zitat mit Formatierungen.

3 Binäre Methoden

Als nächstes wollen wir das Bild anzeigen. Das System bietet zu diesem Zweck binäre Methoden.



Legen Sie nun unterhalb von „quotation“ eine neue binäre Methode an. Diese bekommt den Titel „picture“. Unter Zugriffsschutz wählen Sie „Öffentlich (public)“ und als Datensicht benötigen wir den „content“.

Die Methode, die wir nun anlegen, gibt genau angepasst für das Zitat das Bild aus und ist keine Methode, die generell für die Bildausgabe genutzt werden kann.

```
<xsl:stylesheet
  xmlns:b="http://onionworks.net/2004/renderengine/binary"
  xmlns:onion="http://onionworks.net/2004/data"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0"
>
  <xsl:output
    method="xml"
    omit-xml-declaration="yes"
    indent="no"
  />
  <xsl:template match="/quotation">
    <xsl:value-of select="b.write(image)" />
    <b:output mimeType="{image/@onion:mimeType}">
      <b:webResponse expires="60" />
    </b:output>
  </xsl:template>
</xsl:stylesheet>
```



Geben Sie anschließend die Methode zurück.

Nun steht Ihnen die binäre Methode „picture“ auch in Ihrem Datentyp zur Verfügung.

Wir generieren nun ein HTML img-Tag, wenn ein Bild gepflegt wurde. Für Referenzierungen unserer binären Methode nutzen wir die Funktion „c.binaryUri()“.



Tipp: Die Funktion c.binaryUri() ist wie weitere Funktionen in der Referenz genauer erklärt. Diese können Sie hier öffnen: [c.binaryUri](#)



Gehen Sie nun zurück in die literale Methode „default“ und ergänzen Sie das Template nach dem <xsl:copy-of>-Element um folgenden Eintrag:

```
<xsl:if test="image">
  
</xsl:if>
```

Nachdem Sie diese Änderung gespeichert haben und erneut die Vorschau öffnen, sehen Sie unter dem Zitat das in dem Zitat gepflegte Bild.

4 Anlegen von Transformationen für die Zitatsammlung

Als nächstes erstellen wir für unsere Zitatsammlung eine default-Methode.



Erstellen Sie dazu unterhalb der Transformationsgruppe „Zitatverwaltung“ einen neuen Datentyp mit dem Titel „collection“.

Klicken Sie auf den Datentyp „name“ und ziehen Sie das Schema collections/collection wie oben erläutert aus der Schemaverwaltung in das Feld. Geben Sie dann das Objekt zurück.

Erstellen Sie nun eine literale Methode mit dem Namen „default“ für den neuen Datentyp „collection“, indem Sie rechts auf das Objekt klicken.

Diese literale Methode bekommt, genau wie die andere Methode auch, den Zugriffsschutz „Öffentlich (public)“. Als Datensicht benötigen wir auch den „content“ des Objektes.



Füllen Sie die Felder „Zugriffsschutz“ und „Datensicht“ dementsprechend aus.

Als Inhalt der Transformation nehmen wir zunächst den Standardinhalt ohne Änderungen.

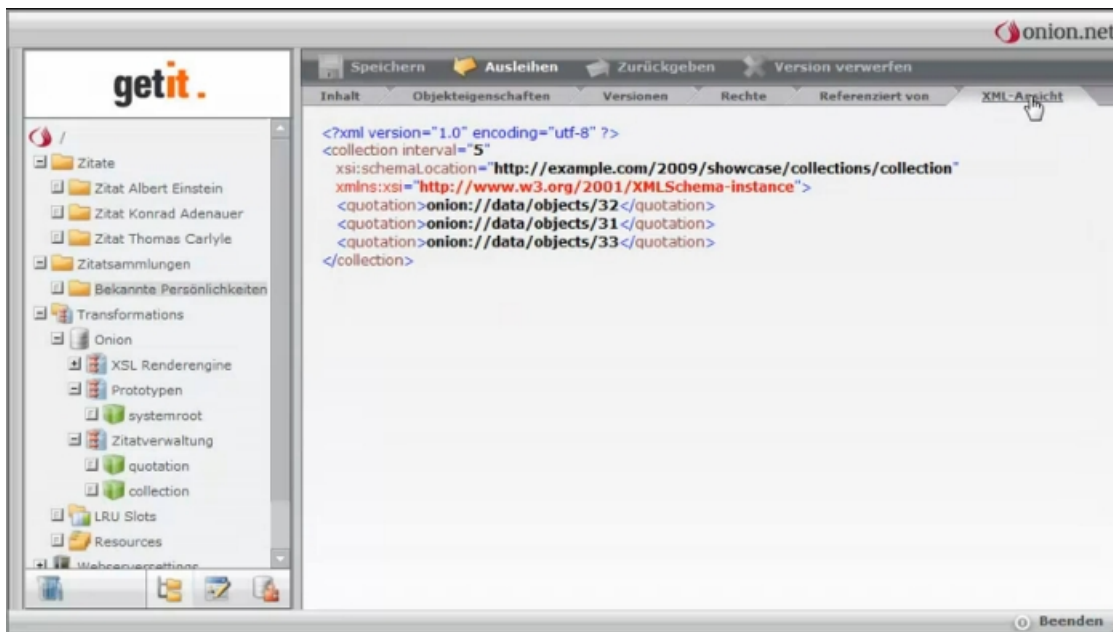


Tippen Sie in das unterste Eingabefeld „<xsl:stylesheet“ und vervollständigen Sie mit STRG+Leertaste den Standard-Inhalt. Ändern Sie lediglich in <xsl:output> das Attribut „method“ auf „xml“. Speichern Sie anschließend die Methode.

Schauen wir uns zur Erinnerung die Zitatsammlung an.



Klicken Sie unterhalb der Ablage „Zitatsammlungen“ auf die Sammlung „Bekannte Persönlichkeiten“. Rechts im Editor geht die Detailansicht auf. Wechseln Sie vom aktuell aktiven Reiter „Inhalt“ auf den Reiter „XML-Ansicht“ ganz rechts.



Unsere neue Methode soll jedes Zitat innerhalb des angegebenen Zeitraums (interval) in Sekunden ausliefern. Das Attribut „interval“ können Sie in dieser XML-Ansicht direkt an dem Collection-Element sehen.



Öffnen Sie nun wieder die literale Methode der collection. Genau wie bei der quotation ändern wir das Template-Element so, dass es das Wurzelement „collection“ selektiert.

```
<xsl:template match="/collection"></xsl:template>
```

4.1 Anzahl der Zitate

Wir geben erst einmal aus, wie viele Zitate sich in der Sammlung befinden. Dazu nutzen wir die Funktion „count()“.



Übernehmen Sie Folgendes als Inhalt des Templates:

```
Anzahl der Zitate:  
<xsl:value-of select="count(quotation)" />  
<br/>
```

Werfen Sie nun einen Blick auf die Vorschau der Zitatsammlung „Bekannte Persönlichkeiten“. Ausgegeben wird nun der Text „Anzahl der Zitate: 3“.

4.2 Aktuelle Uhrzeit

Darunter geben wir nun die aktuelle Uhrzeit aus.



Dazu tragen Sie folgende weitere Zeile in den Inhalt des Templates ein: Aktuelle Uhrzeit: `<xsl:value-of select="dt:time()" />
`

```
Aktuelle Uhrzeit:  
<xsl:value-of select="dt:time()" />  
<br/>
```

Da wir eine Datumsfunktion verwenden, benötigen wir eine Erweiterung des Namensraums. Dies geschieht einfach über die Codevervollständigung.



Platzieren Sie den Cursor direkt hinter „`<xsl:stylesheet`“ und drücken Sie STRG+Leertaste. In dem nun aufgehenden Assistenten „Verfügbare Namensräume“ wählen Sie „`http://exslt.org/dates-and-times`“ aus und klicken auf „Namensräume aktualisieren“. Der neue Namensraum wird nun zur Transformation hinzugefügt. Speichern Sie die Transformation.

In der Vorschau können Sie erkennen, dass nun auch die aktuelle Uhrzeit ausgegeben wird. Drücken Sie den Refresh-Button des Browsers, können Sie sehen, dass sich die Uhrzeit jedes mal ändert.

4.3 Berechnung des Indexes

Als nächstes errechnen wir, welches Zitat zum aktuellen Zeitpunkt angezeigt werden muss. Im ersten Schritt wechseln wir im Sekundentakt.



Erweitern Sie das Template wie folgt:

```
<xsl:variable name="index" select="floor(dt:seconds() mod count(quotation))  
+1" />
```

Zur Kontrolle geben wir den Index anschließend aus:

```
Aktueller Index:  
<xsl:value-of select="$index" />  
<br/>
```

In der Vorschau können wir nun sehen, dass beim Druck auf den Browser-Button Refresh zu jeder Sekunde ein anderer Index angezeigt wird.

Nun fügen wir den gepflegten Interval der Sammlung in die Formel ein.



Dazu ändern Sie das Attribut „select“ beim Setzen der Variable „index“ wie folgt:

```
floor(dt:seconds() div @interval mod count(quotation)) +1
```

Die Kontrolle in der Vorschau zeigt, dass sich der Index alle fünf Sekunden ändert.

4.4 Ausgabe des jeweiligen Zitats

Zuletzt rufen wir unsere default-Methode des x.ten Zitats auf. Dies erledigt der Befehl `c.literalCall`.



Tipp: auch dieser Befehl ist in der Referenz näher erläutert: `c.literalCall`



Erweitern Sie das Template um den Funktionsaufruf:

```
<c.literalCall id="{quotation[$index]}" />
```

Die komplette default-Methode sieht nun wie folgt aus:

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dt="http://exslt.org/dates-and-times"
  xmlns:onion="http://onionworks.net/2004/data"
  version="1.0"
>
  <xsl:output
    method="xml"
    omit-xml-declaration="yes"
    indent="no"
  />
  <xsl:template match="/collection"> Anzahl der Zitate:
    <xsl:value-of select="count(quotation)" />
    <br/>
    Aktuelle Uhrzeit:
    <xsl:value-of select="dt:time()" />
    <br/>
    <xsl:variable name="index" select="floor(dt:seconds() div @interval mod
count(quotation)) +1" />
    Aktueller Index:
    <xsl:value-of select="$index" />
    <br/>
    <c.literalCall id="{quotation[$index]}" />
  </xsl:template>
</xsl:stylesheet>
```

Geschafft!

Im nächsten Teil werden wir den Editor individualisieren.