SQF

Schematron QuickFix im Oxygen XML Editor

Vergleich zwei verschiedener Implementierungen

Donnerstag, 8. März 2018

Nico Kutscherauer

Agenda

SQF

- Idee und Entstehung von Schematron QuickFix
- Konzepte der Sprache
- Build-In-Implementierung des Oxygen
- Das Escali-Projekt
- Das Escali Oxygen Plugin
- Vergleich der beiden Implementierungen anhand von Beispielen
- SQF in der Praxis
- Ausblick

Schematron QuickFix

SQF

- „Quick fix“
  - eine automatische Behebung eines spezifischen Fehlers.
  - Korrekturvorschläge wie bei der Rechtschreibprüfung
- Schematron QuickFix
  - Erweiterungssprache für Schematron
  - Definiert QuickFixes für Schematron-Fehler
  - Standardisiert mit einer W3C-Note:
    - https://www.w3.org/community/quickfix/
- Beispiele:
  - Ersetze eine fehlerhafte ID durch eine aus dem Kontext heraus kalkulierte ID
  - Ersetze fehlerhafte Zeichen
  - Konvertiere fehlerhaftes Datumsformat in `xs:date`

Entstehungsgeschichte

SQF

- 2009-2011: Buchprojekt Schematron
- 2012: Erste Prototypen (Vorgänger vom Escali)
- 2013: George Bina stellt eine ähnliche Idee auf dem Oxygen User Meetup in München in Aussicht
- 2014
  - Treffen auf der XML Prague, Beschluss einer Kooperation
  - Gründung einer w3c-Gruppe zum Thema Quick-fixes
  - Schematron-quickfix.com geht online inklusive einer Commandline Implementierung von SQF.
  - Vorstellung von Schematron QuickFix auf dem Markupforum inklusive eines Prototypen des Escali Plugins

Entstehungsgeschichte

SQF

- 2015
  - Präsentation beim Oxygen User Meetup am PreConfDay der XML Prague zusammen mit Octavian Nadolu
  - Veröffentlichung eines FirstDrafts einer w3c-Note
  - Oxygen 17.0 enthält erste Implementierung
- 2016
  - Vorstellung des Projektes auf der XML Prague (Hauptkonferenz) mit Octavian
- 2017
  - Escali WebImpl geht auf escali.schematron-quickfix.com online
    - Vorstellung auf dem Schematron UserMeetup am PreConfDay der XML Prague
  - Veröffentlichung des Escali Plugins
    - Vorstellung auf dem Markupforum 2018
- 2018
  - Second Draft des Standards ist online

SQF

Die Sprache

Grundkonzepte

SQF

- Schematron erweitern, über foreign-Struktur
- Eigener Namespace
  - Von Schematron unabhängig
- Übernahme vom Schematron-Konzept
  - Einbettung von XPath
  - einfache Fälle mit SQF-Elementen umsetzbar
  - mit XSLT erweiterbar, für komplizierte Fälle
  - XSLT-Abstraktionen, daher sollten Implementierungen auf XSLT basieren

Declare – **A**ct – **C**reate – **R**eact

Declare – QuickFixes anlegen

SQF

```
<sch:rule context="title">
    <sch:report test="comment()" sqf:fix="deleteComments resolveComments">
       Title should not contain comments
    </sch:report>

    <sqf:fix id="deleteComments">
        <sqf:description>
            <sqf:title>Delete the comments</sqf:title>
            <sqf:p>Delete all comments in the current title.</sqf:p>
        </sqf:description>
        <sqf:delete match="comment()"/>
    </sqf:fix>

    <sqf:fix id="resolveComments">
        <sqf:description>
            <sqf:title>Resolve the Comments to text</sqf:title>
            <sqf:p>All comments will be transformed to text nodes.</sqf:p>
        </sqf:description>
        <sqf:replace match="comment()" select="string(.)"/>
    </sqf:fix>

</sch:rule>
```

Titel und weiterführende Beschreibung

Titel und weiterführende Beschreibung

Aktion(en)

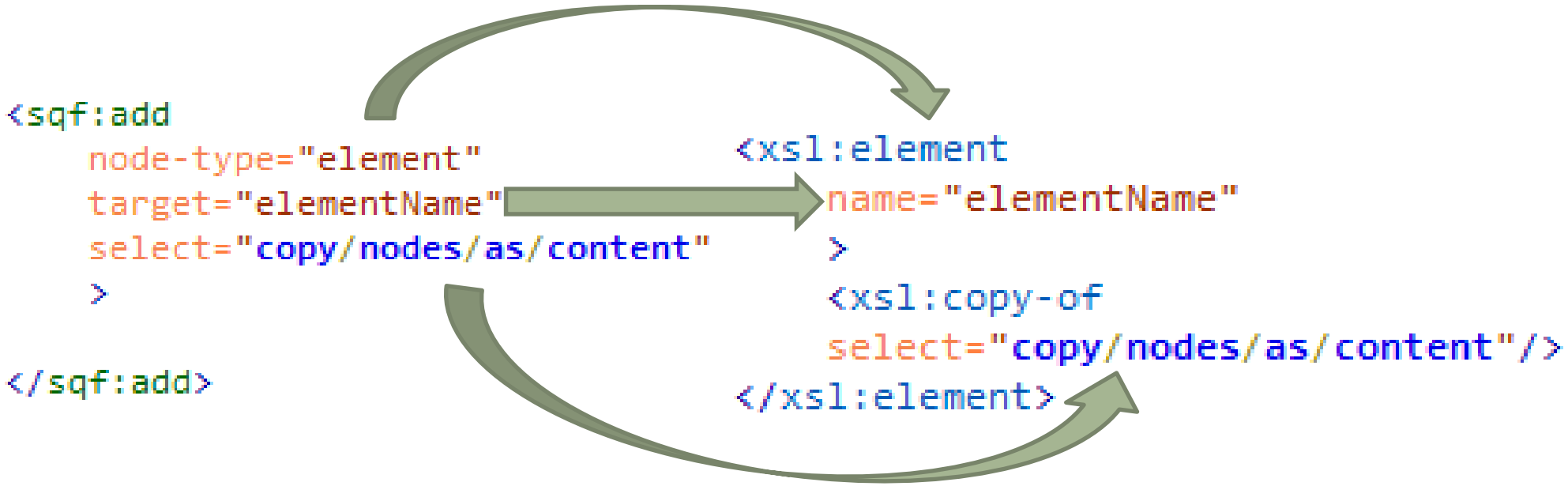
Act – Aktionen ausführen

SQF

- Verfügbare Aktionen
  - Löschen (`<sqf:delete>`)
  - Ersetzen (`<sqf:replace>`)
  - Hinzufügen (`<sqf:add>`)
  - Text ersetzen (`<sqf:stringReplace>`)
- Ausgehend von „Anchor nodes“ mit `@match`
- Spezifizierende Attribute
  - `@position` für `<sqf:add>`
  - `@regex` für `<sqf:stringReplace>`
- Bis auf `<sqf:delete>`, beinhaltet eine Aktion das erzeugen neuer Knoten

Create – Erzeuge neue Knoten

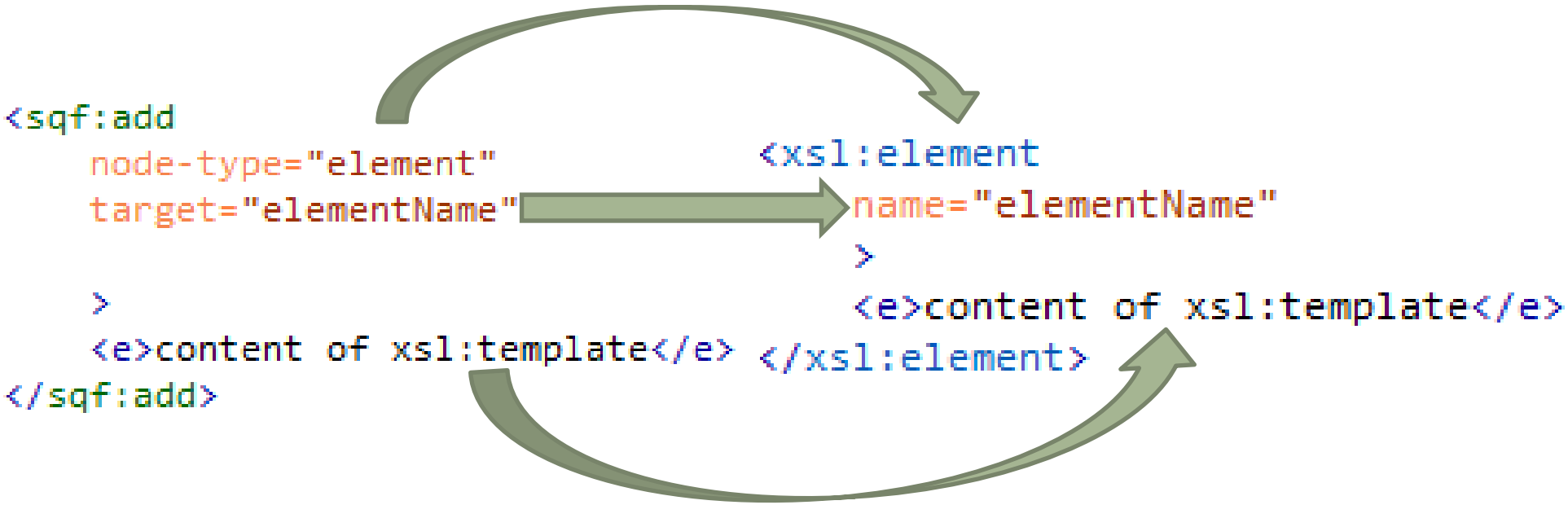
SQF



- Bis auf @select und „Template-Content“ kann hier beliebig kombiniert werden.

Create – Erzeuge neue Knoten

SQF



- Bis auf @select und „Template-Content“ kann hier beliebig kombiniert werden.

React – Fallunterscheidungen

SQF

- UserEntry
  - Parameter, dessen Wert vom User beim Ausführen bestimmt werden kann
- Wiederverwendung von QFs
  - sqf:call-fix / sqf:param
- QF nur zu Verfügung stellen, wenn Konstellation passt
  - `@use-when`-Attribut
  - XPath-Ausdruck muss `true` zurück geben

Neuerungen im Second Draft

SQF

- Generische QFs
  - `@use-for-each`-Attribute
  - XPath-Ausdruck gibt eine Sequenz zurück
  - Für jedes Item wird ein QF erzeugt
- Neues Inhaltsmodell von `<sqf:fix>`
  - Bisher:
    - Entweder QF-Aufruf oder Aktionen
    - Die Description konnte nicht vererbt werden bei QF-Aufrufen, da `<sqf:description>` obligatorisch war
  - Neu:
    - Beliebige Kombinationen aus Aktionen und QF-Aufrufen
    - Description ist nur obligatorisch, wenn
      a) Ein ActivityElement enthalten ist, oder
      b) Mehrere `<sqf:call-fix>`-Elemente enthalten sind

Neuerungen im Second Draft

SQF

- Lokalisierungskonzept
  - `<sqf:title>` und `<sqf:p>` erhalten ein `@ref` Attribut
    - Referenz auf `<sch:diagnostic>`
    - Referenz auf Java-Property files
    - Referenz auf…?

SQF

Die Oxygen-Implementierung

Oxygen Build-in

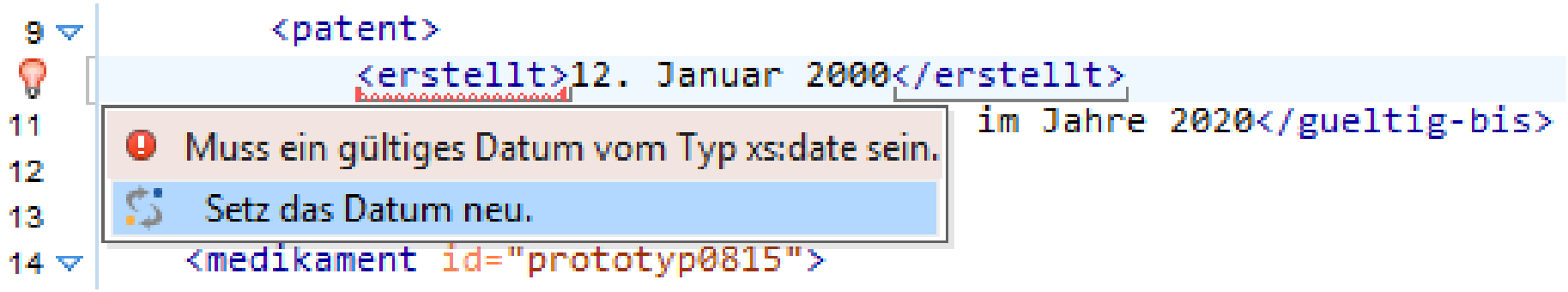
SQF

- Oxygen XML Editor seit der Version 17.0
  - Bisher die einzige Editor-basierte Implementierung
  - Basiert auf XQuery Update
  - Mit der Version 18.1 werden die wichtigsten Features von Schematron QuickFix (First Draft) unterstützt
- Features in 18.1+
  - Framework → Oxygen als SQF-Entwicklungsumgebung
  - XInclude support
  - QFs in externen Dateien

Oxygen Build-in

SQF

- Eingebettet in die normale Schematron-Validierung



- Alternativ:
  - Tastenkürzel „Alt+1“
  - Als Tool-Tipp
- Oxygen stellt so auch andere Refactoring-Quick-fixes zur Verfügung, die nicht auf SQF basieren

SQF

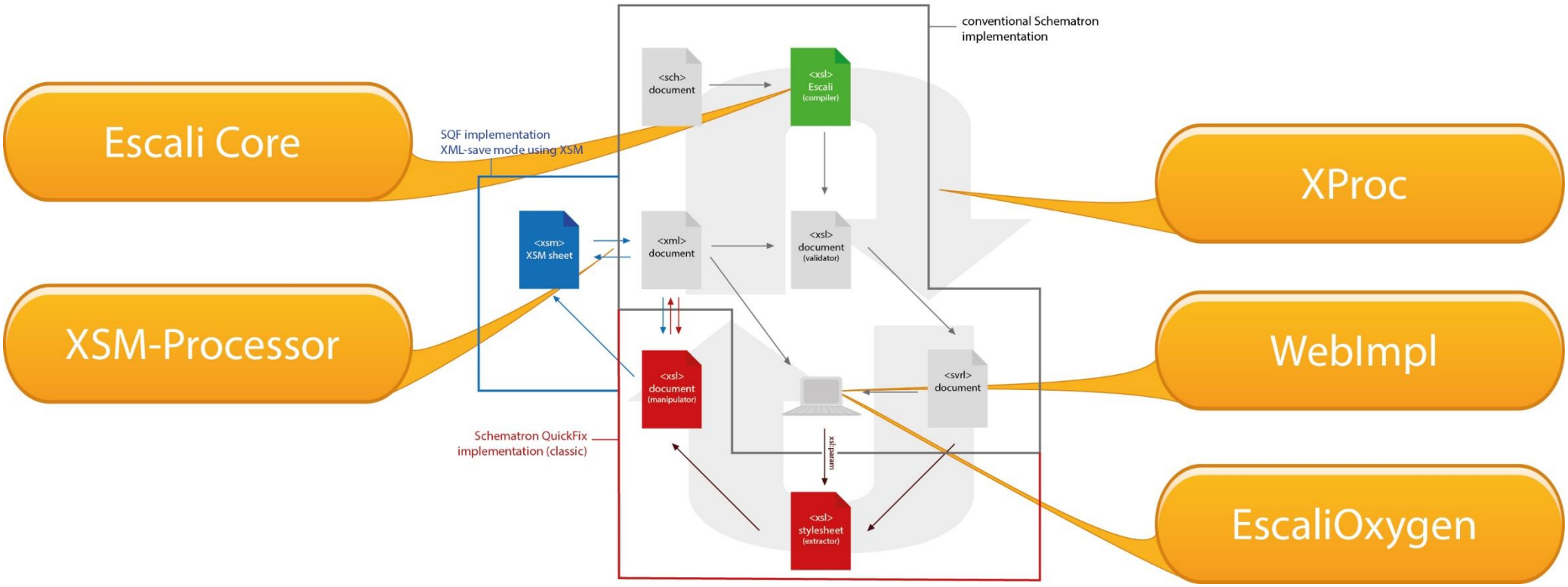
Der Escali

Escali-Projekt

SQF

- Referenz-Implementierung, „Sandkasten“
- Commandline-Implementierung
  - basierend auf XProc / Java
  - Für Unit-Tests
- Im Browser: escali.schematron-quickfix.com
  - Erste OpenSource-Implementierung mit Editor-Character
  - basierend auf Saxon-CE → große Performance-Probleme
- **Escali Oxygen Plugin**
  - https://github.com/schematron-quickfix/escali-package/tree/master/escaliOxygen

Escali Module



SQF

Der Escali Oxygen Plugin

Vergleich Escali / Oxygen

SQF

- Warum eine „neue” Implementierung im Oxygen?
  - Oxygen ist momentan de-facto-Standard
  - Evaluation, für welche Features tatsächlich auch Bedarf existiert
  - Die folgenden Beispiele sollen aufzeigen, dass die Sprache noch mehr Potential bietet, als der Oxygen momentan unterstützt.
- Alternativen zum Oxygen Plugin
  - Eigener Editor
  - Andere Plugin-Schnittstellen nutzen
    - XMLSpy Plugin (C#!), Eclipse Plugin
  - Web-Editor (Server basiert, JS)

Setup

SQF

- Oxygen Add-on-Funktion
  - Hilfe → Neue Addons installieren
  - Add-Ons anzeigen von:
    https://raw.githubusercontent.com/schematron-quickfix/escali-package/master/escaliOxygen/build/extensions.xml
  - Escali Oxygen Plugin Version 0.1.3 installieren
  - Installationsanweisungen folgen und Oxygen neustarten

Überblick

SQF

- 3 neue Views
- Schematron View (Main view)
  - Konfiguration des Plugins
  - Alternative Fehlerview, nur für Schematron-Fehlermeldungen
  - Globale QF-Aktionen
- QuickFix View
  - Zeigt für eine ausgewählte Schematron-Fehlermeldung die verfügbaren QFs an
  - Ausführen einzelner QFs
- UserEntry View
  - Zeigt für selektierte QFs die verfügbaren UEs an

Konfiguration

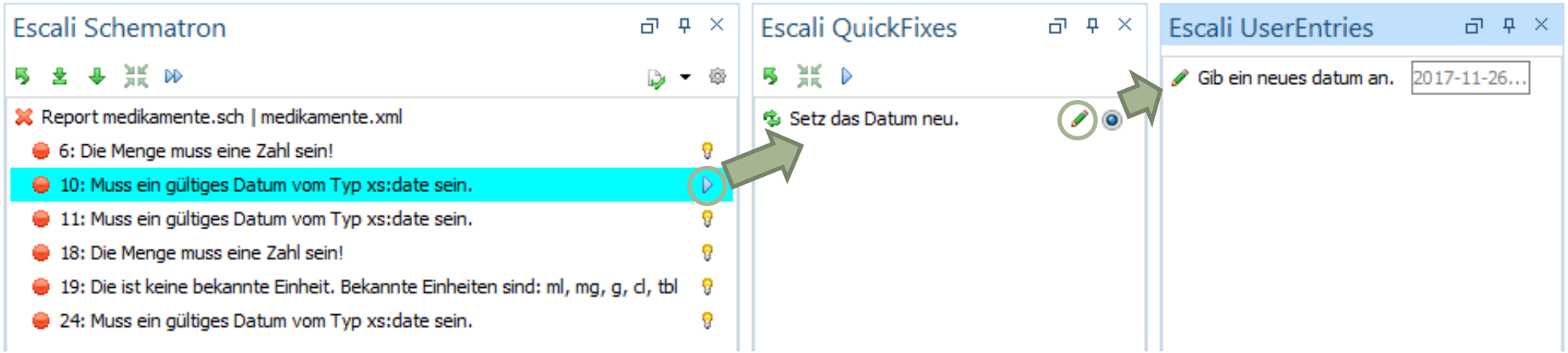
SQF

- Aktivieren / Deaktivieren des Plugins
- Auswahl der Saxon-Version (HE/PE/EE)
- Sprach-Einstellungen für Lokalisierungen
- Assoziations-Regeln für Schemata
  - `<?xml-model?>`-PI Erkennung aktivieren / deaktivieren
  - Assoziations-Tabelle

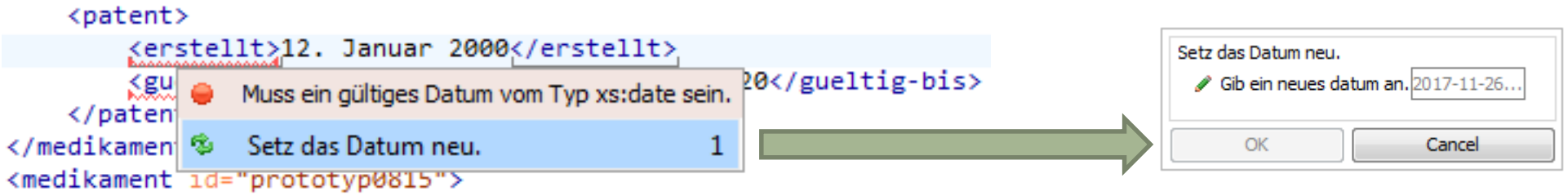
Ausführen von QFs

SQF

Über die Views



Inline-Alternative mit Tastaturkürzel „Alt+1“ oder Strg + Rechte Maustaste



SQF

Der Vergleich

Mehrere QuickFixes auf einmal

SQF

- Oxygen kann nur einen QuickFix auf einmal ausführen
  - Zwei QuickFixes könnten sich in die Quere kommen.
  - Mehrere QuickFixes könnten ein Dokument invalide machen.

- Escali bietet die Möglichkeit, mehrere QFs zu selektieren und auszuführen
  - Es kann sehr praktisch sein.
  - Anwendung: 100 gleiche Fehler, 99 müssen auf die gleiche Weise gelöst werden.
  - Ein QF gibt nie die Garantie, dass das Dokument danach valide ist.

UserEntries – Datentypen

SQF

- Was ist ein UserEntry?
  - Spezieller Parameter für den QuickFix
  - Wird vom User gesetzt, beim Ausführen des QFs
  - Use Case: Ein Wert ist invalide. Ein QF will den Wert neu setzen. Dazu erhält er einen UE, mit dem der Nutzer den neuen Wert setzen kann.
  - Einem UE kann ein Typ zugewiesen werden.
- Oxygen
  - Basis-Unterstützung
  - Keine Unterstützung von Datentypen
- Escali Plugin
  - Unterstützt die Typen `xs:string`, `xs:integer`, `xs:double`, `xs:date`, `xs:time`, `xs:boolean`
  - Bietet abhängig vom Typ unterschiedliche Eingabemasken an

UserEntries – Enumerations

SQF

- Häufig reicht ein generischer Typ nicht aus
- In vielen Fällen sind nur bestimmte Werte erlaubt.
- Wünschenswert wäre eine Drop-Down-Liste mit allen erlaubten Werten („Enumeration“).
- Die Sprache enthält noch keine vollständige Struktur dafür.

- Feature des Escali Plugins
  - Der Escali bedient sich hier eines Tricks
  - UEs können einen Defaultwert haben, der per XPath bestimmt wird.
  - Eine Eingabemaske kann dann diesen Wert als Default verwenden.
  - Das Escali Plugin erzeugt eine Enumeration, wenn ein UE eine Sequenz an Werten als Defaultwert hat.

Umgang mit Entities

SQF

- Generelles Problem bei der XML-Verarbeitung
  - Erster Schritt beim XML-Parsen: Auflösen der Entities
  - Kein Zugriff auf Enities mit XPath/XSLT
- Herausforderung bei SQF:
  - Erhalt aller unbeteiligten Entities beim Ausführen der QFs
  - Oxygen und Escali unterstützen das
- Probleme gibt es dennoch:
  - Kopieren oder Verschieben von Knoten löst die Entities auf
  - Hierzu wird in der nächsten Version des Standards das `<sqf:copy-of>`-Element eingeführt mit einem `unparsed-mode`-Attribut.
  - Ist der `unparsed-mode` aktiv, sollten die Knoten kopiert werden, ohne dass Entities aufgelöst werden.
- Oxygen:
  - Hat bereits angekündigt, diesen Mode wohl vorläufig nicht zu unterstützen.
- Escali Plugin enthält einen ersten Draft für eine Unterstützung

Mikrotypographie

SQF

- Schematron-Probleme in der Mikrotypographie
  - Schematron kennt nur Knoten-basierte Fehler-Locations
  - Wenn einzelne Text-Snippets (einzelne Zeichen, Abkürzungen, Mengenangaben) die Fehlerursache sind, gibt Schematron maximal pro Textknoten, eher pro Absatz eine Fehlermeldung aus
  - Das tatsächliche Text-Snippet muss manuell gefunden werden
  - Mehrere Vorkommen des gleichen Text-Snippets in einem Textknoten / Absatz werden als ein Fehler zusammengefasst.
  - SQF kann deshalb auch nur ein QF pro Fehler anbieten, der dann alle Text-Snippets auf einmal korrigieren muss.
- Escali-Erweiterung für Schematron
  - Erlaubt Textknoten per Regex aufzusplitten
  - Jeder Regex-Treffer wäre dann ein Text-Snippet, der mit Schematron-Mitteln geprüft werden kann
  - Die Fehlermeldung wird für den Text-Snippet angezeigt.
  - QFs können den betroffenen Text-Snippet einzeln behandeln

Nachteile des Escali Plugins

SQF

- Es ist ein Plugin
- Masterfiles-Validierung wird (noch) nicht unterstützt
  - Die Oxygen-API liefert dazu zu wenige Informationen
- Schema-Assoziierung
  - Im Oxygen gibt es viele Möglichkeiten, ein XML-Dokument mit einem Schemtron-Schema zu verknüpfen (u.a. Framework, Validierungsszenario, `<?xml-model?>`-PI)
  - Die Oxygen-API bietet keine Informationen, welches Schematron-Schema mit dem XML-Dokument verknüpft ist
  - Das Escali-Plugin kann deshalb nur auf die `<?xml-model?>`-PI zurückgreifen
  - Zusätzlich gibt es eine Association Table, ähnlich der Association Rules eines Frameworks.
- Doppelte Validierung
  - Das Escali Plugin kann nur nachträglich die Validierung des Oxygens manipulieren
  - Kennt der Oxygen das Schematron auch, findet eine doppelte Validierung statt.

SQF

In der Praxis

SQF im Praxis-Einsatz

SQF

- ParsX-Schematron von Pagina
- Thieme Verlag
- OpenSource
  - Dynamic Information Model (DIM) für DITA
    - https://github.com/oxygenxml/dim
  - FO-Check von Antenna House
    - https://github.com/AntennaHouse/focheck
  - TEI-C Wiki
    - https://wiki.tei-c.org/index.php/Unicode_normalization

Blick über den Tellerrand

SQF

- XSLT Styleguide
  - Mehrköpfiges XSLT-Entwickler-Team
  - Projekt über 9 Jahre
  - > 1000 Stylesheets zu pflegen
  - Einheitliche Schreibweise mit Schematron prüfen
  - QuickFixes anbieten zum schnelleren beheben
- sqf.sch
  - Schematron zum Überprüfen von SQF
  - QuickFixes zur Hilfe bei der Entwicklung
  - Teil des Oxygen SQF-Frameworks
  - Weiterführender Gedanke: QuickFixes als interaktiver Entwicklungsguide verwenden

SQF

Ausblick

Nächste Ziele

SQF

- Lokalisierung von SQF
  - Den W3C-Note-Standard vollenden und veröffentlichen
  - Umsetzung im Escali
- Support von Schematron 2016
- Masterfiles-Validierung
- Weitere Plugins in anderen Editoren
  - Das Eclipse-Plugin ist so aufgesetzt, dass es relativ leicht in andere Editoren portiert werden kann (keine komplette Neuentwicklung)
  - Java-basiert

Weitere Links

SQF

- SQF Website
  - www.schematron-quickfix.com
  - Weitere Escali-Schematron-Erweiterungen: http://www.schematron-quickfix.com/escali/escali-ext_en.html
  - Folien und Beispiele zu diesem Vortrag: https://github.com/nkutsche/SchematronQuickFix-2017-11-17/tree/variants/de/xugber-2018-03-08
- Github-Organisation: https://github.com/schematron-quickfix/
- W3C-Gruppe: https://www.w3.org/community/quickfix/
- Plugin-Dokumentation:
  - https://github.com/nkutsche/presentations-EscaliOxygenPlugin
- oXygen-Dokumentation
  - https://www.oxygenxml.com/doc/versions/19.1/ug-editor/topics/schematron-quick-fixes.html

SQF

Fragen?