XML-Datenbanken

&

XQuery

Digitale Edition

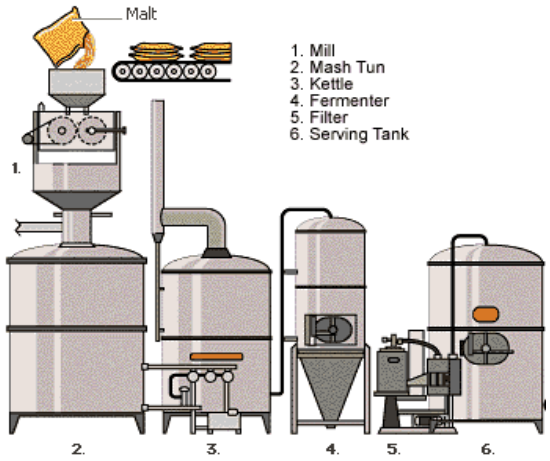
XML

XSLT

XQuery

XML-DB

XSL-FO



Programm

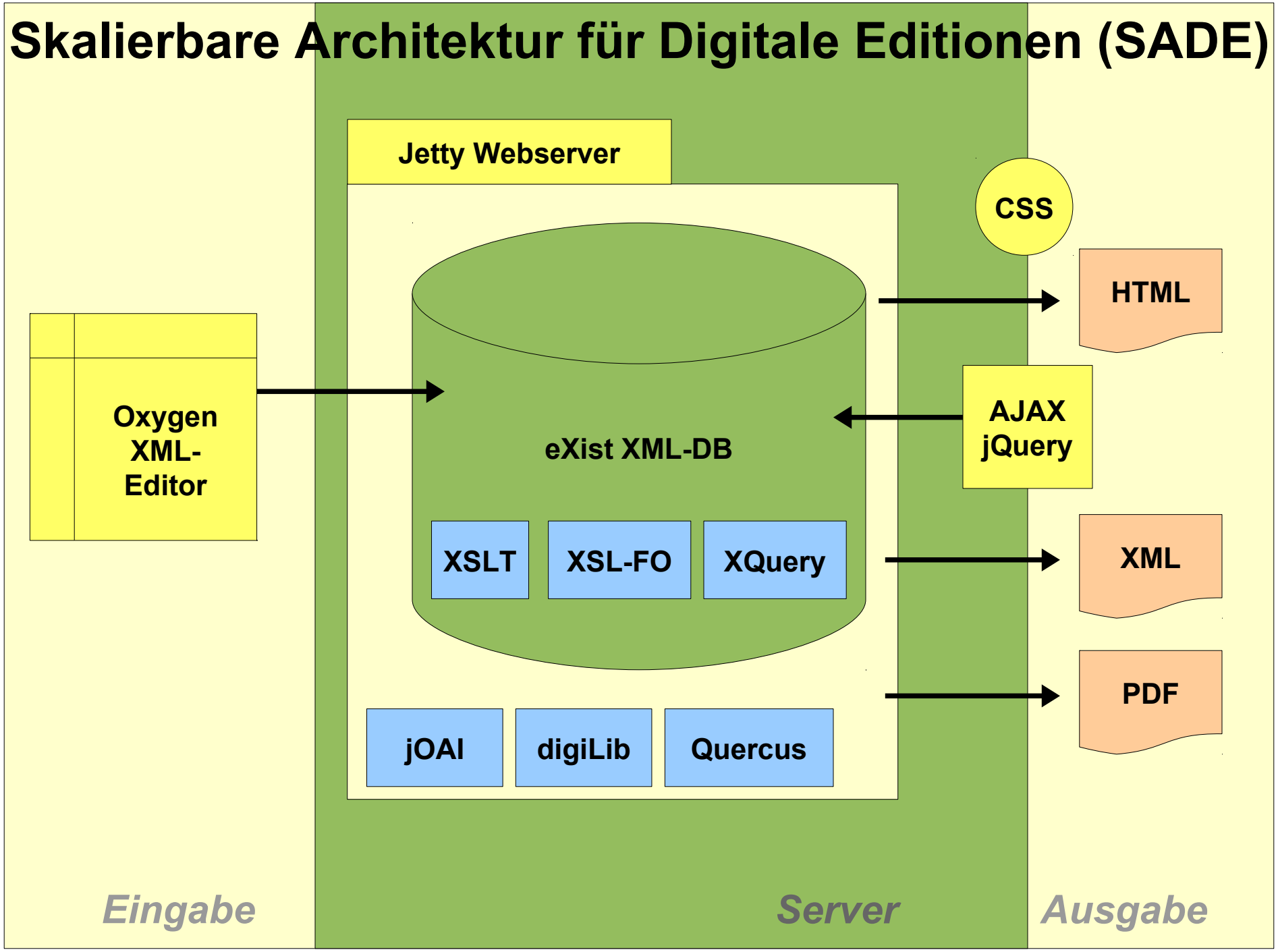
- Ausgewählte Biere verköstigen
- Brauereibesichtigung
- Anwendung der Hefe erlernen
- brauen

Telota - Digitale Editionen

- http://telota.bbaw.de/AvHBriefedition
- http://telotadev.bbaw.de/mgh
- http://localhost:8080/mega

- Mehr unter: http://www.telota.de

Skalierbare Architektur für Digitale Editionen (SADE)



XML - Datenbanken

- Wozu?
  - Analyse, Teilknoten, mehrere Eingabedokumente, verschiedene Präsentationsformate
  - Für den „natürlichen“ Umgang mit Semistrukturierten Daten, da man diese nicht auf das relationale Modell, also Tabellen abbilden muss.
  - Für die leichte und effiziente Verarbeitung von XML-Dokumenten mittels X-Technologien

XML - Datenbanken

- man unterscheidet zwischen *XML-enabled* und *nativen* XML-Datenbanken
- optimiert für den Umgang mit XML-Dokumenten
- erfordern kein bestimmtes physisches Speicherungsmodell
- textbasiert vs. modellbasiert
- Es existieren keine Normalformen

XML - Datenbanken

- organisieren XML-Dokumente in *Collections*
- unterstützen Validierung
- unterstützen XPath, XSLT, XQuery, XUpdate
- bieten viele Schnittstellen: XML:DB, XML-RPC, REST, WebDav, SOAP, XQJ
- ausführliche Liste von nativen XML-Datenbanken: http://www.rpbourret.com/xml/ProdsNative.htm

eXist

- bietet noch mehr:
  - eine Umgebung zur Erstellung vollständiger Webapplikationen
  - Lucene-basierte Volltextsuche
  - verschiedene Indexierungsmöglichkeiten
  - Versionierung
  - sehr gute Dokumentation
  - Java Admin Client
  - Anbindung an Oxygen

SADE

Get & Run SADE

- Download: http://telota.bbaw.de/sade/sade.zip
- in ein beliebiges Verzeichnis entpacken und in dieses wechseln
- Starten:
  - Windows: bin/Jetty-Service.exe
  - Linux/Mac: Konsole öffnen und „*java -jar start.jar*“ eingeben
- im Webbrowser http://localhost:8080/ aufrufen

XQuery

- XML Query Language (http://www.w3.org/TR/xquery/)
- Abfragesprache für XML-Dokumente und XML-Datenbanken
- benutzt XPath als Grundlage
- Datentypen aus XML-Schema
- keine XML-Syntax
- eignet sich für Extraktion, Selektion und Konstruktion neuer Elemente

XQuery

```
xquery version "1.0";

declare namespace  xslfo="http://exist-db.org/xquery/xslfo";

let $col := collection("/db/mgh/data")
let $docID := request:get-parameter("docID", "")
let $qString := request:get-parameter("qString", "")

let $text := $col//text[.//idno = $docID]

let $pdf1 := transform:transform($text, doc('/db/mgh/scripts/pdf.xsl'), $para)
let $pdf := xslfo:render($pdf1, 'application/pdf', (), doc('/db/mgh/scripts/fop-cfg.xml'))

return response:stream-binary($pdf, 'application/pdf', concat($docID,'.pdf'))
```

XQuery

- Grundlegende Datenstruktur ist die *Sequenz*
  - diese kann aus XML-Knoten und/oder anderen Datentypen bestehen (z.B. xs:string, xs:date, xs:integer)
  - Beispiele:
    - (1, 2, 5)
    - ("hallo welt")
    - (text)
    - (text, "text", 12)
    - "atomar"
  - leere Sequenz: ()

XQuery - Syntax

- FLWOR-Ausdrücke
  - **for**: bindet Ausdrücke elementweise an eine Variable
  - **let**: bindet Ausdrücke als Ganzes an eine Variable
  - **where**: schränkt Ergebnismenge durch Bedingungen ein
  - **order by**: sortiert die Ergebnismenge
  - **return**: Rückabe der Ergebnismenge
- Kommentare: **(:** Das ist ein Kommentar. **:)**

XQuery - Syntax

- Beispiele **for**:
  - `for $item in (<b>text</b>, "text", 12) return $item`
  - `for $b in fn:doc("buecher.xml")/buecher/buch return <buch id="{$b}"/>`
  - `for $i in 1 to 10 return <zahl>{$i}</zahl>`

- Beispiele **let**:
  - `let $message := "Nachrichtentext"`
  - `let $result := <notes>{$app//note}</notes>`
  - `let $count := count($result)`
  - `let $qString :=  request:get-parameter("qString", "")`

XQuery - Syntax

- Beispiele **where**:
  - where $b/autor/text()="Meyer"
  - where $b/year>2000
- Beispiele **order by**:
  - order by $b/autor
  - order by $b/@isbn
- Beispiele **return**:
  - return <parameters>{$parameters}</parameters>
  - return "nur Text"
  - return transform:transform($result, doc("/db/telota/xslt/search.xsl"), ())
  - return for $f in $fileList return <option>{$f}</option>

XQuery - Syntax

```
xquery version "1.0";

let $doc := doc("/db/test/books.xml")

let $authors := for $a in distinct-values($doc//author/text()[starts-with(., "m")])
                        order by $a
                        return <author>{$a}</author>

return <authorList>{$authors}</authorList>
```

XQuery - Syntax

- Bedingungen:
  - `let $a := if ($x > 0) then "positiv" else "negativ"`
  - ```
    let $a := if ($x > 0) then "positiv"
              else if ($x < 0) then "negativ"
              else "null"
    ```
- Operatoren:
  - sind identisch mit XPath-Operatoren
- Funktionen:
  - sind identisch mit XPath-Funktionen
  - Zusätzliche eXist-spezifische Funktionen in der *Function Library*
    ```