XSL Transformation

Eine praxisorientierte Einführung

Philip Heck

Agenda

- Motivation
- „Hello World“ - Getting Started with XSLT
- XSLT Verarbeitung im Detail
- XPath

Motivation

- datenzentriertes XML als Standard für den Dokumentenaustausch
- XML Datenrepräsentationen sind nicht sonderlich lesbar

Motivation

```
<dokument titel="XSL Transformation" autor="Philip Heck">
  <kapitel titel="Motivation">
    <paragraph> XSLT ist ... </paragraph>
    <paragraph> ... </paragraph>
  </kapitel>
  <kapitel titel="'Hello World' - Hands on">
    ...
  </kapitel>
  <kapitel titel="XPath">
    ...
  </kapitel>
</dokument>
```

Motivation

XSL Transformation

von Philip Heck

1. Motivation

XSLT ist ...

...

2. Hello World

...

3. XPath

...

Motivation

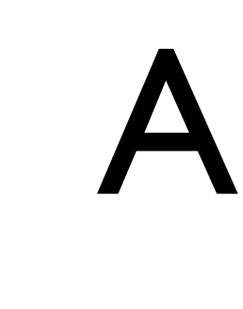
- Das Bedürfnis XML Dokumente in eine andere - z.B. lesbarere - Form zu transformieren hat zur Entwicklung von XSLT geführt

XSLT

- Extensible Stylesheet Language Transformation
- neben XSL-FO und XPath einer der drei Komponenten der XSL
- offizielle Empfehlung des W3C
- XML in HTML, XML, PDF, SVG, Text, ... transformieren

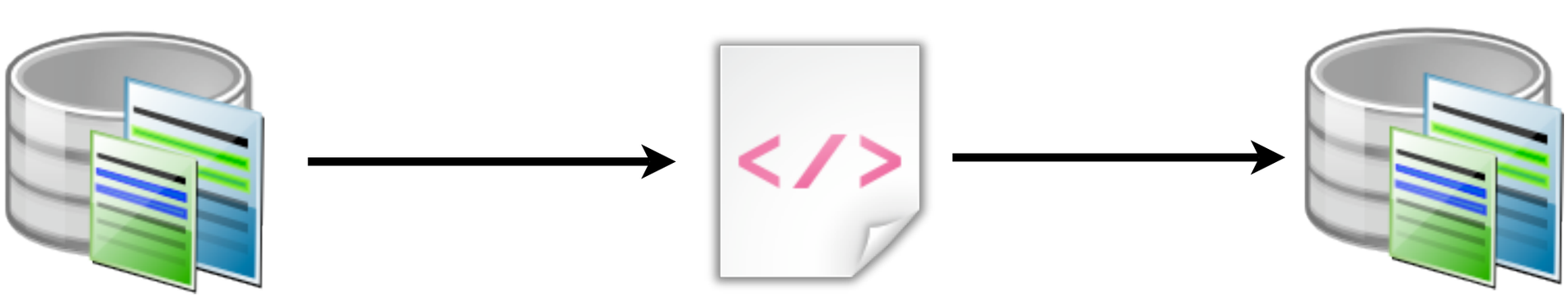
Anwendungsgebiete

- Website liefert Informationen an eine Vielzahl von Endgeräten



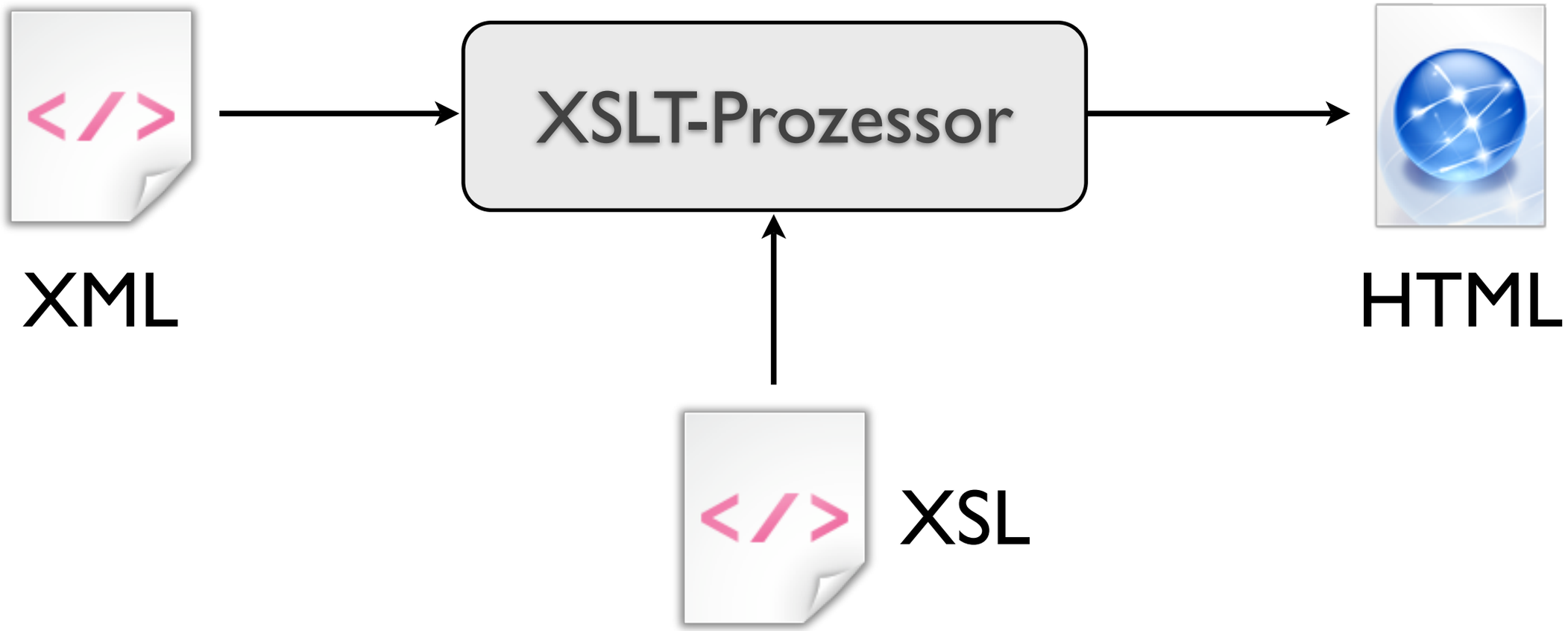
Anwendungsgebiete

- Datenaustausch zwischen heterogenen Datenbanksystemen



Getting Started

Transformationsprozess



XSLT Prozessoren

- *xsltproc* - XSLT C Library des Gnome Projekts

  http://xmlsoft.org/XSLT/xsltproc2.html

  ```
  xsltproc -o AUSGABE.HTML STYLESHEET.XSL QUELLE.XML
  ```

- *Xalan* - XSLT-Engine des Apache Projekts

  http://xalan.apache.org/

- *Saxon* - XSLT-Engine vom XSLT Autor Michael Kay

  http://saxon.sourceforge.net/

  ```
  java -jar $SAXON/saxon8.jar QUELLE.XML STYLESHEET.XSL
  ```

Die Quelldatei

```
<?xml version="1.0"?>
<nachricht>
  Hallo Welt!
</nachricht>
```

Das Ergebnis

```
<html>
  <body>
    <h1>Hallo Welt!</h1>
  </body>
</html>
```

Das Stylesheet

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:output method="html"/>

<xsl:template match="/">
  <xsl:apply-templates select="nachricht"/>
</xsl:template>

<xsl:template match="nachricht">
  <html>
    <body>
      <h1>
        <xsl:value-of select="."/>
      </h1>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Das Stylesheet

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:output method="html"/>
```

- es handelt sich um ein XSLT 1.0 Dokument
- Ausgabe soll *html* sein.
- XSLT definiert 3 Ausgabemethoden: *html*, *xml* oder *text*.

Das Stylesheet

```
<xsl:template match="/">
  <xsl:apply-templates select="nachricht"/>
</xsl:template>
```

- Gibt es ein Template für den aktuellen Knoten (“/“)?
- Der „/“ ist ein *XPath-Ausdruck*, der „Wurzel des Dokuments“ bedeutet. Später mehr dazu ...
- Wende auf alle „nachricht“-Elemente das dazu passende Template an

Das Stylesheet

```
<xsl:template match="nachricht">
  <html>
    <body>
      <h1>
        <xsl:value-of select="."/>
      </h1>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

- XML-Elemente ohne XSL-Namensraum oder reiner Text werden einfach ausgegeben
- `<xsl:value-of select=“.“/>` schreibt den Wert des aktuellen Elements in den Ausgabestrom

vielfältige Möglichkeiten

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
<xsl:output method="text"/>

<xsl:template match="/">
  <xsl:apply-templates select="nachricht"/>
</xsl:template>

<xsl:template match="nachricht">
  #!/usr/bin/ruby
  puts “<xsl:value-of select="."/>“
</xsl:template>
</xsl:stylesheet>
```

```
#!/usr/bin/ruby
puts “Hallo Welt!“
```

XSLT Verarbeitung im Detail

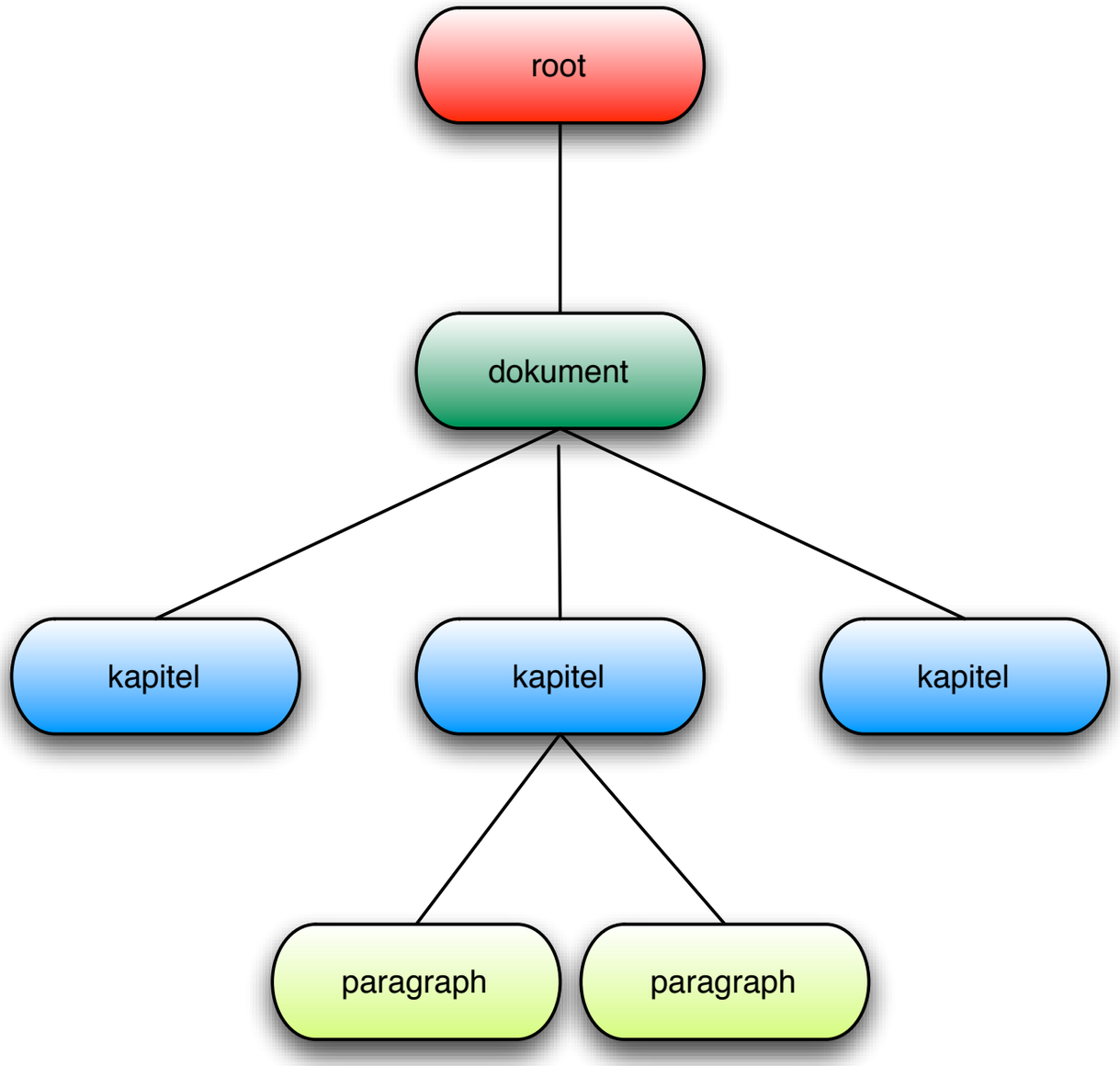
XSLT Verarbeitung

1. Lese und parse die Eingabedokumente in interne Baumstrukturen

Beispiel XML

```
<dokument titel="XSL Transformation" autor="Philip Heck">
  <kapitel titel="Motivation">
    <paragraph> XSLT ist ... </paragraph>
    <paragraph> ... </paragraph>
  </kapitel>
  <kapitel titel="Hello World">
    ...
  </kapitel>
  <kapitel titel="XPath">
    ...
  </kapitel>
</dokument>
```

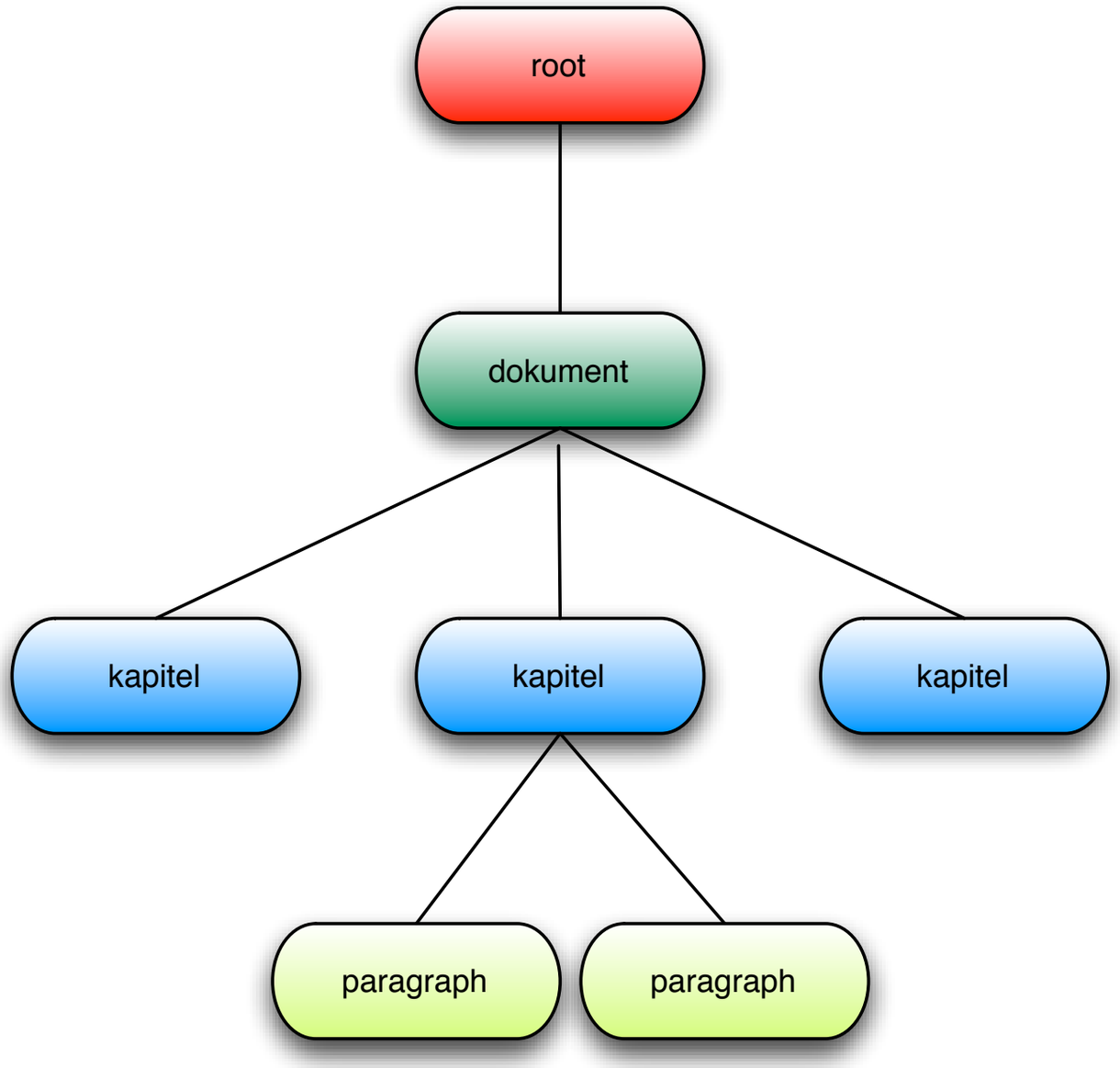
Baumstruktur



XSLT Verarbeitung

1. Lese und parse die Eingabedokumente in interne Baumstrukturen
2. Finde ein `<xsl:template >`, welches auf den aktuellen Knoten zutrifft und führe es aus

XSLT Verarbeitung

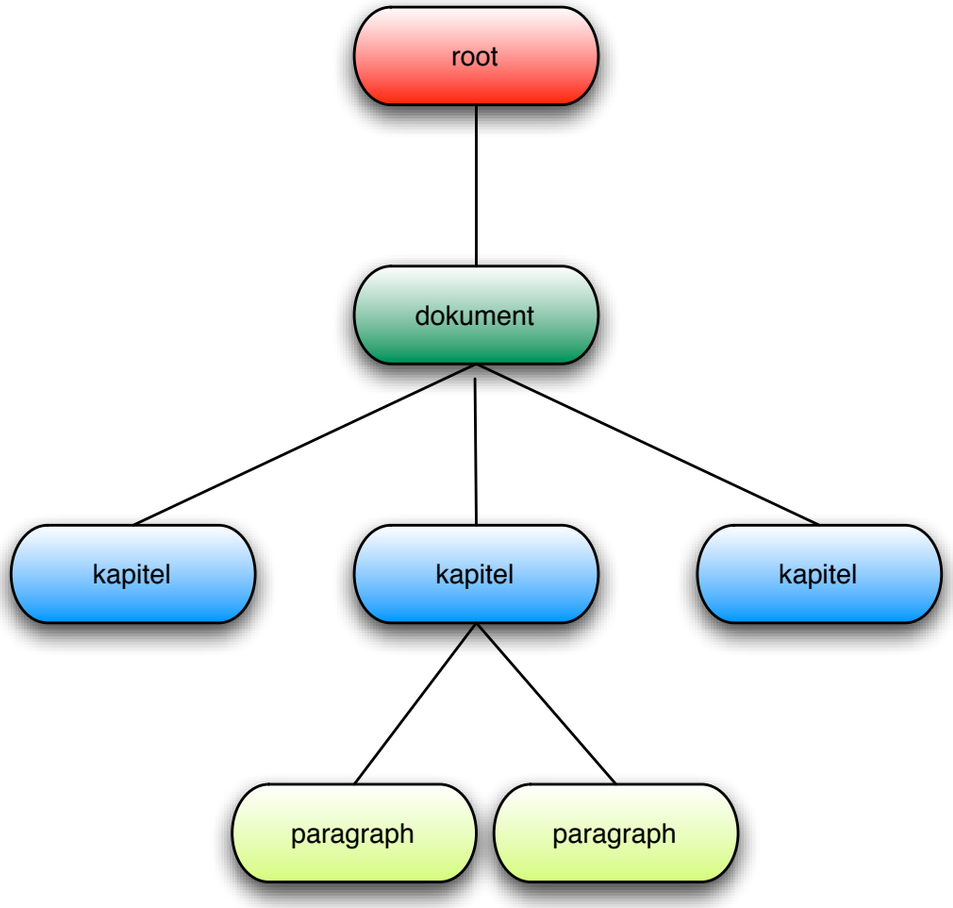



```
<xsl:template match="/">
  <xsl:apply-templates />
<xsl:template/>
```

XSLT Verarbeitung

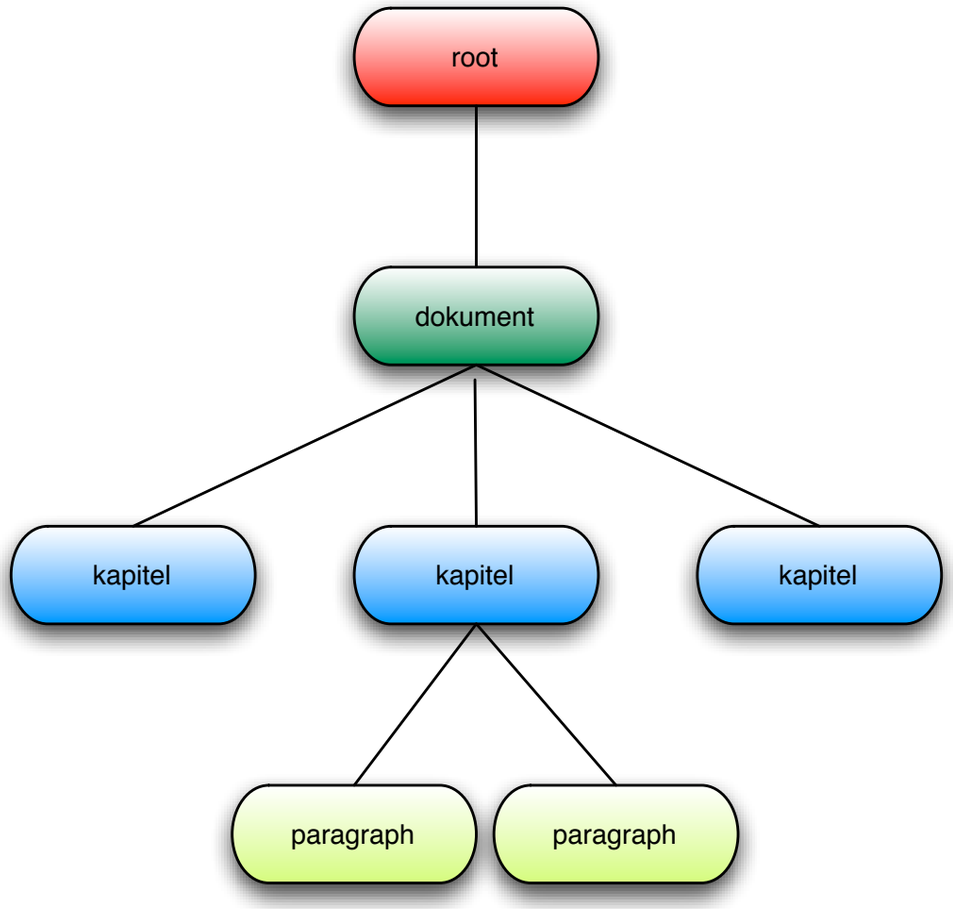
1. Lese und parse die Eingabedokumente in interne Baumstrukturen
2. Finde ein <xsl:template >, welches auf den aktuellen Knoten zutrifft und führe es aus
3. Wenn weitere <xsl:template > Anweisungen vorhanden sind GOTO 2

XSLT Verarbeitung



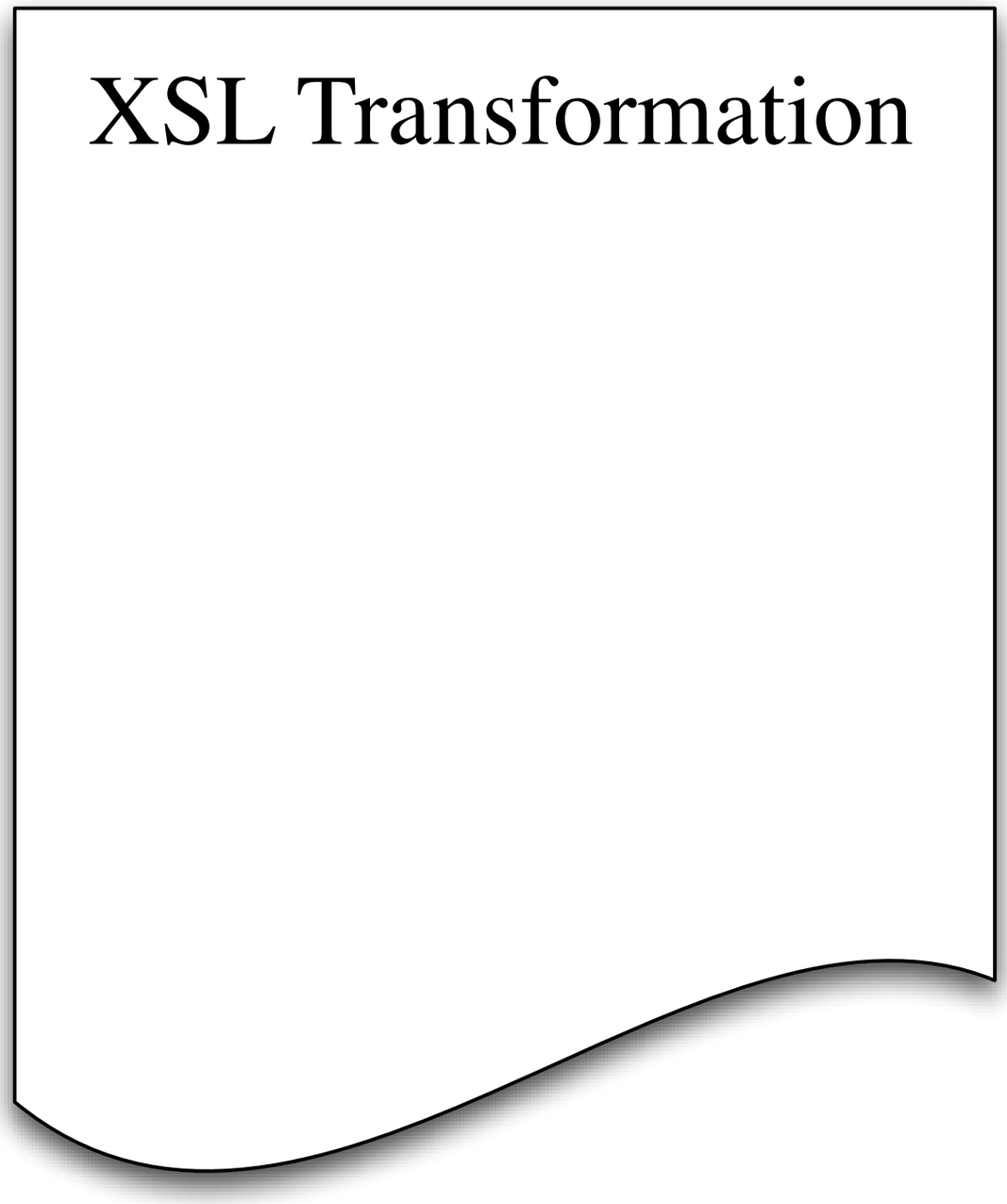

```
<xsl:template match="/">
  <xsl:apply-templates />
<xsl:template/>

<xsl:template match="dokument">
  <h1><xsl:value-of select="@titel"/></h1>
  <xsl:apply-templates select="kapitel"/>
<xsl:template/>

<xsl:template match="kapitel">
  <h2><xsl:value-of select="@titel"/></h2
  <p><xsl:value-of select="." /></p>
  ...
```

XSLT Verarbeitung

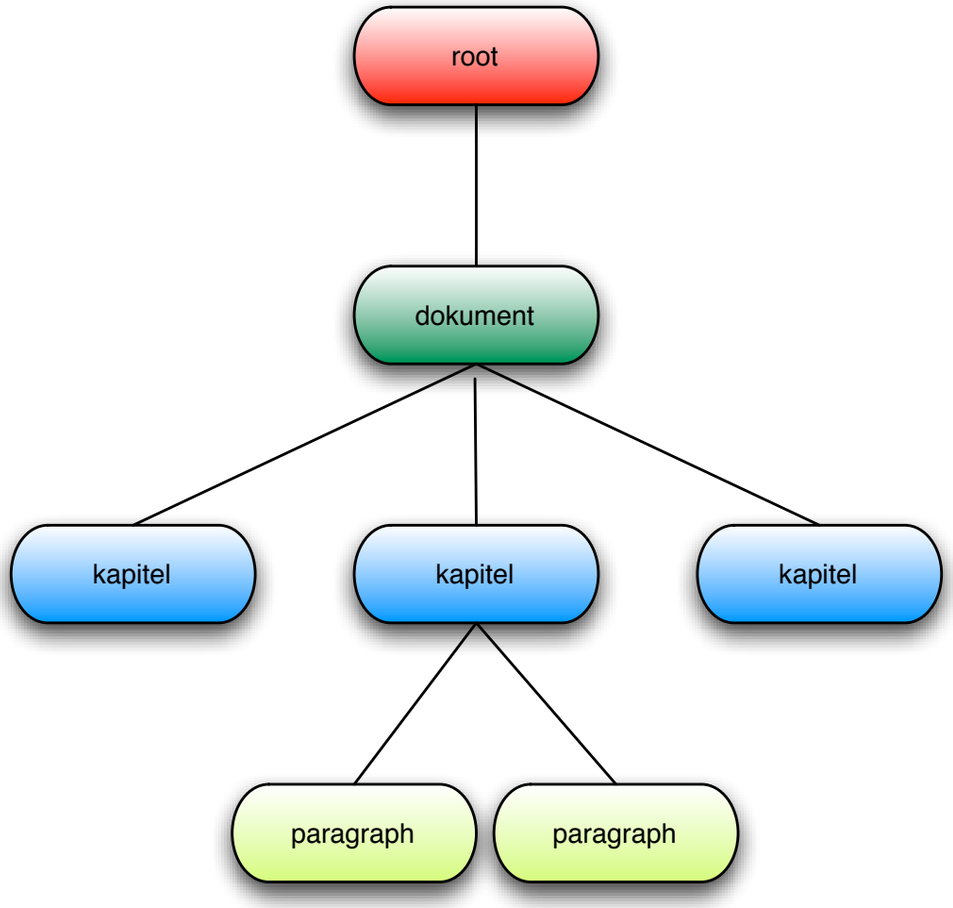



```
<xsl:template match="/">
  <xsl:apply-templates />
<xsl:template/>

<xsl:template match="dokument">
  <h1><xsl:value-of select="@titel"/></h1>
  <xsl:apply-templates select="kapitel"/>
<xsl:template/>

<xsl:template match="kapitel">
  <h2><xsl:value-of select="@titel"/></h2
  <p><xsl:value-of select="." /></p>
  ...
```

XSLT Verarbeitung

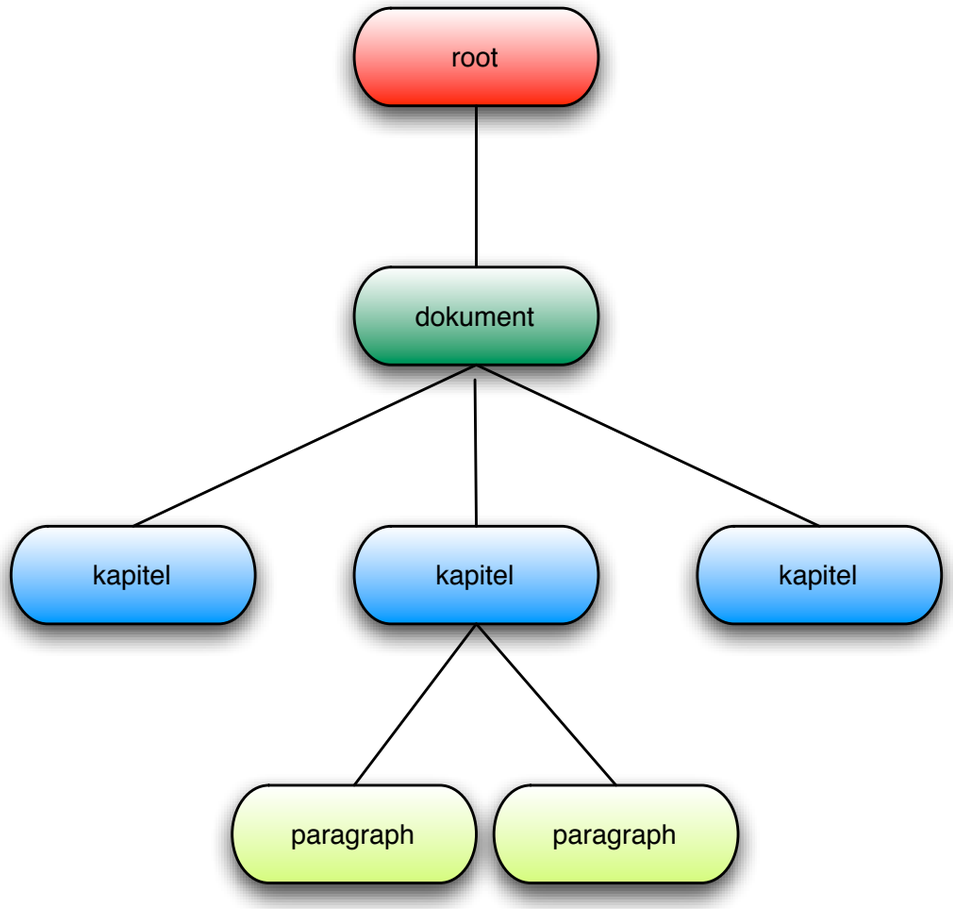


XSLT Verarbeitung




```
<xsl:template match="/">
  <xsl:apply-templates />
<xsl:template/>

<xsl:template match="dokument">
  <h1><xsl:value-of select="@titel"/></h1>
  <xsl:apply-templates select="kapitel"/>
<xsl:template/>

<xsl:template match="kapitel">
  <h2><xsl:value-of select="@titel"/></h2
  <p><xsl:value-of select="." /></p>
  ...
```

XSLT Verarbeitung




```
<xsl:template match=“/“>
  <xsl:apply-templates />
<xsl:template/>

<xsl:template match=“dokument“>
  <h1><xsl:value-of select=“@titel“/></h1>
  <xsl:apply-templates select=“kapitel“/>
<xsl:template/>

<xsl:template match=“kapitel“>
  <h2><xsl:value-of select=“@titel“/></h2
  <p><xsl:value-of select=“.“ /></p>
  ...
```

XSLT Verarbeitung

XSL Transformation

Motivation

XSLT ist ...

...

Hello World

...

XPath

...

XPath

Motivation

- Abfragesprache um Elemente eines XML Dokuments zu adressieren
- XSLT benötigt XPath um Elemente aus dem Quelldokument für die Transformation auszuwählen

Funktionsweise

- XPath sieht ein XML Dokument als einen Baum aus Knoten an, ähnlich Document Object Model (DOM)
- Mit Hilfe von XPath Ausdrücken kann in einem XML Dokument navigiert werden
- 7 Typen von Knoten, die wichtigsten werden kurz erläutert

Beispiel XML

```
<dokument titel="XSL Transformation" autor="Philip Heck">
  <kapitel titel="Motivation">
    <paragraph> XSLT ist ... </paragraph>
    <paragraph> ... </paragraph>
  </kapitel>
  <kapitel titel="'Hello World' - Hands on">
    ...
  </kapitel>
  <kapitel titel="XPath">
    ...
  </kapitel>
</dokument>
```

Wurzelknoten

```
<dokument titel="XSL Transformation" autor="Philip Heck">
```

- der Knoten, der das gesamte Dokument enthält
- in XPath-Ausdrücken durch einen einzelnen “/“ gekennzeichnet
- der String-Wert (`<xsl:value-of select=“/“ />`) ist die Verkettung aller Textknoten der Abkömmlinge des Wurzelknotens

Elementknoten

```
<kapitel titel="Motivation">
  <paragraph> XSLT ist ... </paragraph>
  <paragraph> ... </paragraph>
</kapitel>
```

- jedes Element im Quelldokument
- der Stringwert ist die Verkettung des Textes dieses Knotens und aller seiner Kinder
- `name()` liefert den Namen des Knotens zurück

Attributknoten

```
<dokument titel="XSL Transformation" autor="Philip Heck">
  <kapitel titel="Motivation">
```

- Attribut eines Elementknotens
- Elementknoten sind die Elternknoten ihrer Attributknoten (Attribut → Element)
- Attributknoten sind nicht die Kinder des Elementknotens (Element ↛ Attribut)
- Möchte ein Element auf seine Attribute zugreifen, muss es diese speziell adressieren. Später mehr ...

Kontext

- alle XPath Aktionen werden im aktuellen Kontext interpretiert (`<xsl:value-of select=“.“ />`)
- Kontextknoten: der XPath-Ausdruck wird von diesem Knoten aus ausgewertet
- Ist der Kontext eine Sammlung von Knoten: existieren *Kontextposition* und *Kontextgröße*

Adressierung von Elementen

- Wurzelknoten auswählen

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

- Kontextknoten auswählen

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

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

- Ebenen adressieren

```
<xsl:apply-templates select="dokument/kapitel" />
```

relative und absolute Adressierung

- absolute Adressierung

```
<xsl:apply-templates select=“/dokument/kapitel“ />
```

- relative Adressierung

```
<xsl:apply-templates select=“dokument/kapitel“ />
```

Lokalisierungspfade

- Name des Knoten

```
<xsl:value-of select="name()" />
```

- Attribut des Knoten

```
<xsl:value-of select="dokument/@titel" />
```

- Text eines Knoten (Inhalt ohne Kindelemente)

```
<xsl:value-of select="dokument/text()" />
```

Wildcards/Platzhalter

***** wählt alle Elementknoten im aktuellen Kontext aus

@* wählt alle Attributknoten im aktuellen Kontext aus

//element wählt *element* aus, unabhängig von seiner Position im Dokument. Performancekritisch! Je spezieller der XPath Ausdruck, desto effizienter die Verarbeitung.

Achsen

- Mit Achsen lässt sich ausgehend vom aktuellen Kontext im XML Dokument navigieren
- z.B. alle Vorfahren eines Knoten,
- alle Kinder eines Knoten,
- alle Geschwister, d.h. alle Knoten, welche den selben Elternknoten haben,
- usw.

Achsen

- child-Achse: alle Kinder des Kontextknoten

```
<xsl:apply-templates select="child::dokument" />
```

- parent-Achse: der Elternknoten des Kontextknoten. Äquivalent zu "..".

```
<xsl:apply-templates select="parent::kapitel" />
```

Achsen

- self-Achse: der Kontextknoten selbst. Äquivalent zu “.“.

```
<xsl:apply-templates select=“self::kapitel“ />
```

- attribute-Achse: alle Attribute eines Knoten. Äquivalent zu “@“.

```
<xsl:apply-templates select=“attribute::titel“ />
```

Prädikate

- Mit sog. Prädikaten kann ein Ergebnis weiter eingeschränkt werden
- Prädikate stehen in eckigen Klammern

```
<xsl:apply-templates select="kapitel[2]" />
```

- Ist das Prädikat für einen Knoten `true`, wird dieser ausgewählt

```
<xsl:apply-templates select="kapitel[position()=2 and
@titel]" />
```

Prädikate

- Zur Auswahl können XPath und XSLT Funktionen verwendet werden

```
<xsl:apply-templates select="kapitel[last()]" />

<xsl:apply-templates
select="kapitel[@titel="Motivation"]" />
```

Quellen

- Michael Kay, *XSLT Programmer's Reference*, Wrox Press Ltd. Birmingham, 2000

- Doug Tidwell, *XSLT*, O'Reilly Media, Inc., 2001