

XML-Praxis

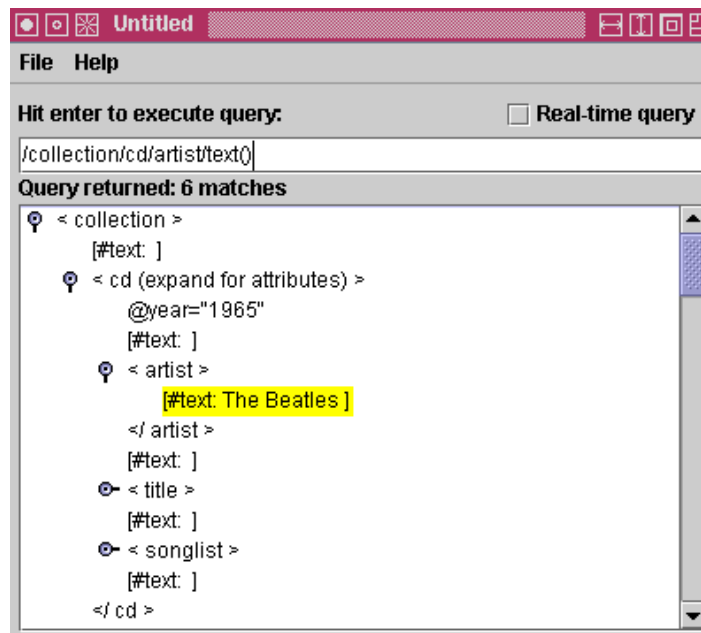
XSLT, Teil 1 – Übungen

Jörn Clausen

joern@TechFak.Uni-Bielefeld.DE

unsere CD-Sammlung ...

- Ausgangspunkte: `cd-collection.xml`, `cd-collection.dtd`
- `xpathtester` in Verzeichnis mit obigen Dateien starten
- `cd-collection.xml` laden



XPath-Ausdrücke ausprobieren

- erst überlegen, dann ausprobieren, dann erklären

```
/collection
```

```
/collection/cd
```

```
/collection/cd/artist/text()
```

```
/collection/cd/@year
```

XPath-Ausdrücke ausprobieren

- erst überlegen, dann ausprobieren, dann erklären

```
/collection
```

```
/collection/cd
```

```
/collection/cd/artist/text()
```

```
/collection/cd/@year
```

```
/collection/cd[@year="1973"]
```

```
/collection/cd[@year="1973"]/artist/text()
```

```
//song[position()=3]
```

```
//song[3]
```

```
//song[position()=last()]
```

XPath-Ausdrücke ausprobieren

- erst überlegen, dann ausprobieren, dann erklären

```
/collection
```

```
/collection/cd
```

```
/collection/cd/artist/text()
```

```
/collection/cd/@year
```

```
/collection/cd[@year="1973"]
```

```
/collection/cd[@year="1973"]/artist/text()
```

```
//song[position()=3]
```

```
//song[3]
```

```
//song[position()=last()]
```

```
count(/collection/cd)
```

```
/collection/cd[contains(artist,"Beatles")]/title
```

die CD-Sammlung in HTML

- rudimentäres Stylesheet in **cd2html1.xsl**

```
<xsl:template match="/collection">
  <html>
    <head>
      <title>CD-Collection</title>
    </head>
    <body>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>
```

- XSLT-Prozessor aufrufen und Ausgabe in Datei umlenken

```
$ xsltproc cd2html1.xsl cd-collection.xml > cd-collection.html
```

Song-Liste

- erzeuge *unordered list* () aus Song-Liste

Song-Liste

- erzeuge *unordered list* () aus Song-Liste
- templates für `songlist` und `song`

Song-Liste

- erzeuge *unordered list* () aus Song-Liste
- templates für songlist und song

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

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

Song-Liste

- erzeuge *unordered list* () aus Song-Liste
- templates für songlist und song

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

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

- [cd2html2.xsl](#)

Überschrift

- erzeuge Überschrift mit Künstler und Titel
`<h1>The Beatles / Help!</h1>`

Überschrift

- erzeuge Überschrift mit Künstler und Titel
`<h1>The Beatles / Help!</h1>`
- In welchem template?

Überschrift

- erzeuge Überschrift mit Künstler und Titel

```
<h1>The Beatles / Help!</h1>
```

- In welchem template?

```
<xsl:template match="cd">  
  <h1>  
    <xsl:value-of select="artist"/>  
    <xsl:text> / </xsl:text>  
    <xsl:value-of select="title"/>  
  </h1>  
  <xsl:apply-templates/>  
</xsl:template>
```

- [cd2html3.xsl](#)

Überschrift, cont.

- Problem?

Überschrift, cont.

- Problem?
- `<artist>` und `<title>` werden doppelt verarbeitet
- Lösungsmöglichkeiten?

Überschrift, cont.

- Problem?
- `<artist>` und `<title>` werden doppelt verarbeitet
- Lösungsmöglichkeiten?
- „leere“ templates

```
<xsl:template match="artist|title"/>
```


Überschrift, cont.

- Problem?
- `<artist>` und `<title>` werden doppelt verarbeitet
- Lösungsmöglichkeiten?
- „leere“ templates

```
<xsl:template match="artist|title"/>
```
- [cd2html4.xsl](#)
- andere Möglichkeit: template-Aufrufe steuern