# Programmiermodul 4
## XSLT

- **XSL-Transformation:**
  - XSL = eXtensible Stylesheet Language
    - Beschreibt wie XML-Elemente angezeigt werden
  - Transformation eines XML-Dokumentes (in bspw. wieder XML, HTML, ...)
  - Verwendet **XPath** um Information im XML-Dokument zu finden (match="...") (um durch dessen Elemente und Attribute zu navigieren)
  - Elemente/Attribute können hinzugefügt oder entfernt werden
  - **Sortierung**/Umordnung von Elementen möglich
  - Liegt üblicherweise als **XSL-Stylesheet** vor (Dateiendung **.xsl**)

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl=...>
    <xsl:template match=...>
    ...
    </xsl:template>
...
</xsl:stylesheet>
```

## Wozu XSTL im Feed-Reader?

- Alle Einträge werden aktuell direkt **im HTML-Dokument** einzeln eingetragen!
- Ermöglicht problemlos ein anderes Ausgabeformat für bspw. API's.
- Inhalt und Darstellung sollen voneinander **getrennt** sein.
  (Dies wird bereits zu einem großen Teil durch CSS erreicht.)
  - Wir wollen den Inhalt ändern, **ohne die Darstellung zu berühren**!

# Programmiermodul 4

## XSLT Beispiel: **example.xml**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
<feed xmlns="http://webis.de/xslt/example">
    <title>Best mysterious articles.</title>
    <entry>
        <id>1</id>
        <title>A mysterious article.</title>
        <text>Never released.</text>
    </entry>
    <entry>
        <id>2</id>
        <title>Another mysterious article.</title>
        <text>Never released as well.</text>
    </entry>
</feed>
```

# Programmiermodul 4

## XSLT Beispiel: **example.xsl**

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:example="http://webis.de/xslt/example">
    <xsl:output method="html" encoding="UTF-8"/>
    <xsl:template match="/">
        <html>
            <head>
                <meta charset="UTF-8"/>
                <title>XSLT Example</title>
            </head>
            <body>
                <div class="content">
                    <xsl:apply-templates select="example:feed"/>
                </div>
            </body>
        </html>
    </xsl:template>
    <xsl:template match="example:feed">
        <main>
            <h1><xsl:value-of select="example:title"/></h1>
            <xsl:apply-templates select="example:entry"/>
        </main>
    </xsl:template>
    <xsl:template match="example:entry">
        <article class="myentry">
            <xsl:attribute name="id">
                <xsl:value-of select="example:id"/>
            </xsl:attribute>
            <h2><xsl:value-of select="example:title"/></h2>
            <p><xsl:value-of select="example:text"/></p>
      </article>
    </xsl:template>
</xsl:stylesheet>
```

# Programmiermodul 4

## XSLT Beispiel: **Erzeugtes HTML-Dokument**

```
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta charset="UTF-8">
        <title>XSLT Example</title>
    </head>
    <body>
        <div class="content">
            <main>
                <h1>Best mysterious articles.</h1>
                <article class="myentry" id="1">
                    <h2>A mysterious article.</h2>
                    <p>Never released.</p>
                </article>
                <article class="myentry" id="2">
                    <h2>Another mysterious article.</h2>
                    <p>Never released as well.</p>
                </article>
            </main>
        </div>
    </body>
</html>
```

# Best mysterious articles.

## A mysterious article.

Never released.

## Another mysterious article.

Never released as well.

# Programmiermodul 4

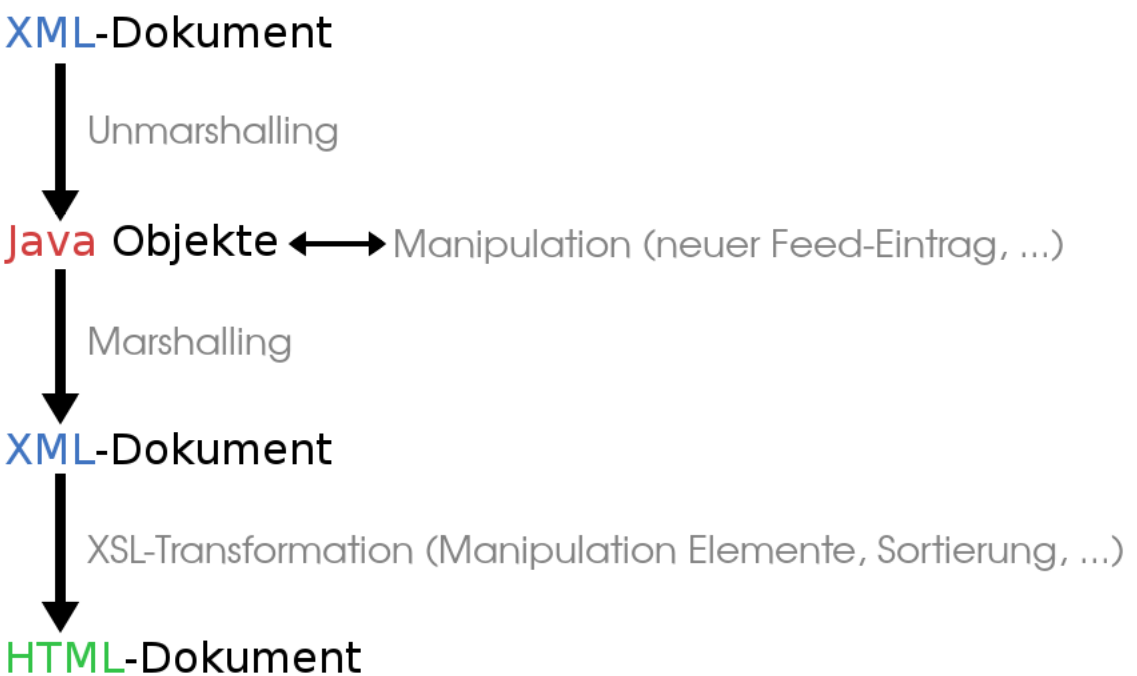
## Feed-Reader: **XSLT**

- **Aufgabe 4:**

# Programmiermodul 4

## Feed-Reader: **XSLT**

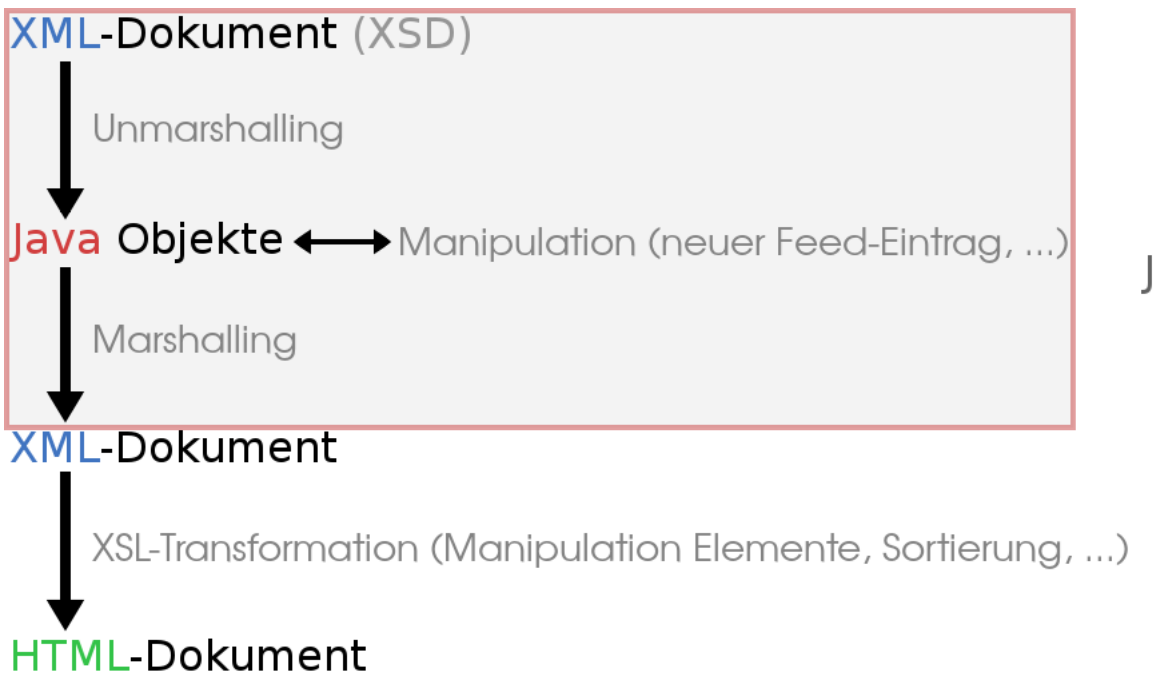
- **Aufgabe 6:**



# Programmiermodul 4

## Feed-Reader: **XSLT**

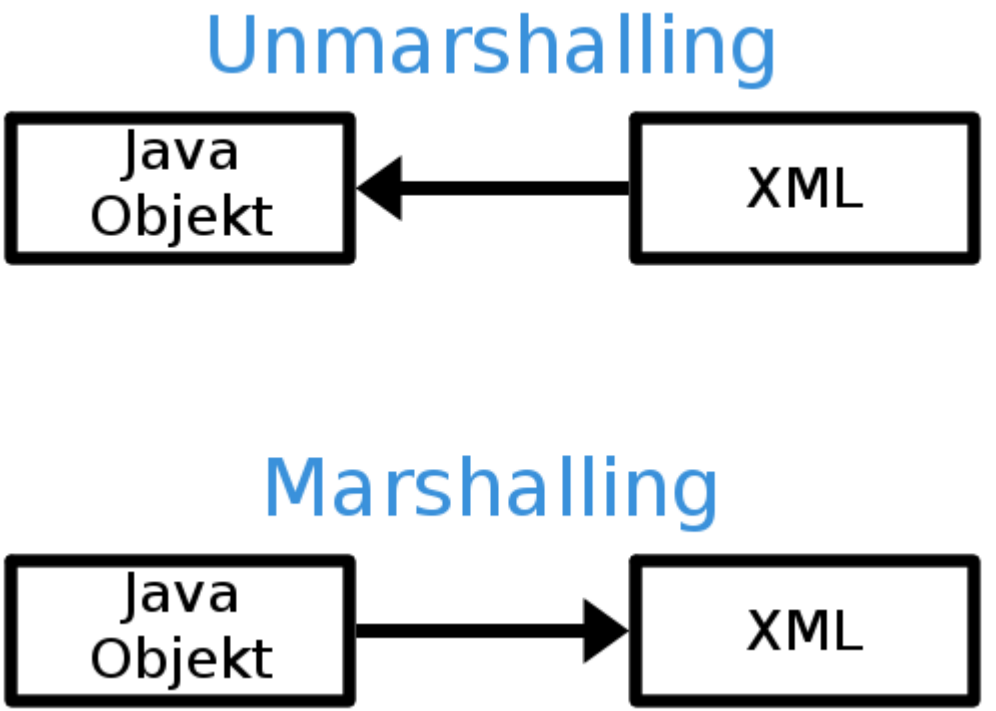
- **Aufgabe 6:**




**JAXB**

Java Architecture
for XML Binding

- **Marshalling und Unmarshalling:**



# Programmiermodul 4

## Feed-Reader: **XSLT**

- **Woher bekommen wir die Java-Klassen?**

  1. Möglichkeit:

     Manuelle Erstellung anhand des zugehörigen Schemas.
     $\rightarrow$ langwierig, fehleranfällig, ... $\rightarrow$ **schlecht**

# Programmiermodul 4

## Feed-Reader: **XSLT**

- **Woher bekommen wir die Java-Klassen?**

1. Möglichkeit:

   Manuelle Erstellung anhand des zugehörigen Schemas.
   → langwierig, fehleranfällig, ... → **schlecht**

2. Möglichkeit:

   XJC ("Compiles an XML schema file into fully annotated Java classes.")

   (a) `xjc -p jaxb atom.xsd`

   (b) Bis Java 8:

   ```
   javac jaxb/package-info.java
   ```

   Java 9, 10:

   ```
   javac --add-modules java.xml.bind jaxb/package-info.java
   ```

# Programmiermodul 4

## Feed-Reader: **JAXB im Code**

- **Marshaller:**

```
// Constant file reference to your XSL style sheet
private final static File XSL_SHEET = new File("xsl/atom-to-html.xsl");

...

private static Marshaller createMarshaller(Class type, Schema schema) {
    try {
        JAXBContext context = JAXBContext.newInstance(type);
        Marshaller marshaller = context.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
        marshaller.setProperty(Marshaller.JAXB_ENCODING, "UTF-8");
        marshaller.setSchema(schema);

        // Add the xml-stylesheet processing instruction
        String xslDeclaration = "<?xml-stylesheet type=\"text/xsl\" href=\""
            + XSL_SHEET.toString() + "\"?>";

        marshaller.setProperty("com.sun.xml.internal.bind.xmlHeaders",
            xslDeclaration);

        return marshaller;

    } catch (JAXBException ex) {
        throw new RuntimeException("Could not create Marshaller.", ex);
    }
}
```

# Programmiermodul 4

## Feed-Reader: **JAXB im Code**

- **Unmarshaller:**

```
// Constant file reference to the feed document
private final static File FEED_FILE = new File(feed.xml");

...

JAXBContext context = JAXBContext.newInstance(Feed.class);
Unmarshaller unmarshaller = context.createUnmarshaller();
unmarshaller.setSchema(schema); // validates against the schema as well

Feed serializedFeed = unmarshaller
    .unmarshal(new StreamSource(FEED_FILE), Feed.class)
    .getValue();

...
```