LINQ to XML

Proseminar *Objektorientiertes Programmieren mit .NET und C#*

Lucas Köhler

Institut für Informatik
Software & Systems Engineering

Motivation: Warum XML-Abfragen?

- Datenaustausch zwischen Programmen
  - Plattformunabhängig
  - Herstellerunabhängig
  - Softwareunabhängig
- Logische Strukturierung von Daten
- Automatische Datenverarbeitung
- Anwendungsbeispiele:
  - Windows Presentation Foundation (in Form von XAML)
  - Design von Android Apps
  - Konfigurationsdateien

Motivation: Warum LINQ to XML?

```
// mit LINQ:
XElement linqdoc = XElement.Load("Beispiel.xml");
IEnumerable<XElement> test =
    from el in linqdoc.Elements("Student")
    where (decimal)el.Element("Semester") == 3
    select el;


// mit dem W3C-DOM:
XmlDocument xmldoc = new XmlDocument();
xmldoc.Load("Beispiel.xml");
List<XmlElement> liste = new List<XmlElement>();
XmlElement root = xmldoc.DocumentElement;
foreach (XmlElement e in root.SelectNodes("Student")) {
    if (e.SelectSingleNode("Semester").InnerText == "3")
        liste.Add(e);
}
```

Agenda

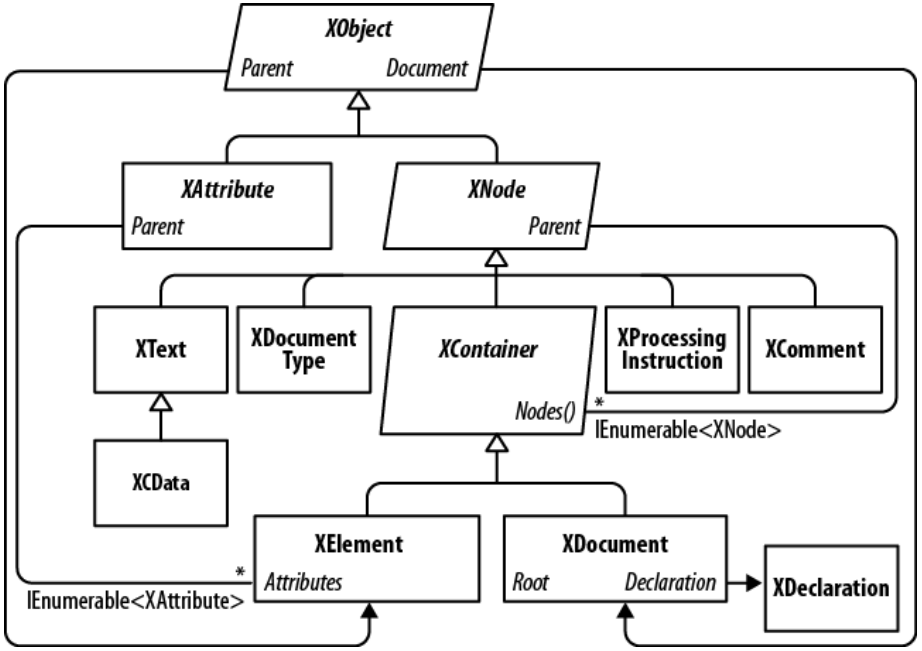
- Motivation
- Einführung in XML
- Grundlagen des X-DOM
- LINQ to XML
- Funktionale Konstruktion
- Zusammenfassung

Einführung in XML

- e**X**tensible **M**arkup **L**anguage
- Entwickelt vom World Wide Web Consortium (W3C)
- Textbasiertes Format
- Tags und Attribute
- Tags selbst definierbar
- Festlegen der Form eines XML-Dokuments mit XSD
- Strukturierung der Daten als Baum
- Dokument besteht aus:
  - Header
  - Daten

```
<?xml version="1.0" encoding="utf-8" ?>
<Studenten>
  <Student MatrikelNr="07654321">
    <Vorname>Maximilian</Vorname>
    <Nachname>Schmidt</Nachname>
    <Semester>3</Semester>
  </Student>
  <Student MatrikelNr="01234567">
    <Vorname>Lisa</Vorname>
    <Nachname>Müller</Nachname>
    <Semester>5</Semester>
  </Student>
</Studenten>
```

Das X-DOM

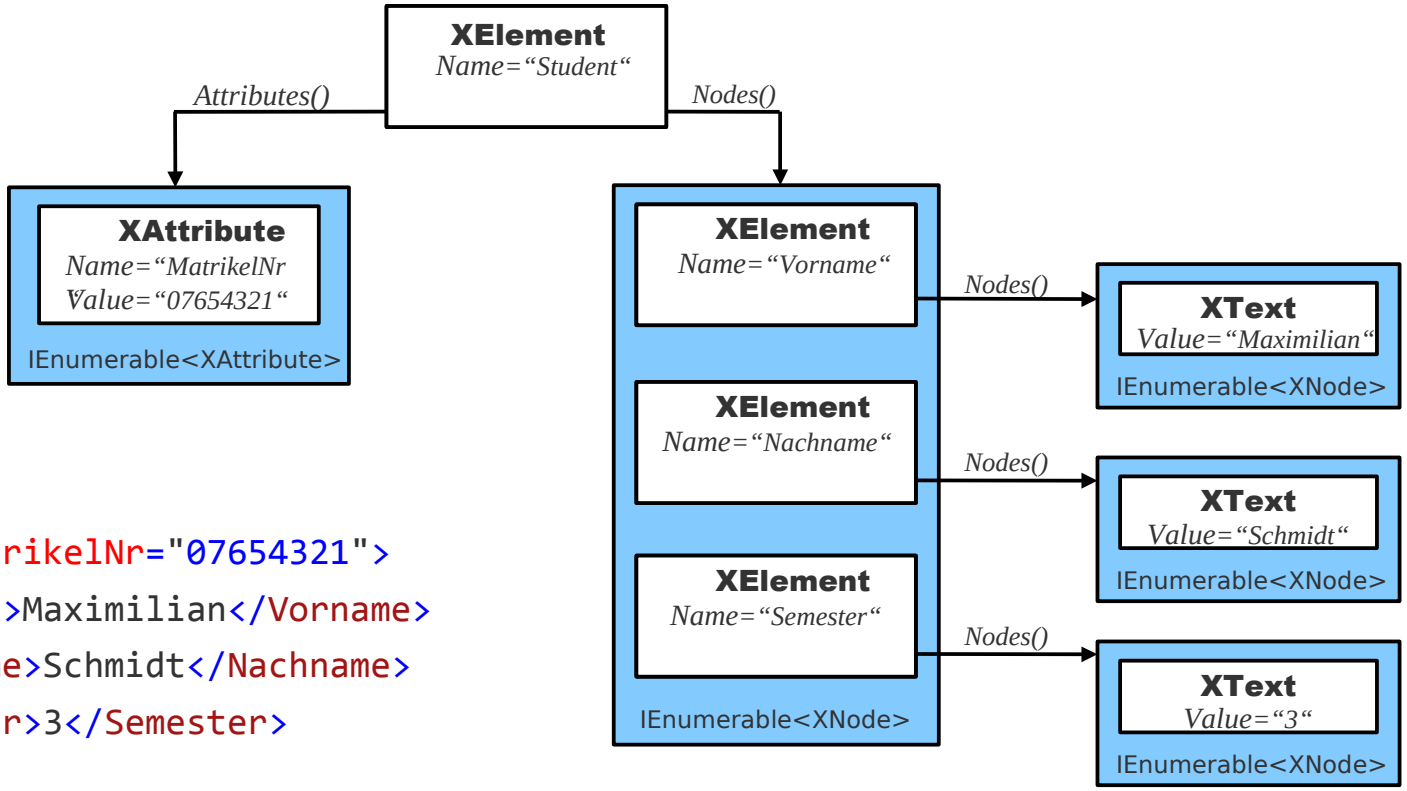


Klassenhierachie mit den wichtigsten Klassen des X-DOM.

Entnommen aus Pialorsi, 2007.

```
<?xml version="1.0" encoding="utf-8" ?>
<Studenten>
  <Student MatrikelNr="07654321">
    <Vorname>Maximilian</Vorname>
    <Nachname>Schmidt</Nachname>
    <Semester>3</Semester>
  </Student>
  <Student MatrikelNr="01234567">
    <Vorname>Lisa</Vorname>
    <Nachname>Müller</Nachname>
    <Semester>5</Semester>
  </Student>
</Studenten>
```

Grundlagen des X-DOM – Beispiel




```
<Student MatrikelNr="07654321">
    <Vorname>Maximilian</Vorname>
    <Nachname>Schmidt</Nachname>
    <Semester>3</Semester>
</Student>
```

Grundlagen des X-DOM

- Elemente und Attribute abrufen
  - Element(XName)
  - Elements
  - Descendants
  - Ascendants
  - Attribute(XName)
  - Attributes
- XML-Daten beschaffen
  - Load
  - Parse

```
<?xml version="1.0" encoding="utf-8" ?>
<Studenten>
  <Student MatrikelNr="07654321">
    <Vorname>Maximilian</Vorname>
    <Nachname>Schmidt</Nachname>
    <Semester>3</Semester>
  </Student>
  <Student MatrikelNr="01234567">
    <Vorname>Lisa</Vorname>
    <Nachname>Müller</Nachname>
    <Semester>5</Semester>
  </Student>
</Studenten>
```

LINQ to XML: Beispieldokument

```
<Studenten>
  <Student MatrNr="01234567" Alter="22">
    <Vorlesungen>
      <Vorlesung Credits="8" Name="Diskrete Strukturen"></Vorlesung>
      <Vorlesung Credits="3" Name="Informatikrecht"></Vorlesung>
      <Vorlesung Credits="5" Name="Info 2">
      </Vorlesung>
    </Vorlesungen>
  </Student>
  <Student MatrNr="01337042" Alter="19">
    <Vorlesungen>
      <Vorlesung Credits="8" Name="Diskrete Strukturen"></Vorlesung>
      <Vorlesung Credits="6" Name="Programmierpraktikum"></Vorlesung>
    </Vorlesungen>
  </Student>
  <Student MatrNr="08763312" Alter="20">
    <Vorlesungen>
      <Vorlesung Credits="5" Name="Info 2"></Vorlesung>
      <Vorlesung Credits="8" Name="Analysis"></Vorlesung>
      <Vorlesung Credits="6" Name="Datenbanken"></Vorlesung>
    </Vorlesungen>
  </Student>
</Studenten>
```

LINQ to XML: Abfrage 1

Ziel:

- Ausgabe aller Studenten, absteigend sortiert nach der Summe der Credits ihrer Vorlesungen.

```
01 XElement doc = XElement.Load("Studenten.xml");
02 IEnumerable<XElement> ergebnis =
03     from el in doc.Elements("Student")
04     orderby
05         (from e in el.Descendants("Vorlesung")
06         select (decimal)e.Attribute("Credits")).Sum()
07         descending
08     select el;
09 foreach(XElement e in ergebnis)
10     Console.WriteLine(e.ToString());
```

LINQ to XML: Abfrage 1 Ergebnis

```
<Student MatrNr="08763312" Alter="20">
  <Vorlesungen>
    <Vorlesung Credits="5" Name="Info 2"></Vorlesung>
    <Vorlesung Credits="8" Name="Analysis"></Vorlesung>
    <Vorlesung Credits="6" Name="Datenbanken"></Vorlesung>
  </Vorlesungen>
</Student>
<Student MatrNr="01234567" Alter="22">
  <Vorlesungen>
    <Vorlesung Credits="8" Name="Diskrete Strukturen"></Vorlesung>
    <Vorlesung Credits="3" Name="Informatikrecht"></Vorlesung>
    <Vorlesung Credits="5" Name="Info 2"></Vorlesung>
  </Vorlesungen>
</Student>
<Student MatrNr="01337042" Alter="19">
  <Vorlesungen>
    <Vorlesung Credits="8" Name="Diskrete Strukturen"></Vorlesung>
    <Vorlesung Credits="6" Name="Programmierpraktikum"></Vorlesung>
  </Vorlesungen>
</Student>
```

LINQ to XML: Abfrage 2

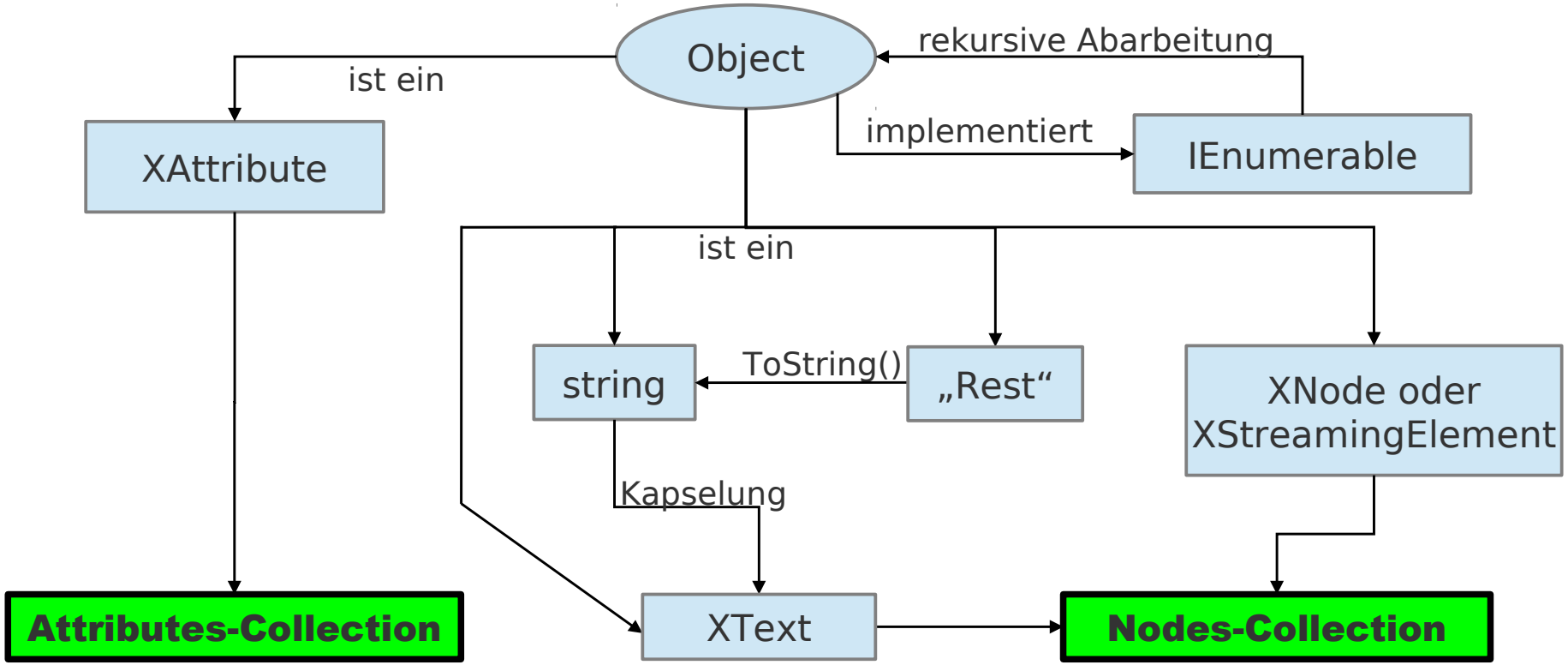
```
01 IEnumerable<XElement> ergebnis2 =
02     from el in doc.Elements("Student")
03     let a =
04             (
05              from v2 in doc.Descendants("Vorlesung")
06              select (decimal)v2.Attribute("Credits")
07              ).Min()
08     where
09         (from e in el.Descendants("Vorlesung")
10         select (decimal)e.Attribute("Credits")
11         ).Min() == a
12     select el;
13 foreach(XElement e in ergebnis2)
14     Console.WriteLine(e.ToString());
```

LINQ to XML: Abfrage 2 Ergebnis

```
<Student MatrNr="01234567" Alter="22">
  <Vorlesungen>
    <Vorlesung Credits="8" Name="Diskrete Strukturen"></Vorlesung>
    <Vorlesung Credits="3" Name="Informatikrecht"></Vorlesung>
    <Vorlesung Credits="5" Name="Info 2"></Vorlesung>
  </Vorlesungen>
</Student>
```

Funktionale Konstruktion

- Einfache Erzeugung neuer XML-Strukturen
- Ermöglicht durch überladenen Konstruktor von XElement
- Direktes Einbetten von LINQ to XML Abfragen in den Konstruktoraufruf



Funktionale Konstruktion + LINQ to XML: Beispiel

```
01 XElement ergebnis3 = new XElement("Studenten",
02     from stu in doc.Elements("Student")
03     orderby (decimal)stu.Attribute("Alter")
04     select
05         new XElement("Student",
06             new XElement("Alter", stu.Attribute("Alter").Value),
07             new XElement("MatrNr", stu.Attribute("MatrNr").Value),
08             new XElement("Vorlesungsanzahl",
09                 (from vor in stu.Descendants("Vorlesung")
10                 select vor).Count()
11             )
12         )
13 );
```

Funktionale Konstruktion + LINQ to XML: Beispiel

```
<Studenten>
  <Student>
    <Alter>19</Alter>
    <MatrNr>01337042</MatrNr>
    <Vorlesungsanzahl>2</Vorlesungsanzahl>
  </Student>
  <Student>
    <Alter>20</Alter>
    <MatrNr>08763312</MatrNr>
    <Vorlesungsanzahl>3</Vorlesungsanzahl>
  </Student>
  <Student>
    <Alter>22</Alter>
    <MatrNr>01234567</MatrNr>
    <Vorlesungsanzahl>3</Vorlesungsanzahl>
  </Student>
</Studenten>
```

Zusammenfassung

- XML: flexibles und weitverbreitetes Format zum Datenaustausch
- X-DOM: neues LINQ-freundliches DOM
- XElement und XAttributes als zentrale Klassen für LINQ to XML-Abfragen
- Einfaches Abfragen von XML-Strukturen mit bekannter LINQ-Syntax
- Einfache Umstrukturierung bestehender XML-Strukturen mit Hilfe von funktionaler Konstruktion und LINQ to XML

Quellen

- Andreas Kühnel. Visual C# 2010: Das umfassende Handbuch. Galileo Press, Bonn, 5 edition, 2010.
- Holger Schwichtenberg. Microsoft .NET 4.0 -Crashkurs: [der Schnelleinstieg für Umsteiger von Java, C++, Visual Basic 6, Delphi & Co]. Microsoft Press, [Unterschleißheim], 1 edition, 2011.
- Joseph Albahari and Ben Albahari. C# 5.0 in a Nutshell: The Denitive Reference. O'Reilly & Associates, Sebastopol and CA, 1 edition, 2012.
- Paolo Pialorsi and Marco Russo. Microsoft LINQ-Crashkurs: [Ihr Schnelleinstieg in neue Technologien und Tools]. Microsoft Press, Unterschleissheim, 2007.
- MSDN Library
- http://www.w3.org/standards/xml