LibXML2 - Tutorium

Helge Löding

hloeding@informatik.uni-bremen.de, hloeding@verified.de

Universität Bremen, Verified Systems Intl.

Spezifikation eingebetteter Systeme
2011-01-31

XML

model.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<model>
  <class name="C">
    <attribute name="a" type="int"/>
    <attribute name="b" type="float"/>
    <method name="f">
      <parameter name="x" type="int"/>
      <parameter name="y" type="int"/>
      <return type="int"/>
      <body>return x * y;</body>
    </method>
  </class>
</model>
```

- Erweiterbare Auszeichnungssprache (*EXtensible Markup Language*)
- Textuelle Darstellung von Daten
- Daten annotiert (*marked up*) durch Dokumentstruktur
- Hierarchische Dokumentstruktur (Baumstruktur)
- Namen der Strukturelemente frei wählbar

XML - Begriffe

model.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<model>
  <class name="C">
    <attribute name="a" type="int"/>
    <attribute name="b" type="float"/>
    <method name="f">
      <parameter name="x" type="int"/>
      <parameter name="y" type="int"/>
      <return type="int"/>
      <body>return x * y;</body>
    </method>
  </class>
</model>
```

- Elemente: Knoten des Baums
  - Ein Start- und End-Tag
    z.B.: <method>, </method>
  - Kurz für leere Elemente
    z.B.: <return .../>
- Attribute: Zusatzinformationen
  - Erweiterung von Start-Tags
    z.B.: name = "a"
- Textdaten
  - Inhalt zwischen Start-, End-Tag
    z.B.: `return x * y;`

XML - Struktur

model.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<model>
  <class name="C">
    <attribute name="a" type="int" />
    <attribute name="b" type="float" />
    <method name="f">
      <parameter name="x" type="int" />
      <parameter name="y" type="int" />
      <return type="int" />
      <body>return x * y;</body>
    </method>
  </class>
</model>
```

- Genau ein Wurzelelement (hier: model)
- Jedes Element hat genau ein Start- und End-Tag
- Elemente durch Start- und End-Tags *sauber* verschachtelt
- Pro Element keine zwei Attribute mit dem selben Namen

LibXML2 - Wichtige Typen und Datenstrukturen

- `xmlDoc`: Struktur für XML-Dokumente
  - `xmlDoc::children`: Wurzel-Element
- `xmlNode`: Struktur für XML-Elemente
  - `xmlNode::name`: Name des Elements
  - `xmlNode::children`: Erstes Kind-Element
  - `xmlNode::next`: Nächstes Element
  - `xmlNode::prev`: Voriges Element
  - `xmlNode::properties`: Element-Attribute
  - `xmlNode::content`: Textinhalt
- `xmlChar`: LibXML2 Charactertyp
  - Zwischen `xmlChar*` und `char*` kann i.A. gecastet werden

LibXML2 - Wichtige Funktionen

- `xmlDocPtr xmlReadFile(const char *filename, ...)`
  - Parsieren einer XML-Datei in einen Dokumentbaum
- `xmlNodePtr xmlDocGetRootElement(xmlDocPtr doc)`
  - Auslesen des Wurzelelements eines Dokumentbams
- `void xmlFreeDoc(xmlDocPtr doc)`
  - Freigeben eines Dokumentbaums
- `xmlChar *xmlGetProp(xmlNodePtr n, const xmlChar *a)`
  - Auslesen eines Attributs eines Elements
- `xmlChar *xmlNodeGetContent(xmlNodePtr node)`
  - Auslesen des Texts eines Elements
- `void xmlFree(xmlChar *buf)`
  - Freigeben eines xmlChar Arrays (String)

Beispiel 1

- Parsieren der XML-Datei

Beispiel 2

- Parsieren der XML-Datei
- Rekursiv alle Elementnamen ausgeben

Beispiel 3

- Parsieren der XML-Datei
- Ausgabe der Dateiinhalte
  - Elementnamen
  - Attributnamen und -werte
  - Elementtext

Beispiel 4

- Parsieren der XML-Datei
- Codegenerierung nach C++

Beispiel 5

- Parsieren der XML-Datei
- Codegenerierung nach C++
  - Visitor-Pattern

Vermischtes

- www.xmlsoft.org
- Linker-Flags und Pakete
- Flex/Bison