

## 19 Bonus-Kapitel: Vom Server laden

*Kombinieren Sie die Stärken von Flash mit den Stärken serverseitiger Programmierung. Flash hat dabei die Rolle Information darzustellen. Das Programm am Server kann Daten auf Dauer speichern, die Kommunikation mit anderen Usern herstellen, etc.*

Betrachten wir noch einmal die Kommunikation zwischen Client und Server im Web:

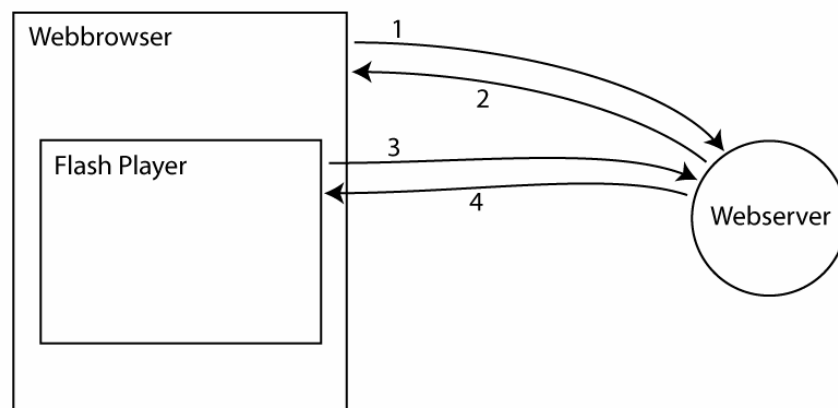


Abbildung 118: Kommunikation zwischen Webbrowser, Webserver und Flash Player, Webserver

Der Webbrowser stellt eine Anfrage an den Webserver (Pfeil 1 in Abbildung 118), der Webserver antwortet in dem er z.B. HTML-Daten sendet (Pfeil 2). Der Browser liest die HTML-Daten und stellt sie dar. Wenn ein **object** Tag enthalten ist, der auf eine swf-Datei verweist dann stellt der Webbrowser eine weitere Anfragen an den Server (Pfeil 1) um die swf-Datei zu laden. Der Server sendet die swf-Datei (Pfeil 2).

Nun startet der Browser den Flash Player als Plugin, und übergibt ihm die Kontrolle über einen Bereich des Browser-Fensters. Der Flash Player erhält die swf-Datei und stellt diese dar. So weit, so bekannt.

In der swf-Datei können nun Actionscript-Befehle enthalten sein, die den Flash Player anweisen eine Anfrage an den Server zu schicken (Pfeil 3). Wenn die Antwort des Server einlangt (Pfeil 4) wird in der swf-Datei eine **onLoad** Funktion aufgerufen.

### 19.1 Einfache Daten laden

Beispiel 21) Vom Server werden aktuelle Daten über den Stundenplan angefordert, und in Flash dargestellt.

Wir verwenden ein PHP-Programm auf dem Server **multimediaart.at** das Informationen aus dem Stundenplan der FH liefert.

---

```
http://---- geheime url ----/next.php?username=bjelline
```

---

Unter dieser URL werden die Daten der nächsten Lehrveranstaltung ausgegeben. ~~Probieren Sie die URL im Browser aus, auch mit verschiedenen Usernamen!~~

Die Ausgabe des Programms ist nicht für die Anzeige im Browser gedacht, sondern für die Weiterverarbeitung in Flash. Der Output sieht so aus:

```
von=17:15&bis=19:30&typ=WK&titel=Webprojekt&raum=LB 313
```

Das kaufmännische Und „&“ als Trennzeichen macht diesen Output schwer lesbar, so sieht der Output ohne diese Zeichen aus:

```
von=17:15 bis=19:30 typ=WK  
titel=Webprojekt raum=LB 313
```

Mit dem (veralteten) Actionscript-Befehl **loadVars** oder mit dem (moderneren) Objekt **loadVars** kann man diese Daten in Flash lesen.

So sieht ein Flash-Programm aus, das diese Daten abholt und liest:

---

```
1 url = "http://multimediaart.at/plan/next.php?username=bjelline";  
2 l = new LoadVars();  
3 l.onLoad = function(success) {  
4     if (success) {  
5         trace("daten sind da: " + this.titel);  
6     } else {  
7         trace("fehler beim Laden von " + url);  
8     }  
9 }  
10 l.load(url);  
11 trace("...gleich geht's weiter");
```

---

Zeile eins bis neun sind nur die Vorbereitung, erst Zeile 10 startet den asynchronen Lade-Vorgang vom Webserver.

Wenn dann viele Millisekunden später die Daten vom Webserver angelangen wird die `onLoad`-Funktion des `LoadVars`-Objekts (die in Zeile 3 bis 9 definiert wurde) gestartet.

Innerhalb dieser `onLoad`-Funktion sind die Daten vom Webserver als Eigenschaften von `this` lesbar:

```
this.von, this.bis, this.typ, this.titel,...
```

**Übung 66)** Setzen Sie die Anzeige der nächsten Lehrveranstaltung in Flash um. Die Bühne sollte klein sein, damit sie diesen Flash-Film als keines Widget in eine vorhandene HTML-Seite einbinden können.

## 19.2 XML für Slideshow

Beispiel 22) Ein PHP-Programm liefert eine Liste der am Server vorhandenen Bilder als XML-Daten an Flash. Flash stellt die Bilder in einer Slideshow dar.

Das PHP-Programm wird als Datei mit der Endung `.php` in den Ordner mit den Bildern gespeichert:

```
<?php
    header("Content-Type: application/xml");

    $bilder = glob("*.jpg");

    echo("<bilder>\n");
    foreach( $bilder as $bild ) {
        echo("<bild imgurl='$bild' />\n");
    }
    echo("</bilder>\n");
?>
```

Der Output des PHP-Programmes sind XML-Daten:

```
<bilder>
  <bild imageurl="img/DSC_3461.jpg" />
  <bild imageurl="img/DSC_3462.jpg" />
</bilder>
```

Testen Sie zuerst das PHP-Programm alleine mit dem Browser: der Browser kann den XML-Output des Pro-

## Bonus-Kapitel: Vom Server laden

grammes gut darstellen. Wenn Sie ein zusätzliches Bild in den Ordner laden sollte es beim nächsten Aufruf im XML erscheinen.

Mit dieser XML-Datei sind anscheinend keine Style-Informationen verknüpft.

```
- <bilder>  
  <bild imgurl="das.jpg"/>  
  <bild imgurl="dies.jpg"/>  
  <bild imgurl="towerbridgeandcorncob.jpg"/>  
</bilder>
```

Abbildung 119: Darstellung von XML-Daten im Webbrowser Firefox

Das Laden der XML-Daten erfolgt wieder asynchron, mit dem XML-Objekt:

```
var xml:XML = new XML();  
  
// Leerzeichen und Formatierungen in XML  
// sollen ignoriert werden:  
xml.ignoreWhite = true;  
  
// wenn Daten fertig geladen wurden  
// rufe die Funktion XMLgeladen auf  
xml.onLoad = xmlGeladen;
```

Die Struktur der XML-Tags wird in Flash als Baum von Objekten dargestellt, das XML-Objekt ist dabei die Wurzel des Baumes.

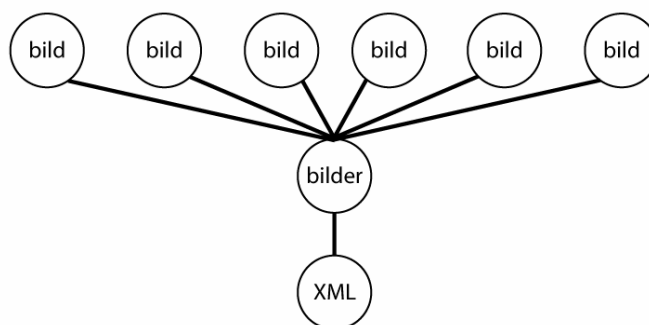


Abbildung 120: Darstellung des XML der Slideshow als Baum

In Javascript wird dasselbe Prinzip für die Darstellung des HTML-Dokuments als Baum von Objekten verwendet, Sie kennen es dort unter dem Namen Document Object Model oder kurz DOM.

Auch die Befehle, Funktionen, Eigenschaften zur Manipulation der DOM sind in Actionscript gleich wie in Javascript:

Mit den Eigenschaften `firstChild` und `childNodes` kann man in diesem Baum navigieren:

`xml.firstChild` entspricht dem `bilder` Tag.

`xml.firstChild.firstChild` entspricht dem ersten `bild` Tag.

`xml.firstChild.childNodes` ist ein Array das alle `bild` Tags enthält, `xml.firstChild.childNodes[0]` entspricht also wieder dem ersten `bild` Tag.

Jedes Objekt im Baum enthält ein Objekt `attributes` das die Namen und Werte der Attribute speichert:

`xml.firstChild.firstChild.attributes.imgurl` liefert den Wert des `imgurl`-Attributes des ersten Bildes.

Die Funktion `xmlGeladen` liest nun die Daten aus dem XML-Baum und schreibt die Namen der Bilder in ein Array namens `bilderListe`; dieses Array wird in weitere Folge von der Funktion `zeigeBilder` verwendet um die Slideshow zu starten.

---

```
function xmlGeladen() {
    var xnode:XMLNode;
    // hier wird ausgelesen wie viele nodes sich im XML befinden.
    anzahlBilder = xml.firstChild.childNodes.length;
    for(var i:Number = 0; i < anzahlBilder; i++) {
        // Bild nummer i
        xnode = xml.firstChild.childNodes[i];
        bilderListe.push(xnode.attributes.imgurl);
    }
    // Darstellung der Slideshow starten
    zeigeBilder();
}
```

---

### 19.3 Komplexe XML-Daten laden

Für größere Datenmengen, und für Daten mit einer komplizierten Struktur werden die XML-Daten entsprechende komplexer.

Beispiel 23) Der Stundenplan einer ganzen Woche soll angezeigt werden

Um die Daten von so vielen Tagen und Lehrveranstaltungen zu übertragen werden unter der URL

---

`http://---geheime-url----/next.php?username=bjelline`

---

die XML-Tags **woche**, **plan** und **termin** verwendet. Diese Tags wurden extra für diese Anwendung erfunden.

Der Tag **woche** hat ein Attribut **datum** das den Montag der Woche entspricht und ein Attribut **uname**.

Der Tag **plan** entspricht einem einzelnen Wochentag. Falls an diesem Tag Lehrveranstaltungen stattfinden, dann werden diese als **termin** Tags dargestellt.

Die Attribute des **termin** Tags sind **datum**, **uname**, **von**, **bis**, **typ**, **raum**, **gr**, **titel** und **subtitel**. Der hier gezeigt Code ist verkürzt, weil er nur Montag bis Mittwoch der Woche zeigt.

---

```
<!-- Zugriff auf die Stundenplaene der FH Salzburg in XML -->
<!-- Sun Dec 16 00:00:00 2007 ist der vorige Sonntag -->
<!-- Mon Dec 17 01:00:00 2007 ist der Montag -->
<woche datum="2007-12-17" uname="mma4" next="2007-12-24" prev="2007-12-10">
  <plan uname="mma4" datum="2007-12-17">
  </plan>
  <plan uname="mma4" datum="2007-12-18">
    <termin datum="2007-12-18" uname="" von="09:00:00" bis="18:30:00"
      typ="UB" raum="U LB 315" gr="B + A" titel="Video / Authoring 2"
      subtitel="" />
  </plan>
  <plan uname="mma4" datum="2007-12-19">
    <termin datum="2007-12-19" uname="" von="09:00:00" bis="18:30:00"
      typ="UB" raum="U LB 352 + U LB 314" gr="A + B"
      titel="Bewegtbilddesign" subtitel="" />
    <termin datum="2007-12-19" uname="" von="10:00:00" bis="16:45:00"
      typ="UB" raum="U LB 313" gr="A + B" titel="3D Advanced 2"
      subtitel="" />
  </plan>
</woche>
<!-- Das wars auch schon -->
```

---

Abbildung 121 zeigt die XML Struktur als Baum.

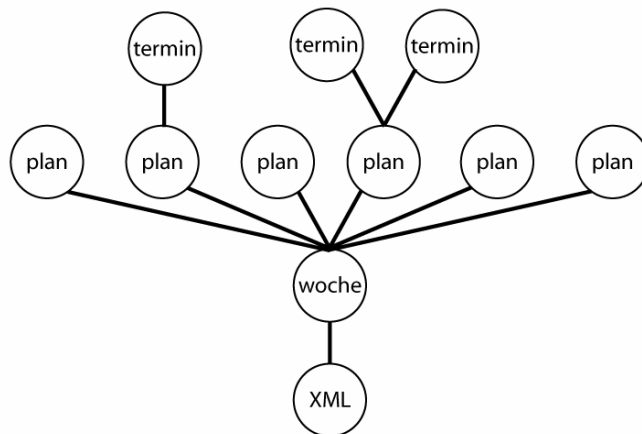


Abbildung 121: Darstellung des XML des Stundenplans als Baum

Die folgende `onLoad`-Funktion analysiert die XML-Daten und legt für jeden Termin ein neues Textfeld an. Die Details der Positionierung und Formatierung der Textfelder sind in einer (nicht gezeigten) Funktion `neufeld` verborgen.

```
xml.onload = function(ok)
{
  if (not (ok)) {
    output_txt.text = "laden funktioniert nicht!";
    return;
  }
  woche = this.firstChild;
  for (tagnr = 0; tagnr < woche.childNodes.length; tagnr++) {
    plan = woche.childNodes[tagnr];
    for (tnr = 0; tnr < plan.childNodes.length; tnr++) {
      termin = plan.childNodes[tnr];
      t = neufeld(tagnr, tnr, termin.attributes.von,
        termin.attributes.bis);
      t.text = von + "-" + bis + ": " + termin.attributes.titel;
    }
  }
  output_txt.text = "fertig geladen";
};
```