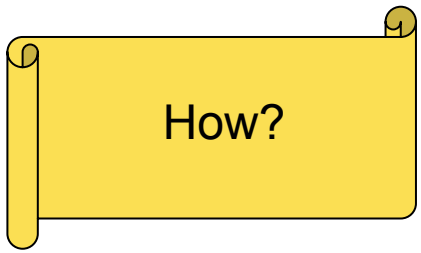
Displaying XML Documents

- XML documents can be displayed within a browser
  - using Cascading Stylesheets (CSS)
  - using data binding with the Data Source Object (DSO)
  - using the Document Object Model (DOM)
  - using Extensible Stylesheet Language Transformations (XSLT)
- XML documents can be displayed within a certain program
  - Java, C#, PHP?



A – Cascading Style Sheets

- are the most simple way to display XML documents
  - most web designers know how to create and work with CSS files
  - current web browsers support CSS to a high degree
- are less flexible than other technologies
  - the content of the elements cannot be modified or re-structured
  - attributes, unparsed entities and processing directives cannot be accessed

- CSS instructions can be stored in separate stylesheet files or be part of the XML document itself

Using Stylesheets

- including separate CSS files:
  - `<?xml-stylesheet type="text/css" href="`*filename*`"?>`
- CSS instructions within the XML file:
  - `<`*element_name* `STYLE='`*instruction1; instruction2;*`'>`
    content`</element_name>`

Resources

- Herold, H. (2002): Das HTML / XHTML – Buch. Suse Press.
- Laborenz, K. (2006): CSS-Praxis. Galileo Press.
- Lubkowitz, M. (2006): Webseiten programmieren und gestalten. Galileo Press.
- Shafer, D. / Yank, K. (2006): Cascading Stylesheets. Dpunkt Publishing.
- CSS specifications:
  - http://www.w3.org/tr/rec-css1
  - http://www.w3.org/tr/rec-css2

B – Data Binding

- XML documents are integrated with (X)HTML documents and (X)HTML standard elements (like <SPAN> or <TABLE> are bound to XML elements
- the (X)HTML elements then display the content of the corresponding XML elements
- data binding can only be used with symmetrical structured XML documents (using a DTD to ensure the file has the required structure is strongly recommended)
  - that means: a document has a root element which contains a certain amount of elements of the same type (the data sets) which in turn contain a certain amount of elements with character data (the fields) – i.e. a library of books with each book consisting of an author and a title
- data binding is not supported by all current browsers
  - MS Internet Explorer: 5.0 or higher / Mozilla Firefox: not supported

Integrating The Documents (I)

- XML and HTML documents are integrated via so-called data islands
- example for a fictional file "library.xml":
  - ```
    <html>
     <head>
      <title> … </title>
     </head>
     <body>
      <xml id="dsolibrary" src="library.xml"></xml>
      <!-- more HTML elements -->
     </body>
    </html>
    ```
- the browser checks the well-formedness / validity of the XML file
    ```

Integrating The Documents (II)

- it is also possible to include the whole XML root element with its content in the HTML file, but
  - this is not quite the XML philosophy, as data and their representation are no longer separated
  - maintenance of the XML document is more lavish, especially if the XML document is displayed in more than one HTML documents

Binding The Elements

- there are two binding methods:
  - table data binding – a <TABLE> element is bound to the XML data and displays all data sets of the XML document
  - data set binding – non-table-elements, e.g. <SPAN>, are bound to XML elements displaying only one element any one time

Table Data Binding (I)

- example: library.xml
  -
```
<body>
 <xml id="dsolibrary" src="library.xml"></xml>
 <table datasrc="#dsolibrary">
  <thead> … </thead>
  <tr>
   <td><span datafld="element_name1"></span></td>
   <td><span datafld="element_name2"></span></td>
   …
  </tr>
 </table>
</body>
```

Table Data Binding (II)

- although only one table row is defined, the browser repeats that row for all available data sets from the XML document
- <SPAN> is used because <TD> is not able to bind XML elements

Grouping Data Sets

- data sets can be grouped in order to create several pages from one XML document
- an id must be specified
- the attribute `datapagesize` defines the number of data sets per page
  - `<table id="books" datasrc="#dsolibrary" datapagesize="6">`
- predefined methods for navigation can be implemented (e.g. by using buttons)
  - `<button onclick="books.nextPage()">next page</button>`
  - if the corresponding page does not exist, clicking the button is ignored by the browser

Navigation Methods

method	result	example
firstPage	shows the first page with data sets	`books.firstPage()`
previousPage	shows the previous page with data sets	`books.previousPage()`
nextPage	shows the next page with data sets	`books.nextPage()`
lastPage	shows the last page with data sets	`books.lastPage`

Data Set Binding

- only one data set is displayed
  - ```
    <body>
     <xml id="dsolibrary" src="library.xml"></xml>

     <br />

    </body>
    ```
- attributes can be bound the same way, just replace *element_name* by *attribute_name*
  - if the attribute's name matches the name of another element, use !*element_name* and *attribute_name* instead
- the data source object provides methods to search through the data sets
    ```

Search Methods

method	current data set changes to	example
moveFirst	the document's first data set	`dsolibrary.recordset.moveFirst()`
movePrevious	the previous data set	`dsolibrary.recordset.movePrevious()`
moveNext	the next data set	`dsolibrary.recordset.moveNext()`
moveLast	the document's last data set	`dsolibrary.recordset.moveLast()`
move	the data set with the specified number	`dsolibrary.recordset.move(5)`

Resources

- for a list of all binding-capable HTML elements see:
  - Young, M. (2002): XML – Schritt für Schritt. Microsoft Press. (p.339ff.)

C – Document Object Model

- the DOM works with all kinds of well-formed XML documents, it doesn't expect a certain structure
- all components are accessible (even attributes, processing directives, notations and comments) – they are represented by a structured set of nodes
- implementations:
  - client side: MS Internet Explorer / Mozilla Firefox with partly different approaches, syntax and functionality
  - server side: e.g. PHP5

Node Types (I)

Node Type	XML component	nodeName	nodeValue
document	root element	*#document*	*null*
element	element	element name	*null*
text	text within the super-ordinated component	*#text*	text of the super-ordinated component
attribute	attribute	attribute name	attribute value
processing directive	processing directive	target of the directive	the content of the directive (w/o target)

Node Types (II)

Node Type	XML component	nodeName	nodeValue
comment	comment	*#comment*	the whole comment
CDATA section	CDATA section	*#cdata-section*	the content of the CDATA section
document type	the document type declaration	root element as used in DOCTYPE declaration	*#null*
entity	an entity (declared within the DTD)	entity name	*#null*
notation	a notation (declared within the DTD)	notation name	*#null*

Using The DOM

- all nodes contain properties and methods that can be used for navigating through and working with the corresponding XML document
  - if a property does not exist for a certain node, it returns *null*
- to access an XML file using the Document Object Model the file must be bound to an (X)HTML document (shown once again for the fictional "library.xml"):
  - `<xml id="dsolibrary" src="library.xml"></xml>`
- the actual implementation depends on the used technology (e.g. script languages like JavaScript, JScript, PHP or PERL)
  - an example using JScript can be found in files_06.zip; it only works with Internet Explorer 6.0 or higher (ActiveX is needed)

Resources

- Carl, D. (2006): Praxiswissen Ajax. O'Reilly.
- Krause, J. (2005): PHP 5 – Grundlagen und Profiwissen. Hanser.
- Shea, D. / Keith, J. (2005): DOM Scripting. APress.

- DOM level 1 specification:
  - http://www.w3.org/tr/rec-dom-level-1
- a list of node objects, properties and methods including the implementation status in common browsers can be found here:
  - http://www.w3schools.com/dom/dom_node.asp