INTRODUCTION INTO SECOND GENERATION WEB APPLICATIONS APPLYING XML

Guenter Pomaska

University of Applied Sciences Bielefeld, Faculty of Architecture and Civil Engineering,
Laboratory for Visual and Virtual Reality, Artilleriestr. 9, D-32427 Minden
gp@imagefact.de

Commission VII/2

KEY WORDS: Mark-up Language XML, Scalable Vector Graphics, Extensible 3D, PHP

ABSTRACT:

Second generation Web applications, are based on Extensible Mark-up Language (XML) and related technologies. HTML based applications are directed towards publication. In future, XML will be focus on structured information storage, interaction and distributed processing of data.

XML is a meta language for structuring data. XML uses tags, keywords in angle brackets, and attributes, followed by values embedded in double quotation. Comparing to HTML, the meaning of the tags and attributes is not defined in XML. Interpretation follows the application. XML files are plain text files, readable with simple text editors. Applying XSL (Extensible Style Sheet Language) provides translation into formats for Web browsers or print media.

SVG (scalable vector graphics) stands as a Web standard for two-dimensional graphics, formulated in XML. Vector graphics need less storage memory and display as enlargement better quality as raster graphics can do.

VRML (Virtual Reality Modelling Language), the former Web standard for three-dimensional graphics, is now redeemed by the standard document type definition for extensible 3D (X3D).

Members of the XML family of languages will become in a short time period substantial impact in the field of E-learning. XML documents can be downloaded from a server and published on any client sided platform. Dynamic Web applications provide documents upon user requests with access to a XML database. Software vendors take advantage from XML files because of the powerful support by object oriented programming languages, like Java or PHP. In addition, the open standard of XML makes it an ideal tool for E-learning communication.

1. STATIC AND DYNAMIC WEB SITES

1.1 Client-Server Architecture

Applying a client-server architecture does not require a computer network configuration. One can install a Web server and a Web browser on the same computer. The client (here the browser) sends a request to the server, addressed by *localhost*. The server is looking for the requested document, transfers the document to the client and the browser is rendering the file information into a readable format. The language, a browser understands, is HTML Hypertext Mark-up Language. Communication between server and client is agreed via a protocol, the Hypertext Transfer Protocol. The process is known under the term static Web application. Major software components of the Web are the Web browser, the http-protocol, the description language HTML and the Web server. static Web publication. Some of the software enhancements are mentioned later in this contribution.

1.2 Client Sided Dynamic

While a Web site is changing its appearance or content by user interaction without loading a new document, we talk about client sided dynamic. The document object model DOM enables access to any object (element) and its attributes, that is included in a Web document. The DOM is implemented in JavaScript, an extension to Web browsers. We focus here to a small application calculating control points by spatial intersection.

JavaScript has a hierarchical structure of application objects. Under the document level there are form objects existing. A form itself, is followed by elements. Forms and elements can be identified by names. If we want to set the value of one of the input fields in the sample shown in figure 1, we have to code *document.vws.xctrlpt.value = "100.00"*; *vws* stands for the name of the form, *xctrlpt* is the name of an input field, value is the attribut for the content. The HTML coding, applying the input-tag, reads as follows:

```
<form  name="vws">
<input name="xctrlpt" value=" "
       type ="text" >
</input>
</form>
```

Inside the angle brackets more attribute definitions can occur. This short explanation will help understanding the data exchange between a Web site and PHP programs.

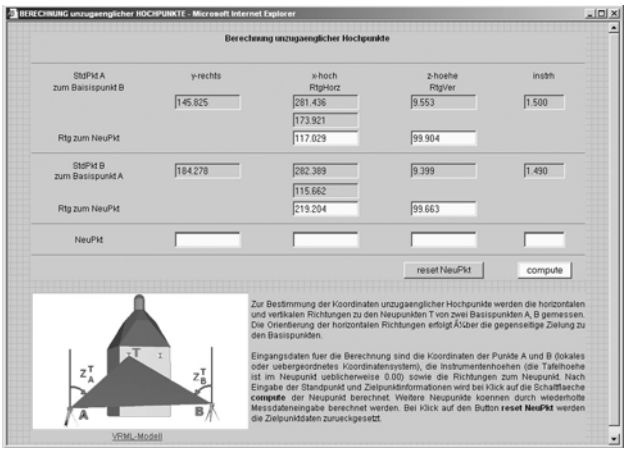

Figure 1. Calculating control points

1.3 Server Sided Dynamic Applying PHP

PHP Hypertext Pre-Processor is a server extension. The language is embedded in HTML documents. If the server detects PHP commands, the code will be carried forward to the PHP interpreter where the HTML code will first generated and after processing submitted to the client. The PHP code is not visible for the client. The browser receives the resulting HTML code only. The PHP interpreter supports amongst others XML processing, SQL data base functions, graphics and file access.

2. DESIGN OF A XML STRUCTURE

2.1 XML Extensible Mark-up Language

XML Extensible Mark-up Language is a meta language to structure information. XML uses tags, keywords in angle brackets with additional attributes, to enclose content. Compared with HTML, the denotation of the tags are defined not and will be interpreted first by the application.

XML files are plain text files. Rendering XML files require other technologies. XML documents are starting with a prologue, followed by the root element. The prologue basically displays details for the (DTD Document Type Definition) as stated below. The data used in the following examples are taken from a close-range photogrammetric Application.

```
<?xml version="1.0" encoding= "ISO-8859-1"
standalone="yes" ?>
<!DOCTYPE imageBundle SYSTEM
"imageBundle.dtd">
<imageBundle>
     <!--Inhalt des Dokuments -->
</imageBundle>
```

The root element *imageBundel* instances a document class, that is defined in the !DOCTYPE statement. An element and its subelements are represented by nodes in the tree structure. The element itself consists of a start-tag, the content and an end-tag. Attributes can be found in the start-tag. A photogrammetric camera definition may look as follows:

```
<cameraData>
  <camera>
    <type>nikon_28</type>
    <ck>-18.23718</ck>
    <xh>-0.09973</xh>
    <yh>-0.01304</yh>
    <a1>-3.03846E-004</a1>
    <a2>6.43569E-007</a2>
    <formX>23.462</formX>
    <formY>15.600</formY>
  </camera>
</cameraData>
```

A Web browser displays the tree structure of that document. By simply clicking the symbols, the nodes can be closed or opened, as shown in figure 2.

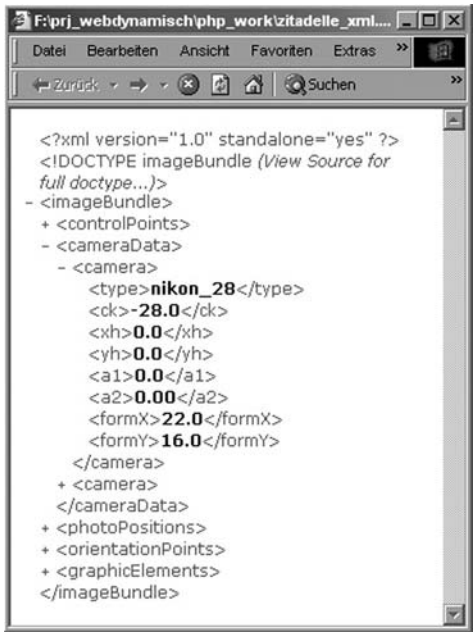

Figure 2. Tree structure of an XML document

2.2 XSL Extensible Stylesheet Language

Translation into another format is provided by the XSL Extensible Style Sheet Language. A XSL processor can be applied by the Web browser or server sided by the application. An off-line translation results in a HTML document to be stored on the server. Another important tool is XPATH. XPATH provides search patterns and enables extractions form XML documents. An external XSL file has to be referenced in the prologue by the statement:

```
<?xml-stylesheet version="1.0"
href="template.xsl" type="text/ xsl" ?>
```

We do not discuss the details of XSL here. An application of the for-each and value-of select statement may be give an idea how a XSL document is parsed by an XSL processor:

```
<xsl:for-each
   select="the search pattern">
   <tr>
    <td class = "tab_value"
    <xsl:value-of select="type"/>
    </td>
  </tr>
</xsl:for-each>
```

HTML tags are combined with XSL statements. The prefix xsl: defines the namespace, the class definition defines the appearance of the element in the browser.

2.3 DTD Document Type Definition

XML documents can be well formed or guilty. A well formed document becomes a guilty document by adding a DTD. A DTD defines the grammar for information processing of the XML file. All elements, attributes entities and specifications about quantity, content and nesting of elements must be predefined in a DTD. An extract of the DTD applied in the example is shown partly as follows:

```
<!DOCTYPE imageBundle [
  <!ELEMENT imageBundle (controlPoints*,
            cameraData* ,
            photoPositions*,
            orientationPoints*,
            graphicElements*)>
     <!ELEMENT controlPoints (point*)>
        <!ELEMENT point (pnr*, code*, x+, y+, z+)>
             <!ELEMENT pnr     (#PCDATA)>
             ...
     <!ELEMENT cameraData (camera+)>
        <!ELEMENT camera
         (type+, ck+, xh+, yh+, a1+, a2+,
          formX+, formY+)>
               <!ELEMENT type (#PCDATA)>
               <!ELEMENT ck   (#PCDATA)>
               <!ELEMENT xh   (#PCDATA)>
               <!ELEMENT yh   (#PCDATA)>
               <!ELEMENT a1   (#PCDATA)>
               <!ELEMENT a2   (#PCDATA)>
               <!ELEMENT formX(#PCDATA)>
               <!ELEMENT formY(#PCDATA)
```

Details can not be discussed here. It must be annotated, that the DTD will be replaced by the more powerful Xschema in future.

With XML, XSL and DTD structured information is separated from processing and prepared for further processing with object-oriented programming languages in client-server environments.

3. PARSING XML FILES WITH PHP

3.1 The Parser

We try to explain parsing and processing XML-documents by using the photogrammetric example from above. Requested are the camera values for the camera referenced by the description *nikon_28*. The client's request typed in the address line of the browser:

```
http://www.imagefact.de/zitadelle-wesel/
 parse_camera.php?camera=nikon_28
```

will be submitted to the server.

A parser will be needed for going through the file until the requested camera is found. The parser analyses and validates the file structure. Expat is an event driven parser and provided by PHP. Events are the occurrence of tags and content. An instance of the parser is created with the statement

```
$parser = xml_parser_create();
```

It is necessary to set parameter for the parser and handler for the elements, for example:

```
xml_set_element_handler( $parser,
   "start_element", "end_element" );
```

or

```
xml_set_character_data_handler(
    $parser, "inhalt" );
```

start_element , *end_elemen*t and *inhal*t are functions called by the parser, if it meets one of the defined events.

3.2 Processing Data

The function *inhalt* will be processed, if the content of a tag was *camera.* The camera data will be stored in arrays.

```
function inhalt($parser,$data ){
 global $curr_tag,$index,$camera
 switch ($curr_tag){
  case "type": $index=$data;break;
  case "ck": $camera[$index][ck]=$data;break;
  case "xh": $camera[$index][xh]=$data;break;
  case "yh": $camera[$index][yh]=$data;break;
  case "a1": $camera[$index][a1]=$data;break;
  case "a2": $camera[$index][a2]=$data;break;
  case "formX": $camera[$index][formX]=
                $data;break;
  case "formY": $camera[$index][formY]=
                $data;break;
  }
}
```

After getting the camera data, the arrays must be evaluated and the requested data, embedded in HTML-tags, has to be generated and must be passed to the client. PHP uses the echo-function for writing HTML commands. Figure 3 shows a request including image data as rendered by the browser. The complete examples can be found on the Web under *www.programmierpraktikum.de*, follow the navigation to the readers section (leserbereich), select there PHP&XML.

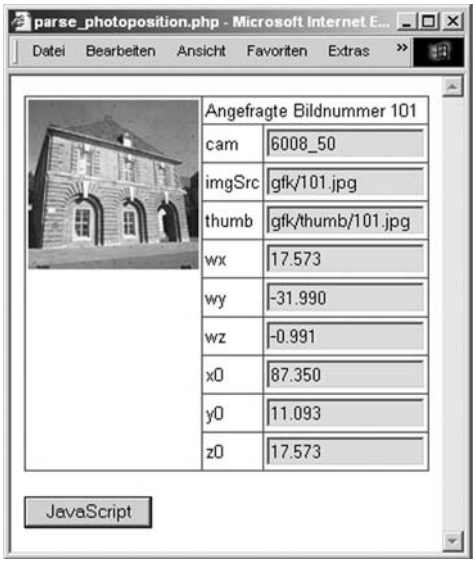

Figure 3. Parsing XML-Information and displaying the result on a Web page

4. PROCESSING GRAPHIC DATA

4.1 SVG Scalable Vector Graphics

Scalable Vector Graphics SVG is the XML formulation of 2D vector graphics. It includes drawing of vector data, displaying of image data, interaction and animation. Structure and appearance of graphic elements is separated by using style sheets

Applying a SVG viewer as a plug-in for the Web browser enables zooming and panning in the graphic area.

Here is a small sample including raster and vector graphics and interaction. Embedded is a script using ECMA (JavaScript), style sheet definition, mouse over events and referencing of external files. The displayed code is a fragment for documenting the integration of Web tools. Visit the a.m. Web site and click into the SVG area with the right mouse button. You have access to the complete code via the context menu.

Figure 4. Interaction with SVG

```
<svg width="640" height="480">
<title>SVG Interaction</title>
<script type="text/ecmascript">
<![CDATA[
function objekt_sichtbar(
  evt,sichtbarkeit){
  var svgDokument;
  svgDokument=
  evt.getTarget().getOwnerDocument();
  ...
   </script>
<style  type="text/css">
<![CDATA[
  path.sp {
  fill        :rgb(192,192,192);
  stroke      : #0000ff;
  stroke-width:2px;
  opacity:0.2;
  }
  ]]>
</style>
<g id="guide"transform="translate(50,50)">
<g id="fritz">
<image xlink:href="fritz.gif" x="0"
   y="112" width="138" height ="263"
   onmouseover="objekt_sichtbar(evt,'1')"
   onmouseout ="objekt_sichtbar(evt,'0')"
></image>
</g>
...
 </g></svg>
```

The bubble in figure 4 at first is hidden. Moving the mouse over the image displays the bubble. The text inside the bubble references links to external files.

4.2 X3D Extensible 3D

3D graphics data for the Web is well known as a VRML Virtual Reality Modeling Language description. That format is updated to the XML-version and named as X3D. The OCTAGA viewer can be applied for VRML and X3D in stand-alone mode or as a plug-in.

5. CONCLUSION

XML as a meta language is designed for structuring information and separating information from processing. The family of XML-languages and tools, like XSL, XPATH, SVG or X3D, provide device and platform independent processing. XML structured information can be prepared for displaying on monitors and PDAs or printing. The above given examples, taken from a photogrammetric project, stay for a variety of application potentials. The sketched code here, is printed to give an impression of the need for teaching and learning XML based Web technology.

References from Books:
Pomaska, G., 2005. *Grundkurs Web-Programmierung*. Vieweg-Verlag, Wiesbaden.

References from Other Literature:

Pomaska, G., 2003. Introduction of SVG as a data interchange format for architectural documentations. CIPA International Symposium, Antalya, Turkey.

Pomaska, G., Dementiev, N. XML basierte Datenformulierung zur Web-konformen Dokumentation photogrammetrischer Bauaufnahmen. PFG Zeitschrift für Photogrammetrie, Fernerkundung und GeoInformation, DGPF 2005, in Vorbereitung

References from Web sites:

http://www.programmierpraktikum.de

http://www.imagefact.de/zitadelle-wesel

http://www.adobe.com/svg

http://www.octaga.com