Building Interactive Books Using EPUB and HTML5

Dimitris Gavrilis, Stavros Angelis, and Ioannis Tsoulos

Digital Curation Unit – IMIS, Athena Research Center, Artemidos 6 & Epidavrou, Athens, Greece
{d.gavrilis,s.angelis}@dcu.gr, itsoulos@gmail.com

Abstract. The recent developments in the digital publishing domain have promoted the use of open formats for digital publishing. This paper presents a study that has been carried out using the EPUB 2.0 format along with HTML5 and JavaScript technologies showing how a digital book can embed HTML5 applications that interact with the user and report back to a publishing server (reader/student analytics).

Keywords: Digital publishing, epub, html5, analytics.

1 Introduction

The recent developments in the digital publishing domain have promoted the use of open formats for digital publishing. Now, books in the EPUB and PDF formats are read using internet tablets and smart phones and trends show they will replace traditional printed books much sooner than we think. The main reasons for this burst of electronic books, journals and newspapers are:

- The advances on mobile computing making mobile devices faster, more powerful and with better user experience
- The low cost of mobile devices
- The improved battery-life on mobile devices
- The lower cost of obtaining a newspaper / ebook in digital format
- The ability to transfer/copy a digital book to multiple media (smart phone, internet tablet, PC).
- The availability of tools (online translators, lexicons, annotations)

The major advantage of printed books and newspapers has always been the much better clarity when reading on paper. However, with the latest high-resolution screens found in high-end mobile devices this advantage is found in both sides.

1.1 The EPUB Format

The traditional EPUB 2.0 format consists of a set of xhtml files structured in folders and packaged using the zip format into a single file with an .epub extension. These folders contain the text (xhtml files), the styles (css file) and the multimedia files

C.T. Angelis, D. Fotiadis, and A.T. Tzallas (Eds.): AMBI-SYS 2013, LNICST 118, pp. 31–40, 2013.

(jpeg, png, mpeg etc) of the ebook. A file with an .opf extension is included, that contains descriptive information about the ebook and its contents in XML and a file with a .ncx extencion that contains the table of contents also in XML format. The protocol does not prohibit the inclusion of files like images, videos as long as they are defined in the OPF file. The OPF file contains the description of every part of the EPUB file.A simple example for this file is given below:

```
<?xml version="1.0"?>

<package version="2.0" xmlns="http://www.idpf.org/2007/opf" unique-
identifier="BookId">

<metadata xmlns:dc=http://purl.org/dc/elements/1.1/
xmlns:opf="http://www.idpf.org/2007/opf">
<dc:title>Pride and Prejudice</dc:title>
<dc:language>en</dc:language>
<dc:identifier id="BookId" opf:scheme="ISBN">123456789X</dc:identifier>
<dc:creator opf:file-as="Austen, Jane" opf:role="aut">Jane Austen</dc:creator>
</metadata>

<manifest>
<item id="chapter1" href="chapter1.xhtml" media-type="application/xhtml+xml"/>
<item id="stylesheet" href="style.css" media-type="text/css"/>
<item id="ch1-pic" href="ch1-pic.png" media-type="image/png"/>
<item id="myfont" href="css/myfont.otf" media-type="application/x-font-opentype"/>
<item id="ncx" href="toc.ncx" media-type="application/x-dtbncx+xml"/>
</manifest>

<spine toc="ncx">
<itemref idref="chapter1" />
</spine>

<guide>
<reference type="loi" title="List Of Illustrations" href="appendix.html#figures" />
</guide>

</package>
```

The NCX file contains the table of contents for the epub file and a relative simple example for this file is given below

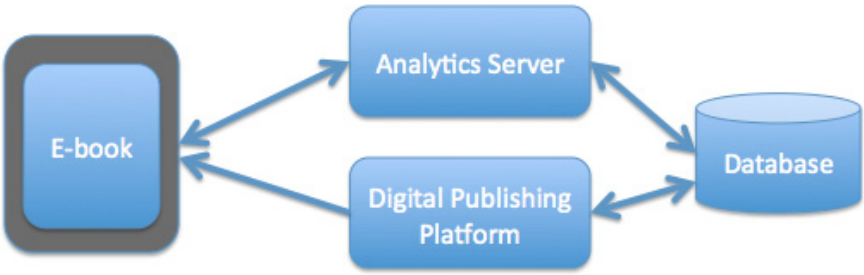
```
<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE ncx PUBLIC "-//NISO//DTD ncx 2005-
1//EN" "http://www.daisy.org/z3986/2005/ncx-2005-1.dtd">

<ncx version="2005-1" xml:lang="en" xmlns="http://www.daisy.org/z3986/2005/ncx/">
<head> <!-- The following four metadata items are required for all NCX documents,
including those that conform to the relaxed constraints of OPS 2.0 -->
  <meta name="dtb:uid" content="123456789X"/> <!-- same as in .opf -->
  <meta name="dtb:depth" content="1"/> <!-- 1 or higher -->
  <meta name="dtb:totalPageCount" content="0"/> <!-- must be 0 -->
  <meta name="dtb:maxPageNumber" content="0"/> <!-- must be 0 -->
</head>

<docTitle>
  <text>Pride and Prejudice</text>
</docTitle>

<docAuthor>
  <text>Austen, Jane</text>
</docAuthor>

<navMap>
  <navPoint class="chapter" id="chapter1" playOrder="1">
    <navLabel>
      <text>Chapter 1</text>
    </navLabel>
    <content src="chapter1.xhtml"/>
  </navPoint>
</navMap>

</ncx>
```

2 Smart Digital Books

The term smart digital books refer to ebooks that can provide interactivity to their readers, gather data and report these data to a cloud based analytics engine. Examples of such applications could be found in the education sector where a geometry application could be embedded in the page of an ebook and can report back the student's results to the analytics server. The results could indicate the types of errors the student makes and suggest an alternative reading path (e.g. suggest another – more basic math ebook to read).

2.1 Overall Technical Architecture

In contrast with traditional approaches, the technical architecture of a smart digital publishing platform requires a central server component that the ebook can link to if/when needed along with a knowledge base that contains the details of the data exchange protocol the specific ebook will use.



There are three main components found in the proposed architecture:

1. A database containing the ebooks, their metadata, a knowledge base of their sub-components (interactive tools)
2. A digital publishing platform containing all ebook information, information on members (e.g. students), book versions, etc. Through this server the client can buy/download the books, search the database, etc.
3. The analytics server is the primary component that receives information from the book itself. Information on opening the book, running one of the embedded applications and reporting back to the server if needed.

2.2 The Interactive Applications

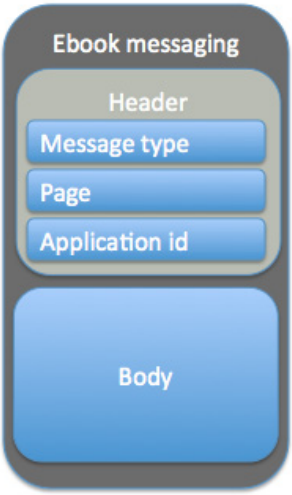
An interactive application is a javascript/HTML5 based application which is embedded in the book on a single page. It is recommended that the application is embedded in a single page so that the re-flow component on the client does display the application on a single page. The application requires a set of libraries (javascript

based) which should be defined under their own namespace/context and be included in the e-book thus can be run offline. A critical component in the application design is the one that handles communication with the analytics server. This particular component which we will focus on must take into account the following:

- Must be able to detect the presence of an Internet connection and consume all events if otherwise.
- Must be able to transmit all messages using the appropriate format for each application (JSON is the format used for information exchange).
- Must be able to periodically poll the analytics server for information (e.g. fetch relevant resources per page) but without interfering with the user's interaction with the ebook.

Each ebook transmits two types of messages (the type of message is encoded in the message header):
a) standard messages that transmit the page the user is on
b) messages that are application specific.

The message format is as follows:



The message contains a message type (0=standard, 1=application).

The page contains the following array of values:
chapter, font-size, width, height, page number.
These values are used to determine the position in the ebook (pages vary due to different screen sizes, font-sizes etc.).

The application id

The body of the message which contains an array of key-value pairs which are application specific.

2.3 The Analytics Server

The analytics server handles all communication with the ebooks using a single data format (e.g. JSON). When the server receives a message, it queries the database for relevant resources and reports back with a reply message to the ebook. In the case of standard messages, usually relevant information is pulled from the database and is returned to the ebook. It is up to the ebook reader to display this information or not. In the event of an application sending a message, the analytics server decides whether to store that message and how to respond.

3 Implementation

The implementation of the above architecture has been carried out using PHP-MySQL on the server side (publishing platform and analytics server) and Java using android - sdk on the client (reader for android smart phones and tablets). The messaging system has been implemented using JSON protocol, which is widely used, simple yet powerful in terms of abstraction. As an example we present below a fraction only of the developed code needed to parse and display EPUB files in the android – sdk environment.

```
package EpubReader.com;
import android.app.Activity;
import android.os.Bundle
import java.io.IOException;
import java.io.InputStream;
import java.util.List;
import nl.siegmann.epublib.domain.Book;
import nl.siegmann.epublib.domain.Resource;
import nl.siegmann.epublib.domain.TOCReference;
import nl.siegmann.epublib.epub.EpubReader;
import android.app.Activity;
import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.util.Log;
import android.view.Display;
import android.view.Gravity;
import android.view.View;
import android.webkit.WebView;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.ImageView;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.view.View.OnClickListener;
public class EpubReaderActivity extends Activity {

        Bitmap coverImage=null;
        ImageView imgButton=null;
        Button homeButton=null;
        Button nextButton=null;
        Button prevButton=null;
```

```
        ImageView img=null;
        WebView view=null;
         List<Resource> l;
        int currentPage=1;
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
 setContentView(R.layout.main);
 Display display = getWindowManager().getDefaultDisplay();
 int width = display.getWidth();
 int height = display.getHeight();
  AssetManager assetManager = getAssets();
  try {
    // find InputStream for book
    InputStream epubInputStream = assetManager
        .open("books/Dan.epub");
    // Load Book from inputStream
    Book book = (new EpubReader()).readEpub(epubInputStream);
    // Log the book's authors
    Log.i("epublib", "author(s): " + book.getMetadata().getAuthors());
    // Log the book's title
    Log.i("epublib", "title: " + book.getTitle());
    l=book.getContents();
   for(int i=0;i<l.size();i++)
   {
        Log.i("epublib","Data="+l.get(i).toString());
   }
    // Log the book's coverimage property
   coverImage = BitmapFactory.decodeStream(book.getCoverImage()
        .getInputStream());
    Log.i("epublib", "Coverimage is " + coverImage.getWidth() + " by "
        + coverImage.getHeight() + " pixels");
    // Log the tale of contents
    logTableOfContents(book.getTableOfContents().getTocReferences(), 0);
  } catch (IOException e) {
      Log.e(null, "adynamia anoigmatos arxeiou");
    Log.e("epublib", e.getMessage());
  }
  imgButton=(ImageView)findViewById(R.id.button1);
  imgButton.setImageBitmap(coverImage);
  imgButton.setClickable(true);

  homeButton=(Button)findViewById(R.id.button2);
  homeButton.setText("Home");
  homeButton.setOnClickListener(new OnClickListener()
```

```
{
    @Override
    public void onClick(View v) {
    currentPage=1;
    byte[] bytes = null;
                        try {
                                bytes=l.get(currentPage).getData();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                String encoding=l.get(currentPage).getInputEncoding();
                String contentsHtml=new String(bytes);
                String type=l.get(currentPage).getMediaType().getName();
                view.loadData(contentsHtml, type, encoding);
                view.invalidate();
    }
});
prevButton=(Button)findViewById(R.id.button3);
prevButton.setText("Prev");
prevButton.setOnClickListener(new OnClickListener()
{
            @Override
            public void onClick(View v) {
            currentPage=currentPage-1;
            if(currentPage==0)
            {
                    currentPage=1;
                    return;
            }
              byte[] bytes = null;
                try {
                                bytes=l.get(currentPage).getData();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                String encoding=l.get(currentPage).getInputEncoding();
                String contentsHtml=new String(bytes);
                String type=l.get(currentPage).getMediaType().getName();
                view.loadData(contentsHtml, type, encoding);
                view.invalidate();
    }
    });
nextButton=(Button)findViewById(R.id.button4);
nextButton.setText("next");
nextButton.setOnClickListener(new OnClickListener()
```

```
{

    @Override
    public void onClick(View v) {
            currentPage=currentPage+1;
            if(currentPage>=l.size())
            {
                    currentPage--;
                    return;
            }
             byte[] bytes = null;
               try {
                            bytes=l.get(currentPage).getData();
                    } catch (IOException e) {
                                    // TODO Auto-generated catch block
                            e.printStackTrace();
                    }
      String encoding=l.get(currentPage).getInputEncoding();
      String contentsHtml=new String(bytes);
      String type=l.get(currentPage).getMediaType().getName();

                        view.loadData(contentsHtml, type, encoding);
                        view.invalidate();
            }
 });
 view=(WebView)findViewById(R.id.webView1);
view.getSettings().setJavaScriptEnabled(true);
 view.getSettings().setAppCacheEnabled(true);
 view.getSettings().setJavaScriptCanOpenWindowsAutomatically(true);
 byte[] bytes = null;
 try {
            bytes=l.get(currentPage).getData();
    } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
    }
 String contentsHtml=new String(bytes);
 String encoding=l.get(currentPage).getInputEncoding();
 String type=l.get(currentPage).getMediaType().getName();
 view.loadData(contentsHtml, type,encoding);
 view.loadUrl( "javascript:window.location.reload( true )" );  }

/**
 * Recursively Log the Table of Contents
 *
```

```
     * @param tocReferences
     * @param depth
     */
    private void logTableOfContents(List<TOCReference> tocReferences,
        int depth) {
      if (tocReferences == null) {
          Log.i("epublib","null toc");
        return;
      }
      Log.i("epublib","size of toc = "+tocReferences.size());
      for (TOCReference tocReference : tocReferences) {
        StringBuilder tocString = new StringBuilder();
        for (int i = 0; i <=depth; i++) {
          tocString.append("\t");
        }
        tocString.append(tocReference.getTitle());
        Log.i("epublib", tocString.toString());
         logTableOfContents(tocReference.getChildren(), depth + 1);
      }
    }
  }
```

4 Conclusions

In this paper a smart ebooks architecture has been presented. This architecture has been designed based on the EPUB 2.0 protocol, it makes use of javascript and html5 technologies and a prototype application has been developed using Java on an android device (smart phone and tablet). On the server side, the server components have been implemented using open source tools: Php, MySQL, Apache. The preliminary results show the capabilities of real interactive ebooks and are very promising. Future work includes the use of a reasoning engine that will process the application messages.

References

1. http://en.wikipedia.org/wiki/EPUB
2. http://idpf.org/epub
3. Miller, B.N., Ranum, D.L.: Beyond PDF and ePub: toward an interactive textbook. In: ITiCSE 2012 Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, pp. 150–155 (2012)