Lectura offline

Exportar posts del blog de HTML a ePub y PDF)

Willem L. Middelkoop
19 ene. 2026

Este año marca el décimo aniversario de mi blog. Lo que empezó como un experimento se ha convertido en más de 170 historias, con un promedio de alrededor de 60 mil pageviews al mes. Recientemente comencé a traducir entradas al neerlandés, alemán y español para ampliar el alcance. Y ahora estoy introduciendo soporte para lectura sin conexión al hacer que todas las entradas estén disponibles como descargas en ePub y PDF.

Lee las entradas de willem.com en tu e-reader favorito (como Kobo o reMarkable)

¿Por qué offline?

Disfruta de una lectura sin distracciones en dispositivos de tinta electrónica como un Kobo o reMarkable: estas pantallas son suaves con tus ojos. Concéntrate únicamente en el artículo, donde y cuando quieras, ya que son descargas que funcionan completamente offline y sin DRM. Todos los ePub y PDF de willem.com incluyen imágenes de alta calidad y una tipografía hermosa y escalable. He preparado un práctico resumen de todas las descargas en ePub y PDF aquí:

Descargar ePub y PDF

¿Cómo funciona?

El sitio willem.com funciona con el sistema de gestión de contenidos Lemmid, un generador de sitios estáticos que produce HTML optimizado, perfecto para alto volumen de tráfico y distribución internacional. He extendido el código NodeJS para exportar automáticamente cada entrada a ePub y PDF a partir de HTML simplificado como entrada. Mira los fragmentos de código a continuación: para **ePub** uso el módulo epub-gen; para **PDF** uso Pandoc y LaTeX. Construí un gestor de tareas sencillo que pone en cola los trabajos de salida para traducciones individuales y formatos de archivo, distribuyendo la carga computacional de forma eficiente en el backend.

Conclusión

Las reacciones que recibo a mis entradas llegan de todo el mundo y me motivan a seguir escribiendo. Con el soporte para ePub y PDF espero ampliar aún más el alcance, ofreciendo nuevas formas de leer los artículos. ¡Disfrútalo!

```
Publisher.prototype.performOutputJobEpub = function(job, callback) {
   // console.log('Publisher: performOutputJobEpub '+job.instance.id+' as '+job.type);
   var instance = job.instance;
   var language = job.language;
   var languageSuffix = language.toUpperCase();
   var epubFilename = instance['slug' + languageSuffix] + '.epub';
   this.base.isFileNewerThanTimestamp(instance['absolutePath' + languageSuffix] + epubFilename, instance.lastUpdated, function(isNewer) {
      if (isNewer) {
         callback(); // no action needed
      } else {
         // the ePub needs to be generated
         const heroImagePath = (instance['image' + languageSuffix])
            ? this.base.resultPath + instance['image' + languageSuffix].url.w1000
            : this.base.resultPath + '/global/icon.png';
         // Localised strings for the Table of Contents
         const tocTitles = {
            en: "Table of Contents",
            nl: "Inhoudsopgave",
            de: "Inhaltsverzeichnis",
            es: "Tabla de Contenidos"
         };
         const fullTitle = instance['title' + languageSuffix] + ': ' + instance['subTitle' + languageSuffix];
         // the inputHtml is based on feedHtml which uses absolute URL's, since
         // we're processing locally on the filesystem into epubs, we're removing
         // online (image) links, transforming them in absolute filesystem paths,
         // we also remove the anti cache parameter ?=LASTUPDATED
         const inputHtml = '<h4>' + instance['dateFormatted' + languageSuffix] + '</h4>\n' +
               this.base.replaceAllInString(
                 this.base.replaceAllInString(instance['feedHtml' + languageSuffix],
                     'https://' + this.base.rootFolderName + instance['relativePath' + languageSuffix],
                     instance['absolutePath' + languageSuffix]),
                 '?u=' + instance.lastUpdated, '');

         const options = {
            title: fullTitle,
            author: this.authorName,
            publisher: this.base.rootFolderName,
            lang: language,
            cover: heroImagePath,
            tocTitle: tocTitles[language] || tocTitles['en'], tempDir: this.rootPath + '/tmp',
            content: this.base.splitHtmlIntoChapters(inputHtml, fullTitle),
            appendChapterTitles: true,
            verbose: this.isDebug
         };
         const outputPath = instance['absolutePath' + languageSuffix] + epubFilename;
         // Generate the EPUB using promise .then/.catch for callback hook.
         new Epub(options, outputPath).promise .then(() => {
            console.log(`Publisher: Successfully generated EPUB for ${instance.id}`);
            if (callback)
               callback();
         }) . catch ((error) => {
            console.error(`Publisher: Error generating EPUB for ${instance.id}:`, error); if (callback)
               callback();
         });
      } // !isNewer
   }
   .bind(this)); // executeIfPathIsOlder
}
```

ePub code uso epub-gen

```
Publisher.prototype.performOutputJobPdf = function(job, callback) {
   // console.log(`Publisher: performOutputJobPdf `+job.instance.id+' as '+job.type);
   var instance = job.instance;
   var language = job.language;
   var languageSuffix = language.toUpperCase();
   var pdfFilename = instance['slug' + languageSuffix] + '.pdf';
   this.base.isFileNewerThanTimestamp(instance['absolutePath' + languageSuffix] + pdfFilename,
                                      instance.lastUpdated, function(isNewer) {
      if (isNewer) {
         callback();
         // no action needed
      } else {
         // the PDF needs to be generated
         console.log('Publisher: generating PDF for ' + job.instance.id);
         var inputHtml = `<html lang="${language}">
                           <head>
                               <title>${instance["title"+languageSuffix]}</title>
                           </head>
                           <body>
                               ${instance["feedHtml"+languageSuffix]}
                           </body>
                       </html>`;
         // the inputHtml is based on feedHtml which uses absolute URL's, since
         // we're processing locally on the filesystem into PDF's, we're removing
         // online (image) links, transforming them in absolute filesystem paths:
         // we also remove the anti cache parameter ?=LASTUPDATED
         inputHtml = this.base.replaceAllInString(this.base.replaceAllInString(inputHtml,
                        'https://' + this.base.rootFolderName + instance['relativePath' + languageSuffix],
                        instance['absolutePath' + languageSuffix]), '?u=' + instance.lastUpdated, '');
         // we can only do the PDF generation if we have the basic html as input,
         // so we make sure that file is written first:
         this.base.makeSureFileIsWritten(instance['absolutePath' + languageSuffix] + 'tmp-pdf-input.html', inputHtml, function() {
            // Then we compose variables used to start a pandoc process to generate a PDF
            // using LateX magic and the basic.html as input:
            const languageSuffix = language.toUpperCase();
            const inputPath = instance['absolutePath' + languageSuffix] + 'tmp-pdf-input.html';
            const outputPath = instance['absolutePath' + languageSuffix] + pdfFilename;
            const templatePath = this.base.rootPath + '/assets/template.latex';
            const title = this.base.escapeLatex(instance['title' + languageSuffix]);
            const subTitle = this.base.escapeLatex(instance['subTitle' + languageSuffix]);
            const author = this.base.escapeLatex(this.authorName);
            const date = instance["dateFormatted" + languageSuffix];
            const heroImagePath = (instance['image' + languageSuffix])
                ? this.base.resultPath + instance['image' + languageSuffix].url.w1000
                : this.base.resultPath + '/global/icon.png';
            // Construct the Pandoc command with the corrected engine
            const command = `pandoc \\
                             -f html \\
                             -o "${outputPath}" \\
                             --pdf-engine=xelatex \\
                             -V title="${title}" \\
                             -V subTitle="${subTitle}" \\
                             -V heroImagePath="${heroImagePath}" \\
                             -V author="${author}" \\
                             -V date="${date}" \\
                             --template="${templatePath}" \\
                             "${inputPath}"`;
            // Execute the command (and pray for some good luck)
            exec(command, (error, stdout, stderr) => {
               if (error) {
                  console.error(`Publisher: Error generating PDF for ${instance.id}: ${error.message}`);
                  callback();
                  // keep continuing on the other jobs
                  return;
               }
               // if (stderr) {
               // console.warn(`Publisher: Pandoc stderr for ${instance.id}: ${stderr}`);
               // }
               console.log(`Publisher: Successfully generated PDF at ${pdfFilename}`);
               // clear tmp input HTML file:
               fs.unlink(inputPath, function(error) {
                  if (error)
                     throw error;
                  console.log('Publisher: Removed tmp HTML file from ' + inputPath);
               });
               callback();
               // tell the job queue we're done by calling back
            });
            // exec pandoc
         }
         .bind(this)); // makeSureFileIsWritten basic.html
      } // !isNewer
   }
   .bind(this)); // executeIfPathIsOlder
}
```

PDF code uso Pandoc y LaTeX