ASPOSE
Your File Format Experts

CASE STUDIES

Aspose.Total for .NET Case Study

Atril Solutions

Using Aspose.Total for .NET to improve filters and provide real-time preview in the Déjà Vu translation tool

Blandine Loze, Managing Director, November 10, 2013

Daniel Benito, CTO, November 10, 2013

About Atril Solutions

Leaders in Computer Assisted Translation since 1993

Atril has a unique reputation for innovating translation technology from the standpoint of the user, not the technologist. We've never considered new functions in isolation or introduced them solely for the sake of the technology. We've always worked to a bigger picture.

Right from our very beginnings in 1993 we've been committed to much more than simply automating the translation process to enhance productivity, aiming above all to provide a tightly-integrated tool that helps everyone in the translation supply chain work as effectively and efficiently as possible. Which shouldn't be surprising, since those are our roots.

Our system designs have as a result never involved any trade-offs between quality and productivity. Just as they have always been conceived so they can be easily and flexibly customized to individual needs.

It's this thinking which has generated our constant technological innovation. Just as it has involved us in a close dialogue with our users that has in itself been innovative, while ensuring our product development makes a real impact on their quality, productivity and reputation.

Over the years we have added a large number of features. As we continued research into the techniques that drive Déjà Vu, we have steadily added new translation features and refined existing techniques to increase both speed and output. And we have also added a large number of filters, both for documents and for database importing and exporting.

Problem

Our translation tool, Déjà Vu, provides an interface that allows users to translate documents in a wide variety of formats. It includes a number of filters which extract the translatable content from the documents, and store any inline markup or formatting information in placeholders. The user therefore does not see the real formatting of the documents; while in some cases this can be an advantage, having to rearrange numerous placeholders in the translated text is a time-consuming task.

The increase in complexity of the Microsoft Office applications over the years means that Office documents, and Word documents in particular, are littered with markup that is not visible to the user when working in Office, but results in a large number of unnecessary placeholders when those documents are imported into Déjà Vu.

Until now, processing the older pre-Office 2007 formats required the users to have Office installed, and to partly or completely process the files through Office automation. Having to rely on automation also limits the stability of the filters, since they cannot be safely used in server-side applications. Having separate filters for the Office Open XML formats creates unnecessary complexity for the end user, who is not interested in whether a DOC file has to be converted into RTF using Word automation before being imported, whereas a DOCX file can be processed by extracting the XML content inside it.

When we began planning the new version of our product, we decided we wanted to change all of this – we needed components that would allow us to process the different types of Office documents consistently without relying on OLE automation, would let us easily extract formatting information so that we could display this to the user in a pseudo-WYSIWYG interface, and easily convert document content to HTML so that we could provide users with a more realistic real-time preview of the formatted translation.

For all our users – over 10,000 across the world – this will be an incredibly welcome improvement, as it will improve the ergonomics and usability of our product and reduce the amount of additional formatting they need to do outside our tool, thus increasing their productivity considerably.

Solution

We chose Aspose because it provided nearly all of the functionality of the Office Open XML SDK, but it also provided consistent solutions across different file formats (for example RTF, DOC and DOCX) and allowed us to easily extract and manipulate basic formatting information.

We completely re-wrote our existing Office document filters using Apose.Total for .NET, which resulted in incredibly clean code, compared to our previous codebase which had extensive classes to convert, clean and parse RTF, or to extract and parse XML files from Open XML files. In particular, the way shared strings and character formatting are handled

in Excel versus the way Aspose.Cells presents it to the programmer made the different between thousands of lines of codes in our old codebase and only a few hundred in the new one.

With our new solution based on Aspose.Words, Aspose.Cells and Aspose.Slides, users simply instruct our translation tool to import a document, and the user is presented with all of the translatable content, with visual representation of basic formatting attributes such as bold, italics, superscript, etc. In addition, users have an HTML-based real-time preview of the translated content, based on HTML generated from the source documents using Aspose components. Once the user has finished translating the document, our tool modifies the original document, replacing the original text with the translation, and using the Aspose components to modify any formatting information that is required, in addition to changing language identifiers, text directionality, etc.

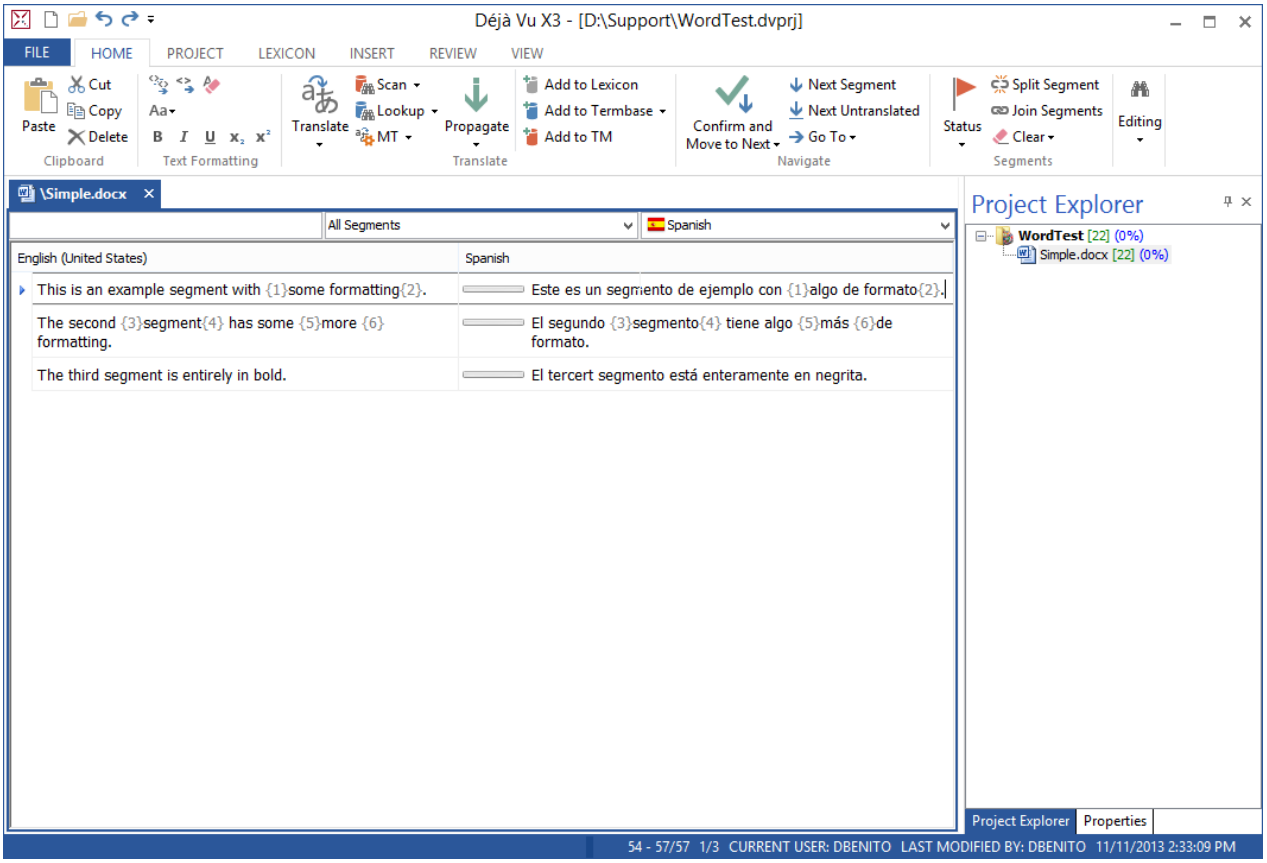

Figure 1: Déjà Vu X3 UI before using previous DOCX filter.

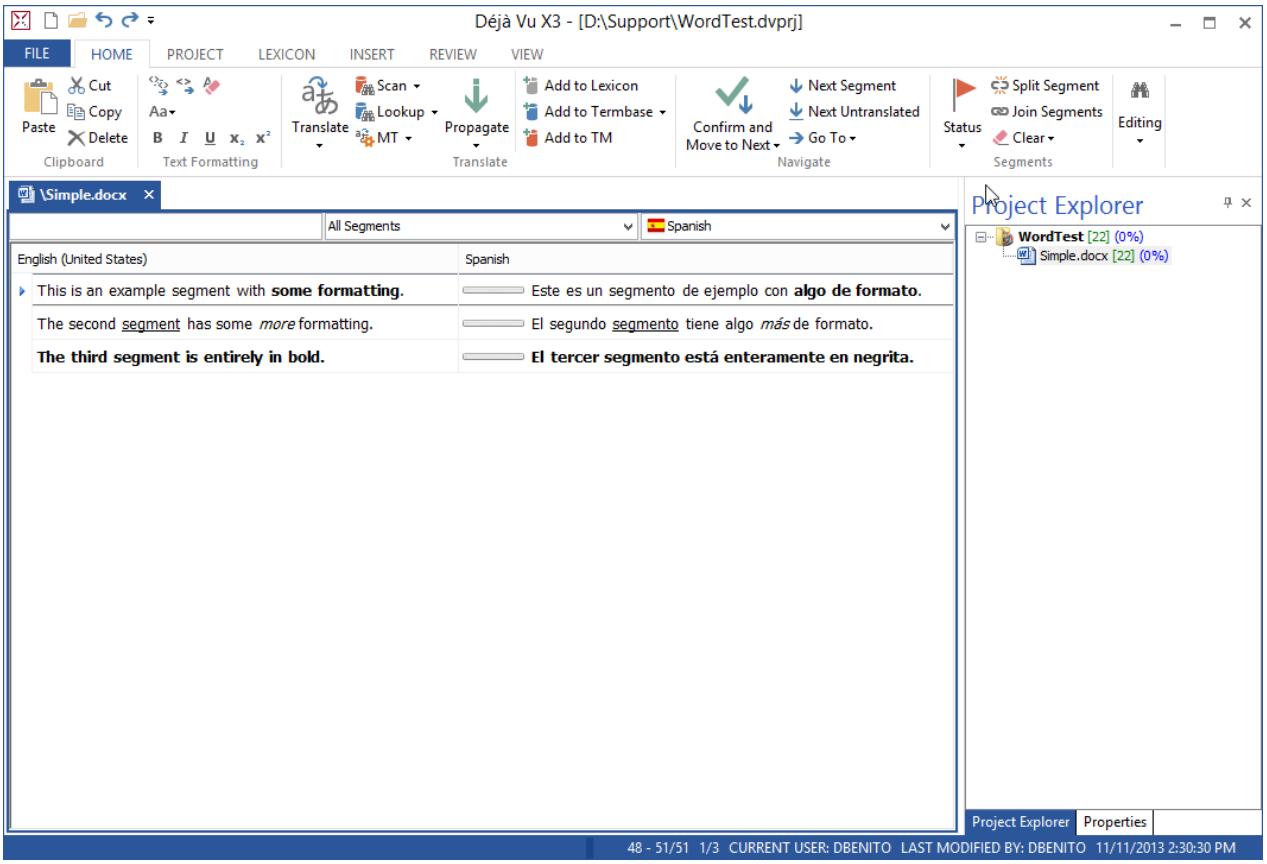

Figure 2: Déjà Vu X3 UI after rewriting Microsoft Word filter using Aspose.Words.

Experience

Finding a solution: Apart from the Office Open XML SDK, we did not find any other tool that provided the functionality that we needed – we weren't interested in simply generating Office documents, but needed to open existing documents and extract information from them. We saw recommendations for Aspose on various web sites, so we downloaded a trial version and began testing it; before the trial had ended, we had fully working versions of the two of the three new filters. We did not need to contact support because the examples and the documentation were extremely clear, and the only issues we came across had already been addressed in the Aspose forums.

Implementation: It only took a single programmer 3 weeks to implement a the first phase of the solution, which included complete import and export including inline formatting. The

object model exposed by the various Aspose components made the newly written code very clean and concise, making it easy to maintain and extend.

Outcome: We are currently about to begin external beta testing of the new functionality, so we don't yet have feedback from our users.

Next Steps

The object model exposed by the Aspose components means we need to spend less time worrying about the internal structure of the document formats and can easily access the document content in a variety of ways. This will allow us to easily extend the filters to add new functionality in future releases. We are also considering using Aspose.Tasks and Aspose.Diagram to process Project and Visio files, respectively, in future releases.

Summary

We were very pleased that, as we began working with Aspose products, they lived up to our expectations. They made the design and coding of the new document filters simpler, cleaner and quicker, helping to greatly reduce complexity (and therefore increase quality) in one of the more complex parts of our codebase.

We would readily recommend Aspose components to anybody who needs to process or generate Office documents, either in a desktop tool or in a server application.