A PDF Storage Backend for Protégé

Henrik Eriksson

Dept. of Computer and Information Science
Linköping University
SE-581 83 Linköping, Sweden

her@ida.liu.se

Introduction

Protégé is a successful environment for editing ontologies and knowledge bases. In Protégé, there are several tab-level extensions available to ontology developers, which add to the user-interface of Protégé [4]. However, there are few alternatives for storing ontologies and knowledge bases. Currently, Protégé stores ontologies and knowledge bases as files or tables in relational databases. The file storage consists of a group of separate files; that is, a project file (with the .pprj extension) and files for classes and instances (individuals). Some storage formats combine classes and instances in a single file. The advantage of using several files for storage is that the files can be used directly by other applications, such as applications processing instances only. The advantage of a single storage file is that it simplifies file management, such as renaming and copying. Furthermore, it might be easier for novice users to handle single-file storage than to manage multiple files.

Fortunately, the Protégé architecture separates its internal knowledge representation from the external serialisation of the knowledge-base content. The Protégé application-programming interface (API) supports storage-backend extensions, which allow developers to change the way Protégé saves and loads ontologies and knowledge bases. The standard Protégé distribution contains different storage-backend implementations for Clips, XML, RDFS, OWL, and databases. The standard file-based backend extensions, however, tend to use multiple files and voluminous syntax without compression. Thus, from a pure data-storage point of view, these formats are inefficient.

The goal of the *PDF backend approach* is to use the Portable Document Format (PDF) as the basis for a Protégé storage backend. This approach allows Protégé users to store ontologies and knowledge bases inside PDF files. It is possible to use pre-existing PDF documents as templates and add ontologies and knowledge bases to them. The resulting PDF files will still be documents that users can view on-screen and print. Tools for handling PDF, such as Adobe Acrobat Reader, will continue to work as before.

We have previously explored the use of PDF documents as the basis for *semantic documents* [2,3]. The goal of the semantic-document approach is to bring documents and knowledge bases closer together. The difference between semantic documents and the PDF storage backend is that semantic documents are intended for document annotations, which relate concepts in the ontology to selected text in the documents, whereas the PDF storage backend stores ontologies as attachments to PDF files.

We believe that the PDF backend approach will help Protégé users manage file-based storage of ontologies and knowledge bases. One of the advantages of using PDF as the storage format is that it supports compression of attached files. In addition to working as a normal document, the resulting file will be compact and will combine the project, ontology, and instance data in one package.

Background

PDF is an open format developed by Adobe [1]. In addition to Adobe products, such as Acrobat, there are several commercial and open-source tools that create, manage, and read PDF documents. Originally, Adobe designed PDF to support platform and device-independent printing and fast on-screen viewing. PDF documents consist of an indexed structure of internal document objects, such as text and binary streams.

The PDF specification supports the *attachment* of files to PDF documents. Just as it is possible to attach an arbitrary file to an e-mail, PDF allows for additional files to be inserted into the internal structure of PDF files. Although this feature is commonly available in Adobe products, such as Acrobat Professional (for reading and attaching) and Acrobat Reader (for reading), it is less known.

Saving to PDF

The PDF storage back-end is a prototype implementation of the Protégé back-end API. It redirects the saving of projects and knowledge bases to PDF files. If the PDF document does not exist, the back-end will create a new one automatically. In the current implementation, it creates a one-page document with the name of the knowledge base and some information about the document. In principle, it is possible to generate other types of initial documents, such as tables with knowledge-base metrics, instructions for downloading Protégé and reports generated from knowledge-base content.

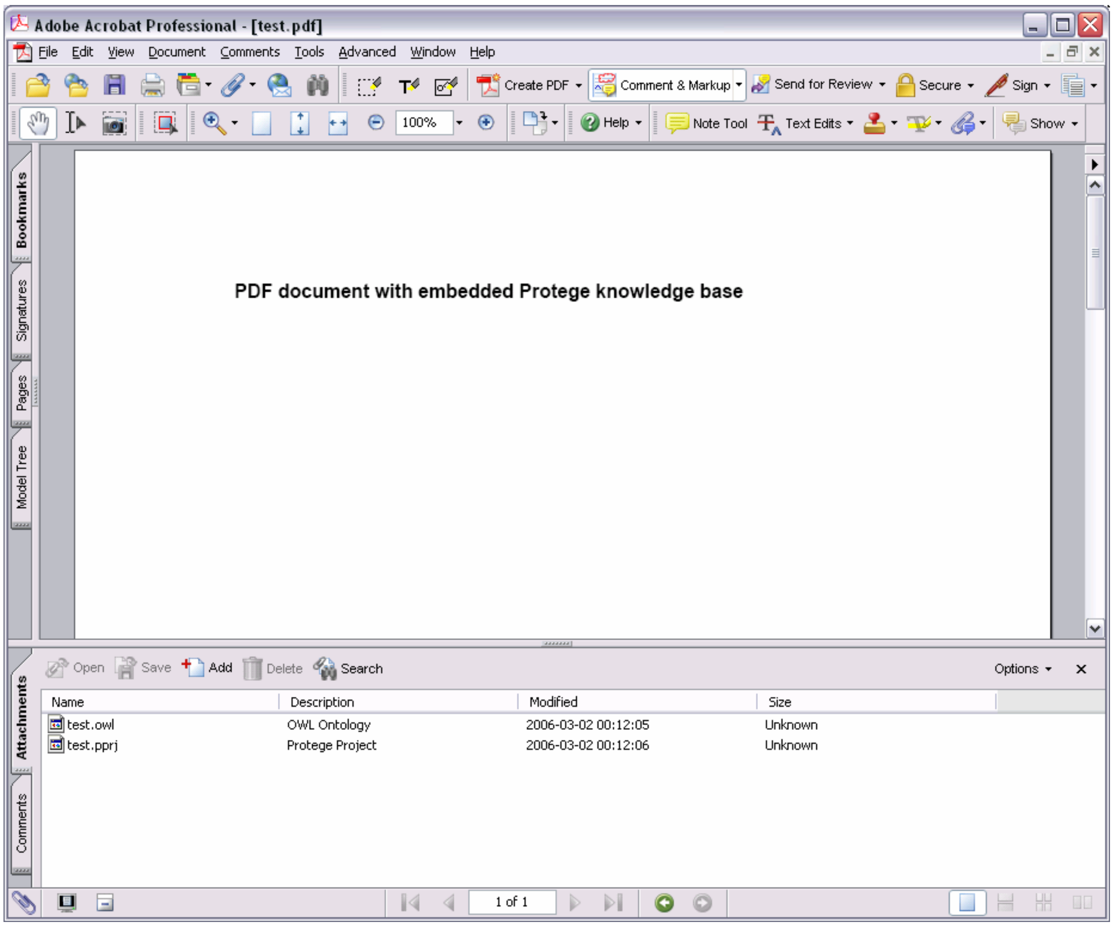
The PDF storage back-end uses PDFBox for accessing PDF documents and adding attachments to them. PDFBox is an open-source Java-based library for reading, manipulating, and writing PDF documents (see http://www.pdfbox.org/). PDFBox can parse PDF documents and represent the document content using object structures in Java. Furthermore, PDFBox supports document creation from Java and insertion of attachments into documents. We have found that PDFBox works well with Protégé and that it provides the functionality required for document generation, and reading and writing attachments to/from streams in a document. In PDFBox, adding compression to attachments is a straightforward task.

The attachments saved with the PDF document are available in tools such as Acrobat Reader (by opening the “Attachments” tab). Users can easily extract these attachments and use them as normal project and knowledge-base files for Protégé. When Protégé loads a PDF document, it first parses the file using PDFBox and then loads the attachments from input streams provided by PDFBox. This process is quite efficient. Because the objects in PDF documents are indexed, PDFBox can use fast random-file access to extract the attachments without processing the entire document.

Example

Let us consider an example of how the PDF storage backend saves Protégé projects and ontologies in documents. Figure 1 shows a sample PDF document generated by the storage backend. This document contains two file attachments, test.owl and test.pprj. These files correspond to the files saved normally by the OWL backend. Furthermore, it is possible to extract these attachments from the PDF document (e.g. using Acrobat) and use them as normal Protégé files. Likewise, ontology developers can manually construct new documents for the PDF backend by attaching normal Protégé files to a PDF document.

The document view in Acrobat shows the generated one-page document with the template text, "PDF document with embedded Protégé knowledge bases." Note that it is possible to open this document with any PDF viewer (although some third-party viewers and old Acrobat versions may show the attachment). In principle, this generated front page can contain any information about the ontology, such as instructions for using the ontology, ontology metrics, and text generated from the ontology content. An alternative is to start with an existing PDF document and let Protégé add the project and ontology attachments to it. Another possibility is to add other project-related files, such as plug-in specific resource files, as document attachments.



PDF document with embedded Protege knowledge base

Name	Description	Modified	Size
test.owl	OWL Ontology	2006-03-02 00:12:05	Unknown
test.pprj	Protege Project	2006-03-02 00:12:06	Unknown

Figure 1. Sample PDF document opened in Acrobat. In this case, the PDF back-end generated a single document page and stored the project and ontology as attachments. The attachments tab in Acrobat shows a list of the document attachments.

Discussion

One of the major challenges for implementing the prototype of the PDF back-end was to modify the libraries and APIs that the back-end uses. The back-end required modifications (bug fixes and/or extensions) to PDFBox, the Protégé OWL extension, and the Protégé core API. In particular, the Protégé core API assumed the existence of a project (.pprj) file for storing project-specific information, such as custom adjustments to forms. It was necessary to add API functionality for storing the project information in alternative formats and locations. Currently, there are still some unresolved issues in terms of using the same storage extension for several knowledge-based formats, such as frames, RDFS, and OWL.

The implementation of PDF back-end shows that it is possible to develop file-based storage solutions that overcome some of the problems with multiple files, and that provide compact storage for XML-based formats. It is possible to use a similar method for developing storage back-ends for other archive and compression formats, such as jar, tar and zip. These formats allow other applications to read files produced by Protégé without the overhead of PDFBox. Furthermore, new storage formats help make Protégé useful for new groups of users, and expand potential uses of ontologies.

Summary and Conclusions

We believe that PDF is a viable storage format for Protégé ontologies and knowledge bases. The advantage of storing project, class and instance files as PDF attachments is that it is a straightforward method that allows document viewing and printing using standard PDF tools. Note, however, that this approach is different from semantic documents, which relate document content to concepts in the ontology. The prototype implementation illustrates that PDF storage works. However, there are still some limitations to the Protégé API that make it difficult to develop a storage plug-in that works for all knowledge-base formats. Nevertheless, we believe that PDF storage will be a highly useful feature of Protégé.

Acknowledgements

This work was supported by Vinnova (grant no. 2003-01415), by The Swedish Research Council (grant no. 621-2003-2991), and by Statistics Sweden.

References

[1] Adobe (2004a). PDF Reference Version 1.6. Adobe Press, Berkeley, CA, 5th edition.

[2] Henrik Eriksson. The semantic document approach to combining documents and ontologies. *International Journal of Human–Computer Studies*, in press.

[3] Henrik Eriksson. Support for semantic documents in Protégé. In *Proceedings of the Eight International Protégé Conference,* Madrid, Spain, July 18–21, 2005.

[4] John H. Gennari, et al. The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human–Computer Studies,* 58(1):89–123, 2003.