Open Digital Forms

Hiep Le, Thomas Rebele, Fabian Suchanek

Télécom ParisTech, Paris, France

Abstract. The maintenance of digital libraries often passes through physical paper forms. Such forms are tedious to handle for both senders and receivers. Several commercial solutions exist for the digitization of forms. However, most of them are proprietary, expensive, centralized, or require software installation. With this demo, we propose a free, secure, and lightweight framework for digital forms. It is based on HTML documents with embedded JavaScript, it uses exclusively open standards, and it does not require a centralized architecture. Our forms can be digitally signed with the OpenPGP standard, and they contain machine-readable RDFa. Thus, they allow for the semantic analysis, sharing, re-use, or merger of documents across users or institutions.

1 Introduction

The world's digital libraries contain millions of documents, visual artwork, and audio material. The maintenance of these libraries often passes through physical paper forms. Out of the 50 first English digital library projects on Wikipedia[1], more than half offer a service that requires sending a paper form by letter. Such paper forms are used for membership requests, for licensing, for revocation requests, etc. In one case, 3 different forms have to be filled out and sent physically to the library[2]. The dependence on paper forms is particularly surprising for projects that aim to digitize paper documents, but the problem is of course in no way restricted to digital libraries.

Paper forms require substantial manual work for the sender and the receiver. Therefore, several commercial solutions exist for the digitization of forms. However, most of these are proprietary, pricy, centralized, or require software installation (see Section 2). Thus, the challenge is to develop a solution so that

- no additional software is required to display and fill forms.
- only open standards are used, i.e. everyone can use the formats and editing software without restrictions.
- the user remains in control of their data.

With this demo, we propose a free, secure, and lightweight framework for digital forms that fulfills these desiderata:

- Our solution is based on HTML documents, and hence does not require any software beyond a browser.
- Our solution uses exclusively open standards, which makes the forms easy to parse, verify, produce, or modify.

[1] `https://en.wikipedia.org/wiki/List_of_digital_library_projects`
[2] `http://www.ahds.ac.uk/depositing/how-to-deposit.htm`

- Our solution does not require a centralized architecture. The user remains in complete control of their data.
- Our forms can be digitally signed with the OpenPGP standard, which can be verified by open software.
- Our forms are machine-readable in RDFa, thus allowing the semantic analysis, sharing, re-use, or merger of forms.

2 Related Work

Commercial Solutions. Adobe Acrobat, LibreOffice and others support the creation of Portable Document Format (PDF) forms. The full range of capabilities (such as electronic signature, digital signature, or importing from other forms) requires the installation of one or several proprietary software packages. Microsoft Word documents can also be used as forms, and can be digitally signed. The certificates are mostly not free[3]. In addition, such forms require the installation of large software packages. The same goes for IBM Lotus.
Online Services. Several cloud-based services (such as Google Forms) offer online polls or online collaboration on documents. However, they typically do not allow digital signing. Other services provide form creation and filling capabilities[4]. The user creates forms and sends a link to the person, who fills out the form online. However, the form creator needs to pay per form. Furthermore, the user data has to pass through a central architecture.
Open Standards. Various form description formats exist, such as XForms, XUL, Vexi, ZF, XAML, and others[5]. However, all of them either require programming experience or special software to create and fill the forms.
XML-based Solutions. Kuo et al. generate HTML forms from a XML schema, but do not allow signing the documents [3]. Boyer proposed a combination of ODF and XForms, which allows saving and sending the forms per email, filling them offline, and signing them with XML Signatures [1, 2]. However, users would need additional software to use or edit the combination of ODF and XForms.

3 Open Digital Forms

Infrastructure. We decided for a decentralized solution, where forms are not stored in a central repository, but kept and sent by the users themselves. This has two advantages: First, the users remain in total control of their data, and they remain independent of a service provider. Second, our format can be used in a grass-root fashion. It can easily complement other types of forms that are mandated by the host institution.
Format. Our forms are in HTML, because this format allows basic formatting while at the same time being human-readable. HTML can also be displayed on almost any digital device with no additional software. A form consists of a title and several sections (`DIV`s). Each section describes one particular entity of interest, such as a client, a digital library, or a licensed content. Each section

[3] `https://help.libreoffice.org/Common/Applying_Digital_Signatures`
[4] `http://www.jotform.com/`, `http://www.moreapp.com/`
[5] `https://en.wikipedia.org/wiki/List_of_user_interface_markup_languages`

contains a list of attributes of that entity, such as the birth date and address of a client, the name of a library, or the type and the price of the licensed content. Each such attribute comes with an HTML `INPUT` field, containing the actual value in the `value` field.

Semantic Annotation. Our forms are semantically annotated, so that machines can read and analyze them. We use RDFa for this purpose, the W3C standard for semantical annotation of Web documents. This makes our forms compatible with the Semantic Web and Linked Data standards. By default, we use the vocabulary from `schema.org`, which is designed, supported, and maintained by Google, Bing, Yandex, and Yahoo[6]. Each section of our document defines a new entity (by help of `typeof`), and each attribute in that section creates a new property of that entity (by help of `property`).

Signing. Our documents can be electronically signed by an image of a scanned signature. For this purpose, the form contains a button which allows the user to select the image file from their hard drive. The image is then embedded as a base-64 encoded string in a `data:image` URL directly into the HTML document. This way, the form remains self-contained, and the user has to send only a single file if they want to share the signed document.

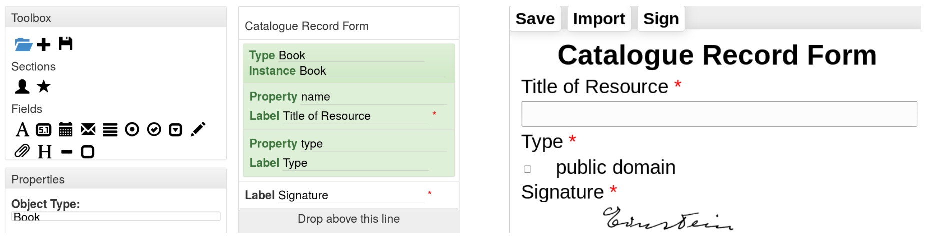
Digital signing. The forms can also be digitally signed. We opted for PGP for this purpose, because unlike S/MIME it does not require a central certification authority. PGP allows signing documents with *ASCII armor*, i.e. a plain text header and footer containing the signature. GnuPGP accepts a signed document enclosed by additional text. We sign only the content of the `HTML` tag, so that we retain a valid HTML document. CSS styling hides the ASCII armor.

4 Demo

Creating a form. Our tool for creating forms is an open source JavaScript program[7]. If Alice wants to create a form, she can just visit our Web page[8]. She can drag-and-drop section elements from the toolbar into the form (Figure 1 left). For each section, Alice can choose the section type (book, institution, etc.) from a drop-down menu from the schema.org vocabulary. She can also name each section (which creates an instance of the type in the underlying RDFa code). Alice can then drag controls such as text fields and check boxes into the sections. She can assign a caption and an attribute name to each control. We offer predefined attributes such as `birthDate`, `email`, and other properties from schema.org. Alice can also create her own property, which will then live in the local namespace of the document. In the end, Alice can download the form as a self-contained HTML document, which she can store for later modification, send by email or, if need be, print.

Filling a form. If Bob receives the form, he can display it on any HTML-enabled device (Figure 1 right). He can fill out the form by typing into the `INPUT` elements of the form. The input elements use standard HTML 5 validation to check the

[6] `http://schema.org`
[7] `https://github.com/lenguyenhaohiep/formless3`
[8] `https://rawgit.com/lenguyenhaohiep/formless3/master/`


Toolbox
Sections
Fields
Properties
Object Type: Book

Catalogue Record Form
Type Book
Instance Book
Property name
Label Title of Resource *
Property type
Label Type
Label Signature *
Drop above this line

Save Import Sign
Catalogue Record Form
Title of Resource *
Type *
☐ public domain
Signature *

Fig. 1. Our tool (left) and the resulting form (right)

format of dates, emails, etc. Filling and saving the form is done by embedded JavaScript, and does not need an Internet connection.

Importing. Since our forms are semantically annotated, Bob can also import data from other forms he already filled. This functionality is provided by a JavaScript program that is loaded from our Web page on demand if Bob clicks on the *Import* button. The program proposes a match between the existing and the incoming sections, and Bob can accept, reject, or modify this proposed match. Since the semantic annotations are taken from a standard vocabulary, such imports work also across forms from different institutions.

Signing the form. Finally, Bob can sign the document by attaching the image of a scanned physical signature to the document. This functionality is embedded in the document, and thus available offline. Bob can also add a digital signature with his private PGP key. This functionality is provided by JavaScript code that is loaded on demand. The private key never leaves the computer, and is used just to generate the signature. After this step, no modifications of the form are possible, and Bob can send around the signed document. Alice can check the validity of the signature in our tool, by providing the form and the public key of Bob. She can also verify the signature with any standard tool such as OpenPGP, Gpg4win, or GPGsuite. Since the document is self-contained and nearly human-readable, users can also always inspect the workings of the form.

5 Conclusion

Our demo proposes a free and secure format for digital forms, which is based completely on open standards. We expect our demo to appeal to the intended community of TPDL in particular, because the conference is concerned also with information integration, the use of Semantic Web standards, and supporting infrastructure for digital libraries[9], which are issues that our demo addresses.

References

1. J. M. Boyer. Interactive office documents. In *Document Engineering*, 2008.
2. J. M. Boyer, C. F. Wiecha, and R. P. Akolkar. Interactive Web Documents. *Computer Science - R&D*, 27(2), May 2012.
3. Y.-S. Kuo, N. C. Shih, L. Tseng, and H.-C. Hu. Generating form-based user interfaces for XML vocabularies. In *Document Engineering*, 2005.

[9] `http://www.tpdl2016.org/callforpapers`