Open Digital Forms

Nguyen Hao Hiep Le[1] and Fabian Suchanek[1]
hle@telecom-paristech.fr, fabian@suchanek.name

[1]Telecom ParisTech, Paris, France
February 10, 2016

Abstract

This project aims to propose a new form format based on HTML, which is readable in any browser and users are able to sign and verify forms with the often-used open standard OpenPGP. Especially, this form format has a functionality to reuse common information between forms by using RFDa. We also introduce a lightweight tool to create the form with the proposed format

1 Introduction

The use of forms is compulsory in lots of contexts. Traditionally, paper forms are widely used. Thanks to the growth of technology, we have been shifting to digital forms in order to eliminate traditional ones. Additionally, we also use digital signatures to justify forms.

However, we do not have any standard for digital forms. We normally use .doc, .pdf, .odt as formats which require users software installations. Moreover, we can have situations that we have to fill lots of forms whose differences between them are very slight. Thus, the reuse common information would be the problem to be considered.

Thus, this paper will introduce a solution to address these issues. The results of this project would be:

- A new form format based on HTML and RDFa. The form itself has basic features such as fill, sign, verify and import data.
- A JavaScript tool which allows design the form.
- No centralized architecture is required
- The form uses OpenPGP standard for signing and verifying and it is capable of other PGP tools.
- No extra installation is required, the tool and the form can work properly with a web browser.
- The tool and the form are open sources

The rest of this report will be constructed as follow: section 3 introduces the related work, section 3 represents techniques related to our work and section 4 shows how we implement the form and its creation tool. Finally, section 5 for the conclusion, pros and cons and also the future works, appears last.

2 Related Work

There are several softwares which support form representation and form features (sign, verify, fill and import data), however they exist some issues.

Commercialization PDF and Doc format could be the often-used solutions for the creation and representation of forms. However, tools support these formats (Adobe Acrobat, Microsoft Word) are commercial or the free version has limited features. They also allow sign the form with using the certificate of other vendors. Our solution, in contrast, is open source.

Platform and software dependencies There are different tools corresponding to different platforms for the same solution. Take the case of Microsoft Word. In order to read the Doc form, users have to install Microsoft Word, again it is commercial. It is true to say that a Linux user hesitates to open the Doc document with LibreOffice due to the software capacity problem. Thus, our solution aims to solve this problem, it means that our solution can work across platforms without extra installation.

Data importing Adobe Acrobat and PDF Form Filler[1] supports data importing from PDF, CSV and Text file to a form. These applications have the problem of detecting and matching fields from data sources if fields have different names. This is because the source data and destination form could use different vocabulary for data description. Thus, our solution will use schema.org vocabulary as the common data description so that forms can understand each other.

3 Technique description

In this section, we introduce main techniques for implementing our plugin.

HTML is the markup language standard for web pages and it is visualized by browsers. It can embed script languages such as JavaScript to manipulate the behavior of HTML web pages. It also supports CSS for the style of HTML elements.

It is true to say that every Internet user has to install a browser (normally installed along with operating system) for Internet access. Thus, applying HTML format as the form format could reduce the complexity of software dependency. This is to say that users do not need extra software just for filling and reading a form. As a result, HTML-based forms could works in multiple operating systems.

In fact, JSON or XML could be alternatives to express a form, but these formats are better for storing data rather than displaying data. As a result, HTML format outweighs JSON and XML.

RFDa Resource Description Framework in Attributes [11] is a standard that defines new attributes to be embedded in the HTML code. Values of these attributes define the semantic information. Owings to RDFa, computer can understand information inside the HTML code.

[1] https://www.pdfill.com/pdf_form_filler.html

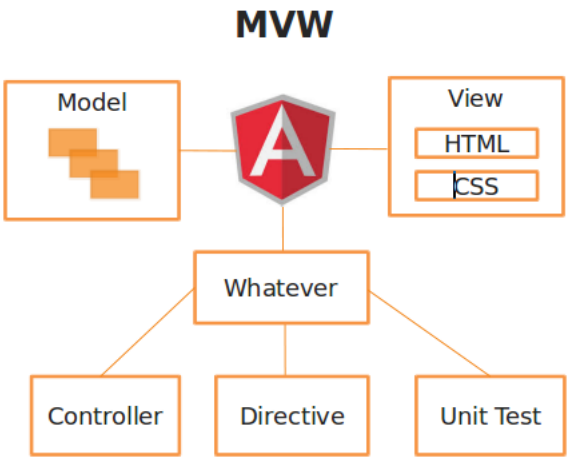

Figure 1: The overview of the plugin based on MVW

[Source: http://www.edureka.co/blog/angularjs-superheroic-javascript-mvw-framework]

Schema.org This is structured data description in the internet, web page, email and so on. It contains vocabularies covering entities, relationships between entities and actions.

These shared vocabularies aim to structure the information so that computer can understand the unstructured data in Internet and also enrich the knowledge from many sources. Schema.org is sponsored by Google, Microsoft, Yahoo and Yandex.

Thus, we use this schema to express and integrate semantic information into forms so that every application, which can understand RDFa, can reuse this information. Notice that schema.org is only constructed in English.

OpenPGP is a standard for data encryption and decryption. It is often used for signing, encrypting, and decrypting e-mails to ensure the security thought networks. It uses public-private key pair with asymmetric encryption algorithms such as DSA or RSA (more details at [12] and [5]).

The OpenPGP [9] also defines standard formats for encrypted messages, digital signatures and certificates for exchanging keys, it is an open-source standard for public use and it is approved by the Internet Engineering Task Force, an organization who develops and promotes Internet standards.

AngularJS is an open-source JavaScript framework for web development introduced by Google [2]. It extends HTML attributes to manipulate DOM elements and it also supports two-way data binding between a JavaScript variable and the value of an HTML element.

It works at client-side with the pattern Model-View-Whatever (shown as Fig.1) to separate data, presentation and logic components. As we can see, the AngularJS engine plays the intermediate role between View, Model and Whatever.

View consists of HTML pages which contain DOM elements. Each DOM element can be assigned with a model which represents its data by a Directive.

Model is any JavaScript variables with in-memory data when a website is running. AngularJS supports two way data binding, which is the automatic synchronization of data between Views and Models, any change of data in Model will reflect View and vice versa.

The data of Model can be changed by user interaction with DOM element (enter the value, select from a dropdown list or check a value) or by a Whatever.

Whatever which could be Controller or Directive.

Controller handles requests from View and processes Model data. For instance, when a user clicks a button to show the value from an input text, there is a request from View to Controller, and Controller will read the value of Model assigned to input text and then show this value in View. Controllers aims to process data.

Directive defines HTML components which are used in Views. Directives are markers on DOM element (name, attribue, tag-name) to match DOM Element to a Model or to attach DOM behaviors.

AngularJS outperforms JQuery in term of two-way data binding

4 Implementation

4.1 Form format

Form format is a valid HTML file with the extension of RFDa. It contains sections (div) as objects of interest of forms. Each section contains HTML controls (input, text-area, checkbox, radio box or select) as their properties. Fig.2 shows an example of a form displayed in the web browser.

Form contains semantic annotation (RDFa) in which a section is an object (typeof attribute) and form controls inside are its properties (property attribute). The Fig. 3 shows an example of semantic information validated and extracted by RDFa validator, as we can see namespace *ns*1 indicates *schema.org* meanwhile objects and properties which do not belong to *schema.org* have the namespace *ov*.

Form itself is basically fillable and able to be saved after filling. However, it can support advanced features such as import data, sign and verify on demand if users have the Internet connection because advanced features heavily depend on external JavaScripts sources.

Form just needs Internet connection to load external scripts into local computer, however, all features run locally without any data exchange through network so that user can control their own data.

Form is one-file-only, it means that all images attached will lie inside the form. In this case, the image data will be convert to base:64 format and lives inside the form.

Form can be signed and verified by using OpenPGP standard.

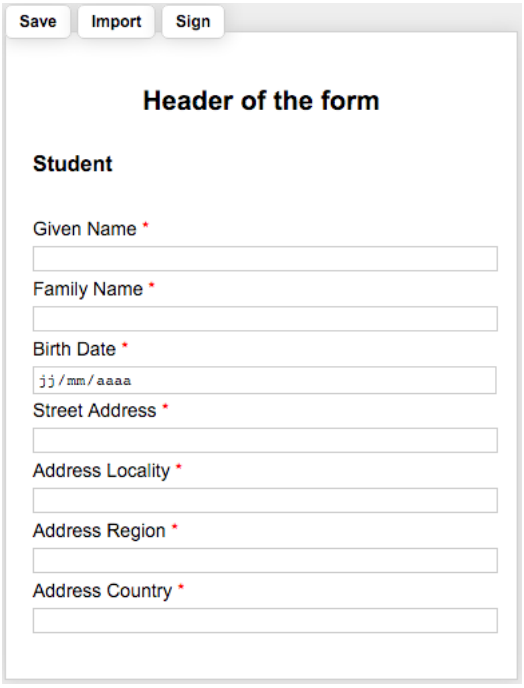

Figure 2: An example of Form

```
@prefix ns1: <http://schema.org/> .
@prefix ns2: <http://www.w3.org/ns/rdfa#> .
@prefix ov: <http://personal.schema.example/> .

<> ns2:usesVocabulary ns1: .

<Project2> a ns1:Thing;
    ov:ProjectEnreprise "Non";
    ov:UERequise "INF228";
    ov:semestre "S1";
    ov:year "2015-2016";
    ns1:description "Details";
    ns1:name "Documents sans papier - Formless";
    ns1:option "Project PRIM" .

<Student1> a ns1:Person;
    ov:cursus "24 mois (ASTM)";
    ns1:familyName "LE";
    ns1:givenName "Nguyen Hao Hiep" .

<Teacher3> a ns1:Person;
    ov:date "";
    ov:signature <
    ns1:familyName "Fabian Suchanek" .
```

Figure 3: An example of semantic information validated and extracted by RDFa Validator

4.2 Form creation tool

We also build a tool to create our proposed form format. This tool could be run as a website[2] or user can download and use as a standalone tool. Fig.4 shows the main

[2]https://rawgit.com/lenguyenhaohiep/formless3/master/

screen of the tool. The repository of this project can be found at [7].

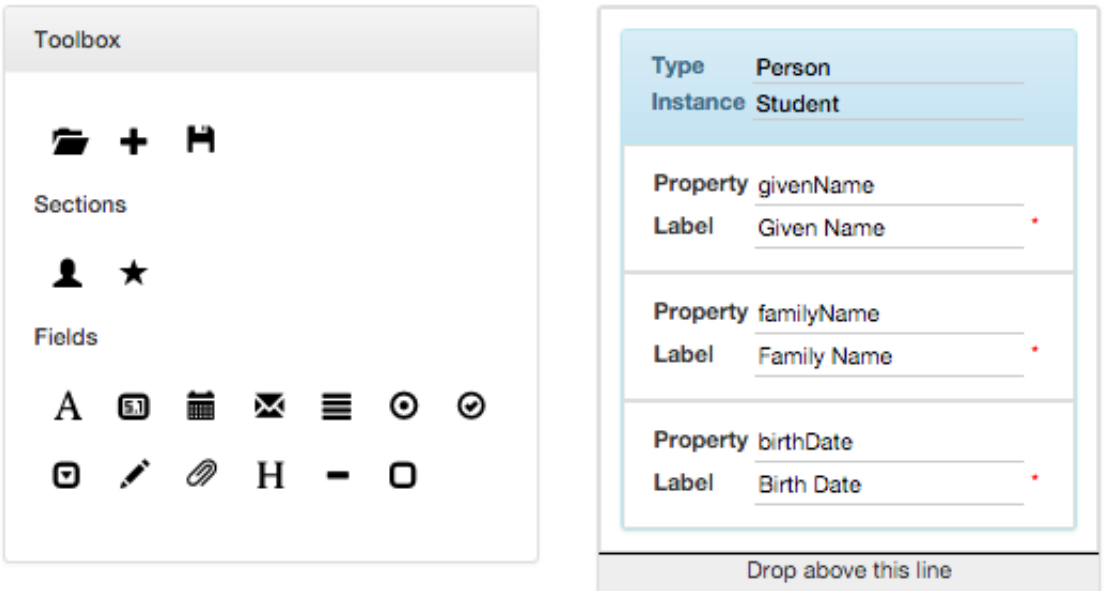

Figure 4: Main screen which a working zone in the center and a toolbar which provides sections and fields for the creation of form

In order to create a form, users just need to drag and drop any component into the drop-zone. There are three types of components:

Section The object of interest in the form, for example, a student, a project or a teacher. Normally, we have two main types *Person* and *Thing*, and others derived from these two types.

Section has two properties which are *type* and *instance*. *Type* is any class supported by schema.org, if you cannot find any class that fits your need, just mark it as *Person* or *Thing* and then provide the *Instance*. For instance, we do not have class *Student*, thus just create a *Person* with an instance *Student*. The value of *Instance* displays as the section title in the form as well.

PropObject The property of an object as an object representation. For instance, property *address* of *Person* has type of *PostalAddress* which consists of several properties such as *addressCountry*, *addressLocality*, *addressRegion* and *streetAddress*.

PropObject has 2 properties: *property* of *Object* and *Type* which is the class of this *PropObject*. If property does not match any property provided in schema.org, it will be processed as in a different namespace.

Field Any form component such as text, number, dropdown, checkbox, radio box, file and so on. It contains an HTML component (text, select, checkbox, etc) and a *property* of *Object* to which it belongs.

It is noticeable that *PropObject* and *Field* have to be placed inside a *Section* so that RDFa can be integrated and vice versa. *Section* can be nested but *PropObject* and

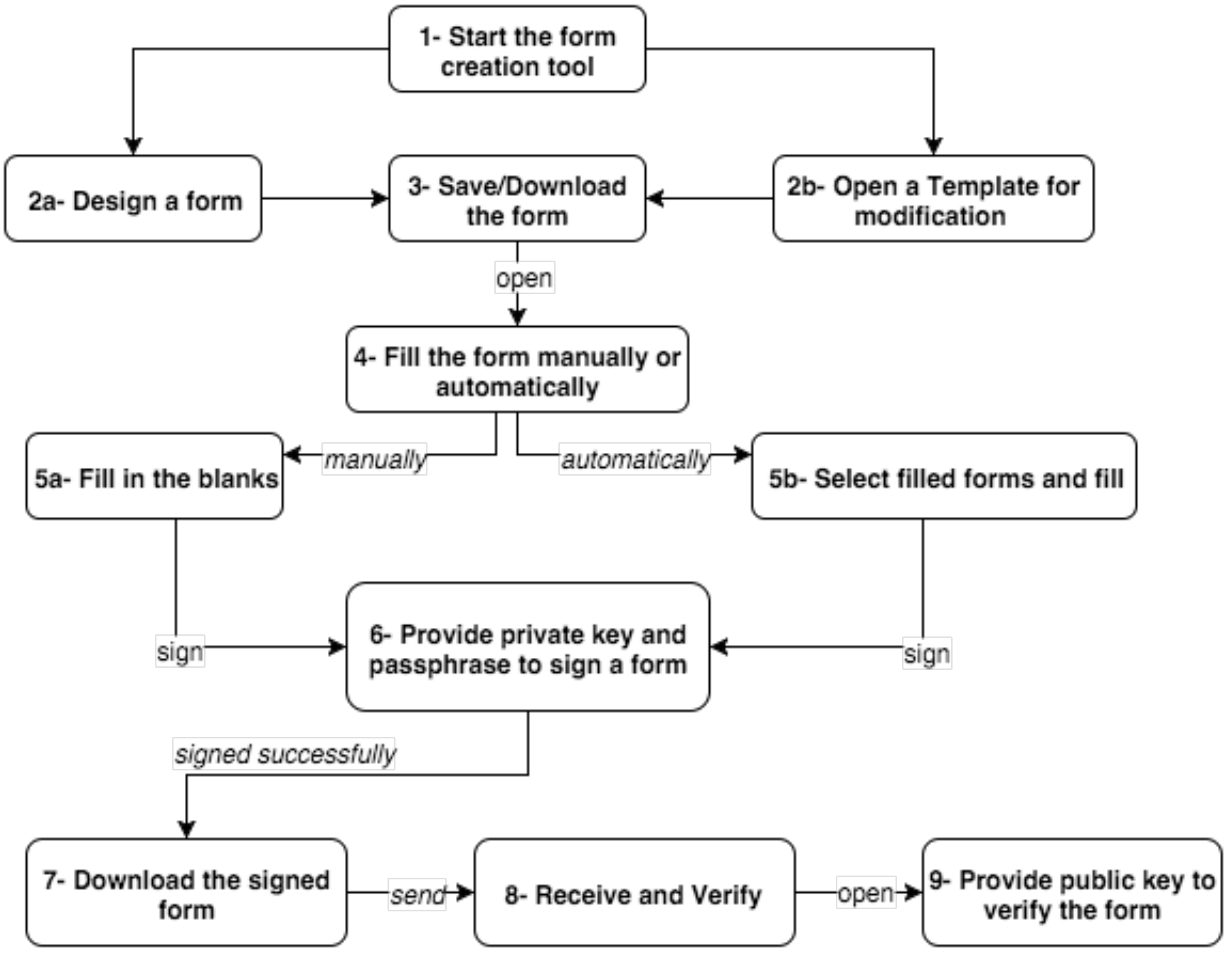

Figure 5: The workflow of using the plugin

Field belong only to the nearest *Section* which holds them. The example below shows a *Person* with *familyName*, *givenName*, *birthDate* and *address*.

4.3 Scenario

Imagine a context that a secretary wants to send a form to a colleague, what is she going to do?

1. She firstly has to create a form, precisely a template
2. She then fills in the blanks
3. She signs the form by her real signature or her digital signature
4. She finally sends it to her colleague by mail or by hand
5. Her colleague will verify the form by looking at her signature or by validating her digital signature

Our plugin will work as the same nature like this, the workflow is illustrated in Fig.5

5 Conclusion and future works

In this report, we introduce a new form format which is based on HTML with abilities to fill, sign, verify and import data. We also introduce a tool to create forms with this format.

On the one hand, the form has enough basic features and the form is one-file-only, which means that all attached documents will go along with the HTML file. By using OpenPGP standard, users can use any application to sign/verify forms and our plugin can verify/sign them. We do not need to install any software except for only web browser with JavaScript. It is open-source as well.

On the other hand, the form output is not enough well styled and it does not have the same capacity in different browsers. Additionally, we cannot verify the authentication of the owner of the public key (certificate verification). The form supports only attached images. In terms of performance, the signing process is slower than GnuPG tool because it uses only JavaScript to performance, but the latency is acceptable. The form needs to load external scripts on demand for advanced features.

Actually, there could be lots of unexpected bugs due to lack of a full developing and testing procedure. The number of form controls is limited and we do not have advanced controls such as table or file uploaded (just images in this current version).

In the near future, we tend to add the missing features and also fix encountered bugs.

References

[1] Angular directives that allow you to build sortable lists with the native html5 drag and drop api. https://github.com/marceljuenemann/angular-drag-and-drop-lists.

[2] Angularjs. https://angularjs.org/.

[3] Beautify javascript or html. http://jsbeautifier.org/.

[4] Bootstrap components written in pure angularjs by the angularui team code on github. https://angular-ui.github.io/bootstrap/.

[5] Digital signature algorithm. https://en.wikipedia.org/wiki/Digital_Signature_Algorithm.

[6] File saver. http://eligrey.com.

[7] Formless project. https://github.com/lenguyenhaohiep/formless3.

[8] Html, css, and js framework for developing responsive, mobile first projects on the web. http://getbootstrap.com/.

[9] The openpgp alliance. http://www.openpgp.org.

[10] Openpgp javascript implementation. http://openpgpjs.org/.

[11] Rich structured data markup for web documents. https://www.w3.org/TR/xhtml-rdfa-primer/.

[12] Rsa (cryptosystem). https://en.wikipedia.org/wiki/RSA_(cryptosystem).

Appendices

A Code reuse detail

In this plugin, we modify and inherit some open source JavaScript codes under MIT License, which are

JS Beautifier [3] is a library to format beautifully HTML code.

FileSaver.js [6] is a JavaScript code which allows users download an HTML page.

Openpgp.js [10] is a library which provides encryption, description, signing and verifying functions under the OpenPGP format.

Angular-drag-and-drop-lists v1.3.0 [1] is an AngularJS directive that allows drag and drop DOM elements into a zone. The original version is very simple and we adapt this directive so that it fits our need in this plugin.

AngularJS [2] The framework core which supports MVC pattern in the client side.

Bootstrap [8] An open-source HTML template along with CSS and JavaScript for interactions.

UI Bootstrap [4] Bootstrap components with AngularJS