A Web-Based Submission System for Meeting Abstracts

Hu Wang
American Institute of Physics
1NO1
2 Huntington Quadrangle
Melville, NY 11747
hwang@aip.org

Abstract

As one of the services provided to AIP's Member Societies, we publish program books and abstract books for meetings sponsored by the societies. In this paper, we describe a Web-based client-server abstract submission system. It uses HTML forms to gather and validate user input of abstracts and related information, provide on-line proof before officially submitting, and store the input in LaTeX files and a database. From the publishing production's point of view, the major benefit of such a system is the greatly improved quality of well-structured submissions, which makes it possible to streamline the whole production cycle. Some possibilities with its integration into a database publishing system are briefly discussed afterwards.

Introduction

The American Institute of Physics (AIP) publishes program books and abstract books for some of its member societies' meetings. The last few years have seen many changes in the ways some societies' meeting abstracts are submitted and published.

At the beginning, hardcopy abstracts prepared in various text formatting software were mailed in and published with the cut-and-paste method. It was non-digital, inefficient, and the resulting quality of finished products was understandably poor.

Then, with the popularity of LaTeX in scientific communities and the widespread availability of email, came email-based electronic submissions. Authors would download an appropriate LaTeX template file that included predefined tags (LaTeX commands and environments), fill it out, and email it to a designated address. A program would collect the submissions and save them as individual LaTeX files.

The email approach had the potential advantages that all files were standardized, well tagged, and the publishing process could be highly automated. Unfortunately, since there are still many authors who are not familiar with or do not have LaTeX, many submissions had to be manually corrected for syntax errors by the production staff, which was time consuming and sometimes impractical. This often led to syntactically invalid submissions, which in turn prohibited auto-processing. Another drawback was that the process was not interactive. As for authors, those without LaTeX could not proofread their abstracts, and figures could not be easily submitted with the abstracts.

We needed to develop a system that could take advantage of the potential strength of the email-based method while avoiding its shortcomings, namely, the non-interactiveness and lack of control on the quality of submissions.

A Web-based system

As the World Wide Web spreads around the world (for some society meetings, over one third of the contributed papers come from abroad), it lends itself naturally to a solution of our problems.

We have developed an automated, Web-based, client-server meeting abstract submission system with the following features:

- interactive user interfaces via HTML forms
- on-line LaTeX help info
- choice of LaTeX or non-LaTeX entry methods
- uploading figures with abstracts
- on-line proof viewing and syntax validation
- editable abstracts
- easy integration with publishing phases

Regardless of the input method chosen by the authors, each submission results in a well-tagged LaTeX file that is free of syntax errors, a valid HTML file into which the input data is injected, and updated database records.

LaTeX is used as the ultimate file format for the abstract collecting phase for two reasons: it fits

well with our existing production systems and it is a widely accepted standard for publishing scientific and technical documents. In fact, the on-line proof is produced by LaTeX, along with `dvips`, and some PostScript-to-gif conversion program.

System requirements

Client: Internet connection and a conventional Web browser.
Server: Web server, LaTeX, `dvips`, Image Alchemy or `ps2gif`, GhostScript, and CGI scripts.

In the following section, we describe the implementation of this system.

Implementation

A user accesses the system by using a Web browser to connect to a designated URL and entering the meeting password included in the calls for paper distributed to the society members.

Once logged in, the user indicates whether he or she is entering a new abstract or wishes to edit a previously submitted abstract by clicking on the appropriate radio button. The former choice prompts the author to indicate, via radio buttons, the total number of mailing addresses needed for all the authors of the abstract and the total number of figures (up to 2) in the abstract. An appropriate template is then displayed for the user to fill out. The latter choice requires the user to enter the abstract number and the PIN that were both assigned and emailed to the author by the system when the abstract was initially submitted. In this case, their filled-out template will be shown for editing. Authors who forget their abstracts numbers and PINs can query the system and the results will be emailed back to them.

Template. One of the key components of the system is the Web template. The elements from this HTML form are captured for insertion into the database and are also tagged for the LaTeX file that results from completing and submitting the template form.

To accommodate users who are unfamiliar with or do not need LaTeX, the top of the template has two radio buttons labeled "Straight Text" (i.e. non-LaTeX) and "LaTeX", respectively, for choosing the method of entry. With the former method, no LaTeX commands are recognized because everything entered will be treated literally, which means it is impossible to choose fonts or set math expressions. The usual LaTeX commands are allowed with the LaTeX entry method — except for `\thanks`, `\footnote`, `\begin{center}` and the like.

The template has hyperlinks to sample inputs for both entry methods, and to LaTeX help on how to input symbols and mathematical expressions. When clicked, these links open separate windows for easy reference.

The following form elements are used in the template:

- radio buttons for choosing the entry method
- presenting author's name
- corresponding author's name, address, country, phone, email address, fax
- title and short title
- author's name and address
- `<textarea>` for abstract body
- file selection fields for uploading figures (if any)
- figure caption (if any)
- topics of paper
- requested presentation method
- hidden fields

Hidden fields are for passing state variables information from one invocation of a CGI script to the next. They are embedded in the template to identify the client session, to indicate if it's a new or previously submitted abstract, as well as the number of figures and author groups in the abstract.

CGI scripts. Processing of submitted forms is handled by CGI scripts, which are programs communicating (via the Web server) with clients. The essential scripts are those that deal with filled templates and proof screens. They perform the following tasks sequentially:

1. Validate the form and figures (PostScript or Tiff only, if any). This checks if any required input fields are empty, if illegal input is found (e.g. no TeX or LaTeX commands are allowed in the Straight Text entry method, `\thanks` is not allowed in either method, and so on.), and if the uploaded figure is a valid PostScript file (the PostScript language interpreter Ghostscript is used for this purpose) or a Tiff file.

   Any input that fails to pass the validation will cause an appropriate error message to be displayed.

   Here, JavaScript, which is faster on the client side, can be used for form validation also.
2. Build a temporary LaTeX file and process it. If there is no syntax error, run `dvips` and Alchemy to generate the gif image and display the image to the user for proof. If there are syntax errors, extract the error message from the log

file, display it, and ask the user to go back to the template screen and fix the LaTeX error.

We choose the gif format for proofing because it is accessible without requiring browser plug-ins or help applications. Of course, the PDF format may be used for proofing, which requires the Adobe Acrobat Reader for viewing.

In order to show the user the dynamically generated gif, we must prevent browser caching from displaying the old gif. This can be done by proper HTTP headers such as 'Cache-Control: no-cache' or 'Cache-Control: no-store'. The gif file for a typical abstract is about 20Kb in size.

Building LaTeX files via the Straight Text method warrants a little note here, as whatever the user has entered will be treated as literal. The CGI script must handle the following TeX special characters carefully to properly escape them:

```
# $ % & _ { } ~ ^ \ < > |
```

For example, author's input `<` should be converted to `$<$` in the LaTeX file; otherwise, unexpected output will result even though LaTeX does not complain.

It's rather straightforward to build LaTeX files with the LaTeX entry method — just insert the author input data into the arguments of appropriate LaTeX control sequences. Not surprisingly, the LaTeX file's tags correspond nicely to the template's elements. For instance, these tags are used:

- `\pauthor`
- `\cauthor`
- `\caddress`
- `\cphone`
- `\cemailaddr`
- `\cfax`
- `\title`
- `\author`
- `\authaddress`
- `\begin{abstract}`
- `\caption`
- `\category`
- `\pmethod`

Let us now demonstrate the relationship between author input in each entry method and the resulting LaTeX file.

The Straight Text method. In this sample, the user faces a screen with the prompt "Title" on the left and a blank box. The title text is input literally, like this:

Title: `Straight Text Mode #$% Title`

which will be converted into the following in the resulting LaTeX file:

```
\title{Straight Text Mode \#\$\% Title}
```

Notice that the literal input of the special characters `#$%` has been correctly converted with the addition of backslashes.

The LaTeX method. By selecting this method, authors can directly input LaTeX code. In this sample, the author is facing a similar screen, with "Presenting Author" as the prompt to the left of the boxed area for the name:

First name: `P\'al` *Last name:* `Kn\"oll`

will result in this statement in the LaTeX file:

```
\pauthor{P\'al}{Kn\"oll}
```

3. Build a temporary HTML file into which the user-input data is inserted. This file will be needed if the user wants to edit a previously submitted abstract later on.

   Here, care is needed too. First, every `"` character entered by the user must be replaced in the resulting HTML file by the *entitized* version, i.e. by `&quot;` so that it does not interfere with the HTML form's element value delimiter `"`. Second, any scrolling list element in the template must be placed in the HTML file and a `SELECTED` attribute inserted inside the `<option>` that has been selected by the user.

   For instance, the previous example of entering the presenting author name would yield the following lines in the HTML file:

```
First name:<input name=p_fname
           type=text value="P\'al">
Last name:<input name=p_lname
     type=text value="Kn\&quot;oll">
```

   Please note, when displaying an HTML file, the browser replaces all entity references such as the above one with the corresponding characters. Therefore, the CGI script that processes the submitted form does not need to *de-entitize* any form data.
4. Insert relevant information into the database. This may be needed later for searching, reporting, mailing label printing, etc.

   After the LaTeX source file is error-free and a proof of its output is displayed, the user may decide to finally submit it to the system or may go back to the template to massage it further

and then go through the proof and final submission cycle again.

5. Final submit. For a new submission, assign a new sequential abstract number; for a re-submission, extract its abstract number from the corresponding hidden field. In either case, rename the source file, the HTML file, and any figure files to the abstract number with proper extensions, email the abstract number and a randomly generated PIN to the corresponding email address, and insert or update the database records accordingly.

Conclusions

This system offers many benefits to both users and the production staff.

Users' benefits. Self-evident HTML form templates, choice of entry methods, easy figure handling, LaTeX syntax validation (this could be a mixed bag for LaTeX newbies because the error messages may be too cryptic for them), on-line proof viewing, and editing previously submitted abstracts.

Production benefits. Since the rigorous validation processes shift more responsibilities to the users, the abstract submission quality is greatly improved. The production staff now start the game with standardized, well-structured data, which makes it possible for the subsequent events to be highly automated. In fact, this system can be expanded into a comprehensive database publishing system for handing the following conferences-related tasks:

- During the period of collecting abstract submissions, set up a cron job for the conference coordinator to print out abstracts and accompanying figures received the previous day, to generate a sort-by-category and sort-by-presenting-author list of abstracts received so far.
- For program committee use: abstract database search for various fields.
- Insert into the database the program committee's decisions of acceptance and rejection, as well as the meeting sessions information (such as the session name, schedule, location, chairperson, invited talks, contributed talks, posters, time slot for each presentation, etc.).
- Generate letters of acceptance and rejection, as well as mailing labels.
- Generate program books and abstract books.
- Searchable on-line program listing and viewing.

Acknowledgements

Several people at the AIP were involved in this project: Don Lang, Chris Hamlin, and Kevin McGrath. My thanks to all of them — especially Chris Hamlin, from whom I have learned many TeXniques.

I would also like to thank the TUG99 reviewers and the Proceedings editor, Christina Thiele, for their constructive suggestions and comments.