Web applications with the Tcl web server

Delivery Service

Programmed in Tcl, the Tclhttpd web server is an ideal platform for advanced web applications that can also profit from the ease and speed of development that Tcl offers. Tcl library functions and numerous extensions are available including a library that takes the hard work out of generating HTML code.

BY CARSTEN ZERBST

Deutsche Post World Net

A number of techniques are now available for web applications in Tcl. In addition to CGI scripts, Tcl modules for Apache or fully featured application servers, you might like to take a look at Tclhttpd. The webserver is a 100 per cent Tcl code development with a long history, and this is reflected in the current version. The Tcl webserver's functions provide an ideal platform for advanced web applications. This article demonstrates various approaches for the creation of HTML pages and also in the processing of requests.

Tclhttpd's origins go back to 175 lines of Tcl that Brent Welch wrote in the mid 90s. The codebase has grown since to encompass something in the region of 12,000 lines, and this does not include the extensive Tcllib library. This stable codebase supports speedy deployment in various areas. Tclhttpd can:

- Serve static web sites
- Run Server Side Includes
- Link individual URLs, even whole directories, or various MIME types with Tcl scripts
- Embed Tcl code in HTML
- Read and write cookies
- Manage sessions
- Authenticate users
- Evaluate forms
- Upload files to servers
- Support email

The development activities were never intended to rival the king of the hill, Apache. If you are having to contend with several hundred requests per second, Tcl modules such as *mod_tcl* [5] or *mod_websh* [6] for Apache are definitely a better bet. But if you are looking to develop web applications for small to medium volume web sites, Tclhttpd will provide you with a solid base from which you can work.

Suitable for projects of all sizes

But Tclhttpd does not need to hide its light under a bushel – after all it does host *http://www.tcl.tk* and this website has to cope with a considerable volume of traffic. Other reference applications include a global network for meteorological data from airports or the Medusa Project [4] that accesses a large-scale database. But if you talk to the users, you will normally find that Tcl has mainly been used for internal projects.

Sourceforge has the Tclhttpd source files [1]. There are two versions: the all-inclusive variant *tclhttpd-3.2-dist* including Tcl, Thread and Tcllib [3] and the current version *tclhttpd-3.3.1*. The older package's advantage is ease of installation. The following steps are all that you need to do to get a web server up and running:

```
# tar -xzf tclhttpd3.2-dist.tar➥
.gz
# cd tclhttpd3.2-dist/tclhttpd3➥
.2
# make
# make install
# cd bin
# wish httpd.tcl
Running with 256 file ➥
descriptor limit
httpd started on port 8015
```

Just launch your browser and point it at *http://localhost:8015* to view the sample files that show you some of the package's capabilities. You can configure the server via the *tclhttpd .rc* file; the following is a listing that contains an example of some of the options available.

```
# Sample configuration
# httpd running as user 500
# in group 100
Config uid 500
Config gid 100

# httpd listening on port 8015,➥
  normal hostname
Config host     [info hostname]
Config port     8015

# Custom scripts in
# .../custom directory
Config library  [file join ➥
[Config home] .. custom]

# HTML files in
# /usr/local/httpd/htdocs
Config docRoot  /usr/local/➥
httpd/htdocs
```

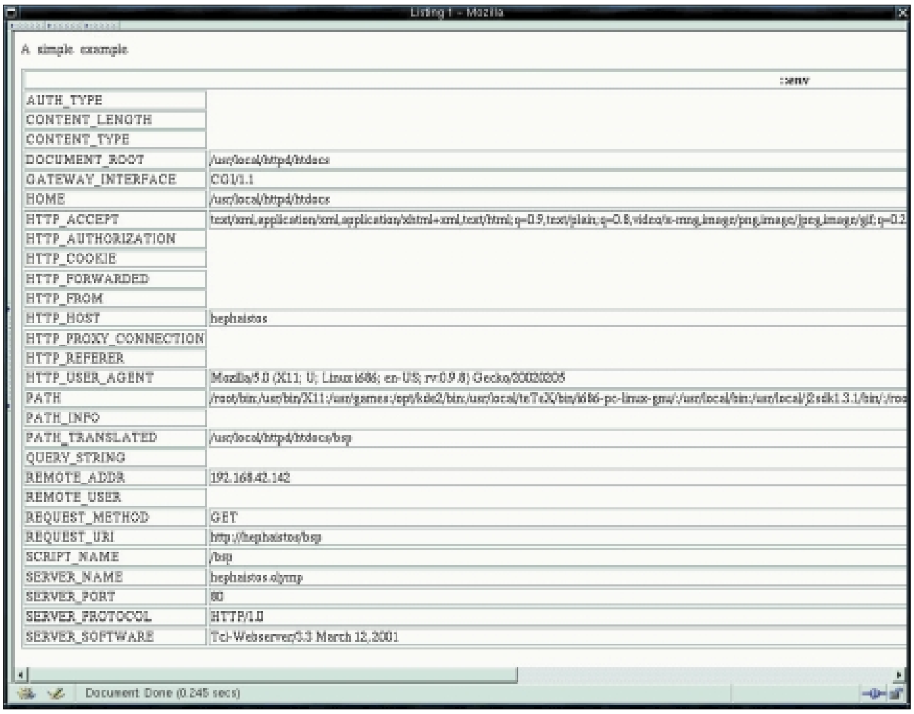
```
# Do not create threads
Config threads  0
Config main     [file join ➥
[Config home] httpdthread.tcl]

# Logfile: /usr/local/httpd/log
Config LogFile  /usr/local/➥
httpd/log

Config LogFlushMinutes 0
```

When you start out on a development project, it makes sense to use the content in the sample directory, to leverage the control panel and statistics features. The control panel reads variables from browsers or reloads the libraries, and both these functions are very useful for when it comes to debugging.

Tclhttpd can create web page content dynamically at runtime and it supports various approaches to do so. The easiest way to go is to configure a *Direct_Url*: The server will then pass requests for the URL to the configured Tcl procedure. In contrast to a CGI script the server will not spawn a new process but it will run the procedure directly in its own server process. This allows you to use variables from the server for counters or to open database connections.

Direct Url Dynamics

The next listing shows a simple example.



A simple example

	::env
AUTH_TYPE	
CONTENT_LENGTH	
CONTENT_TYPE	
DOCUMENT_ROOT	/usr/local/httpd/htdocs
GATEWAY_INTERFACE	CGI/1.1
HOME	/usr/local/httpd/htdocs
HTTP_ACCEPT	text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2
HTTP_AUTHORIZATION	
HTTP_COOKIE	
HTTP_FORWARDED	
HTTP_FROM	
HTTP_HOST	hephaistos
HTTP_PROXY_CONNECTION	
HTTP_REFERER	
HTTP_USER_AGENT	Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.8) Gecko/20020205
PATH	/root/bin:/usr/bin/X11:/usr/games:/opt/kde2/bin:/usr/local/teTeX/bin/i686-pc-linux-gnu/:/usr/local/bin:/usr/local/j2sdk1.3.1/bin/:/roo
PATH_INFO	
PATH_TRANSLATED	/usr/local/httpd/htdocs/bsp
QUERY_STRING	
REMOTE_ADDR	192.168.42.142
REMOTE_USER	
REQUEST_METHOD	GET
REQUEST_URI	http://hephaistos/bsp
SCRIPT_NAME	/bsp
SERVER_NAME	hephaistos.olymp
SERVER_PORT	80
SERVER_PROTOCOL	HTTP/1.0
SERVER_SOFTWARE	Tcl-Webserver/3.3 March 12, 2001

Figure 1: The procedure detailed in Listing†2 outputs the content of the global Tcl variable *::env*. The variable contains entries that you should recognize from CGI scripts, such as *HTTP_USER_AGENT*.

The *Direct_Url /listing2 .html listing2* command assigns the URL *http:// localhost:8015/listing2.html* to the *listing2* procedure. The Tcl procedure creates and returns the required page. The variables available in the script, for example *env*, are interesting, as they are being used to store information on the current client connection. The *html::tableFromArray* command formats the content of the global variable producing the result shown in Figure 1.

```
Direct_Url /listing2.html ➥
listing2

proc listing2 {args} {
  puts stderr $args

  set html    "<html>"
  append html "<body>"
  append html [html::tableFrom➥
  Array ::env "border=1" *]
  append html "</body>➥
  </html>"
  return $html
}
```

The script must be stored in the *contrib* directory in order for the server to find it. Tclhttpd reads all the scripts in this directory automatically at startup.

During the development phase the *lib* directory is also useful. Scripts stored in this directory still need to be loaded explicitly in the main script, but this allows you to reload them later from the control panel with the *Reload Source* function, which provides a facility for on the fly code modification. If an error is discovered in a script the Tclhttpd displays the debugging information directly as an HTML page for easy viewing.

Elegant Templates

Templates are a more elegant solution than using Direct Url and comprise an HTML document with embedded Tcl code. The Tcl elements are encapsulated in brackets – the return value of the function that immediately precedes the closing bracket is passed to the HTML page. You can use the value of the variable in the whole template and not only in the Tcl elements.

If you want to work with templates, you will need to place a copy of the *.tml* file and the *libtml* directory taken from the distribution in the *htdocs* directory. The sample template in the next listing first defines a variable, that contains the value of the *env* in HTML code. Lower down in the template, the content is then integrated into the page using *$later*. The last section of the template contains the date of the last modification. We can see that a template can therefore encompass a varying mixture of scripts, variables and HTML.

```
<html>
  <head> <title> Simple➘
  Template</title> </head>

  <body>
    A simple template.
    [ set later "watch this"
      html::tableFromArray ➘
      ::env "border=1" *
    ]
    <p>
      $later
    <hr>

    Last change
    [clock format [file mtime ➘
    $::env(PATH_TRANSLATED)]]
  </body>
</html>
```

Templates use the .tml file suffix and are stored just like normal HTML documents in htdocs. When a browser requests the listing3.html, the server first actions the listing3.tml template and then returns the result. Tclhttpd addtionally writes the result to listing3.html on the hard disk and uses it for any further requests. This kind of caching is particularly practical for templates that either perform some complex calculations or contain slow database queries.

Tclhttpd automatically updates the saved version if the template is newer than the saved results, or if the browser calls the template directly. As you can see in the following listing, the *[Doc_Dynamic]* can be used to suppress the caching functionality.

Templates provide an easy migration path that you can use to gradually upgrade static web sites with dynamic functions. In addition to simple counters, navigation toolbars would be obvious candidates, since a single procedure could create them for the whole site. htdocs/libtml/sunscript.tcl contains a sample of source code from the former Sunscript page.

Interactive Templates

Web pages with user input are the next hurdle for a web application to overcome. The interactions of this kind basically comprise of two elements: an HTML page provides the user interface in the browser and a script running on the server that evaluates the input. It makes sense for a template to create the form and evaluate it, allowing the template in turn to return a modified version of the formula in case of input errors. In case of valid input, the template would then transfer the browser to a different page.

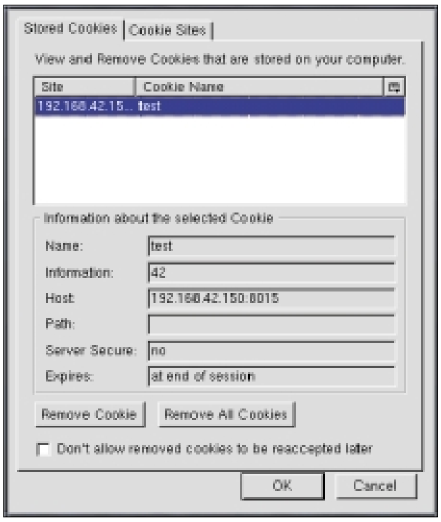
```
<html>
  <head> <title>Entries<➘
  /title></head>
  <body>
   [Doc_Dynamic]

   [ if {![ncgi::empty ➘
   project] } {
     Doc_Redirect [ncgi::value➘
     project].html?[ncgi::query]
    } else {
      set message "no project ➘
      selected"
     }
   ]
   <hr>
   $message
   <form action=$page(url) ➘
   method=POST>
    Text: <input type=text
    [html::formValue text]> <br>
     Project: [html::radioSet ➘
     project { } {
       "Project 1" project1
       "Project 2" project2
     }] <p>
     <input value="Send" ➘
     type=submit>
    </form> <p>

    Input was:
    [html::tableFromList ➘
    [ncgi::nvlist] "border=1"]
  </body>
</html>
```

The sample script in the listing above shows how a template can edit a form. Although the template is short, it provides heaps of functionality. First the [Doc _Dynamic] command prevents the form from caching the template, which would make no sense at all.

The next Tcl block handles the data input using a few functions from the Ncgi package in Tcllib. For example, *ncgi::empty* checks whether an entry for the *project* field in the form exists. In this case the request is passed via *Doc_Redi-*

TABLE 1: INSTRUCTIONS FOR HTML AND CGI

Instruction	Meaning
HTML	
html::h1 Title	Produces a heading, also "html::h2" and "html::h3"
html::tableFromArray ArrayName	Produces a HTML table from a Tcl array
html::checkbox Name Value	Produces a Checkbox
html::trxtInput Name Parameter	Produces a text input
Ncgi	
ncgi::cookie Cookie	Returns a list of values for Cookie
ncgi::setCookie - name Name -Value Value	Sets a Cookie
ncgi::empty Name	Indicates whether an input value is present
ncgi::value Key	Returns the CGI value identified by Key
ncgi::nvlist	Returns all the query as a name, value list
ncgi::query	Returns the raw query data



Site	Cookie Name
192.168.42.15...	test

Information about the selected Cookie

Name:	test
Information:	42
Host	192.168.42.150:8015
Path:	
Server Secure:	no
Expires:	at end of session

□ Don't allow removed cookies to be reaccepted later

Figure 2: Mozilla displaying the content of a cookie that originated in the template of the "cookie" listing above. The cookie was set by the server at *192.168.42.150*, is called *test* and contains the value *42*.

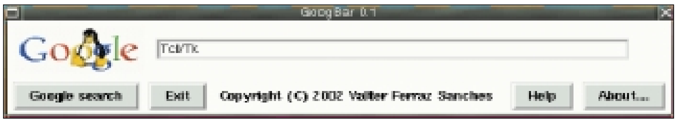


Figure 3: The googbar looks like the Google home page, but in fact it is a Tcl program that redirects search operations directly to Google.

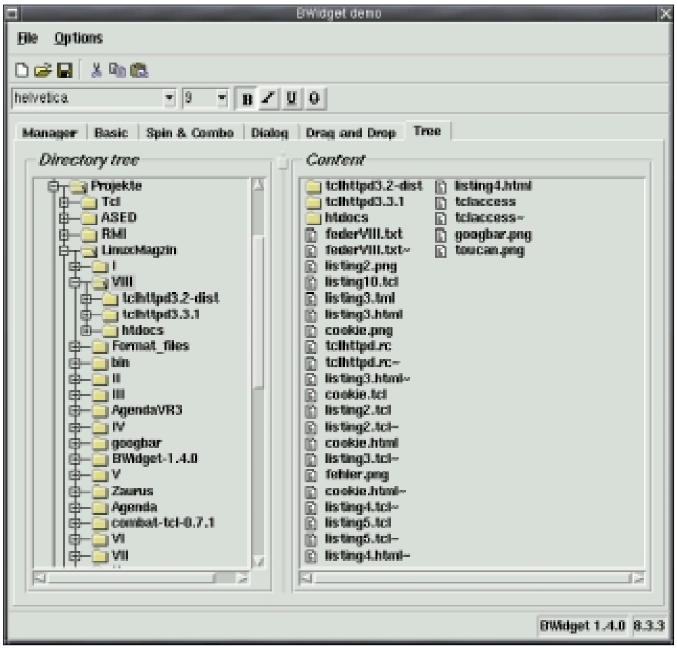
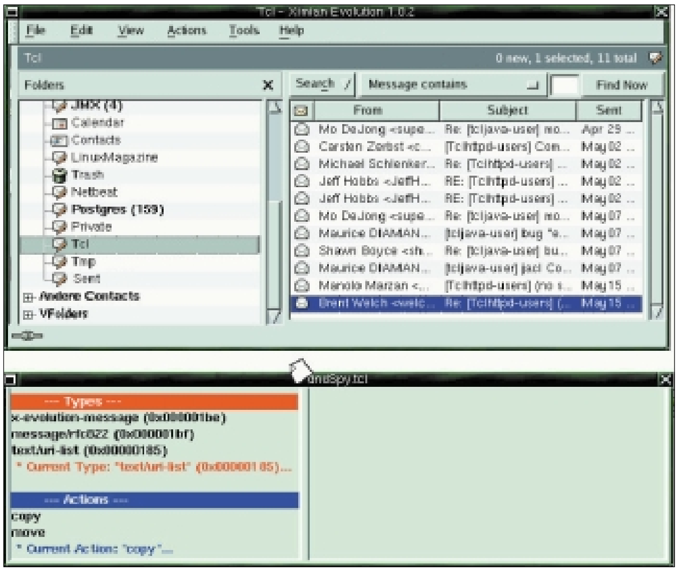


Figure 4: The BWidgets demo application shows the capabilities of this toolkit. The widgets are written in Tcl and only use Tk.



From	Subject	Sent
Mo DeJong <supe...	Re: [tcljava-user] no...	Apr 29 ...
Carsten Zerbst <c...	[Tclhttpd-users] Com...	May 02 ...
Michael Schlenker...	Re: [Tclhttpd-users] ...	May 02 ...
Jeff Hobbs <JeffH...	RE: [Tclhttpd-users] ...	May 02 ...
Jeff Hobbs <JeffH...	RE: [Tclhttpd-users] ...	May 02 ...
Mo DeJong <supe...	Re: [tcljava-user] no...	May 07 ...
Maurice DIAMAN...	[tcljava-user] bug "e...	May 07 ...
Shawn Boyce <sh...	Re: [tcljava-user] bu...	May 07 ...
Maurice DIAMAN...	[tcljava-user] jacl Co...	May 07 ...
Manolo Marzan <...	[Tclhttpd-users] (no s...	May 15 ...
Brent Welch <welc...	Re: [Tclhttpd-users] (...	May 15 ...

```
--- Types ---
x-evolution-message (0x000001be)
message/rfc822 (0x000001bf)
text/uri-list (0x00000185)
 * Current Type: "text/uri-list" (0x00000185)...

--- Actions ---
copy
move
 * Current Action: "copy"...
```

Figure 5: ***dndspy*** **(bottom) shows the MIME types and actions allowed with drag objects. Grabbing a message (flying page) from Evolution (top).**

rect to the HTML page for this project. If this does not occur, a message is then stored in the variable *message* and it is written to the page by *$message*, which is the message construct.

The middle section inside the template contains a simple form, whose input can be returned to the template by the use of *HTTP-POST*. The form contains a text input box and radio buttons, which have been assigned to *project1* and *project2*, and are labeled *Project 1* and *Project 2*. At the end of the script the user input is formatted. This is then output so that it can be used for debugging purposes.

When a user first opens the *listing4.html* page, no project has been selected. In this case clicking on the *Send* button will return the user to the same page. Any entries in the text boxes are retained due to *html::formValue text*. Users must first select a project before the input validity check can redirect them to another page.

The Html and Ncgi packages comprise solutions for numerous tasks that involve formulae and their evaluation. Table 1 contains some practical functions. The complete documentation for the packages is available in Tcllib[3].

Data crumbs

Having to retype data is clumsy, and not all browsers can or want to store user input. Although it is easy to store data on the server, it is by no means trivial to assign it to a specific user. HTTP is stateless and requests are normally independent of any previous requests.

Cookies provide a solution to this known problem: The server asks the browser to store some data (a cookie) on the client. If the user revisits the site, and has not deleted the cookie meanwhile, the server can read the cookie it asked the browser to store.

The template in the next listing first checks the browser for the *answer* cookie. If the cookie exists, it outputs the value, if not, the template attempts to write a cookie. Apart from the server that sets the cookie values the browser itself can also read the cookie, as you can see in Figure 2. Cookies can often be troublesome, especially from a legal point of view (data protection and so on); they also pose a security risk. You will want to try to avoid using them as a building block of your application logic.

```
<html> <head> <title>↩
Cookie</title> </head>
 <body> <h1>Cookie</h1>
  [ Doc_Dynamic
    if {[string length ↩
    [Doc_Cookie answer]] > 0} {
     set html "Cookie answer is ↩
     [Doc_Cookie answer]"
    } else {
     Doc_SetCookie -name answer ↩
     -value 42 -path $page(url)
     set html "Set Cookie ↩
     answer=42"
   }]
  </body>
</html>
```

Login for Web Applications

Although open networks are a good thing, you might need to implement access restrictions for part of your web site. In addition to the *.htaccess* files you will be familiar with from Apache, Tclhttpd supports authentication by a Tcl procedure. A file called *.tclaccess*, which is stored in the individual directories, takes care of this. You need to set the variables *realm* and *callback* here.

```
set realm "tickle"
set callback aok?

proc aok? {sock realm user ↩
password} {
 if {[string match $password ↩
 tickle]} {
   return 1
 } else {
   return 0
 }
}
```

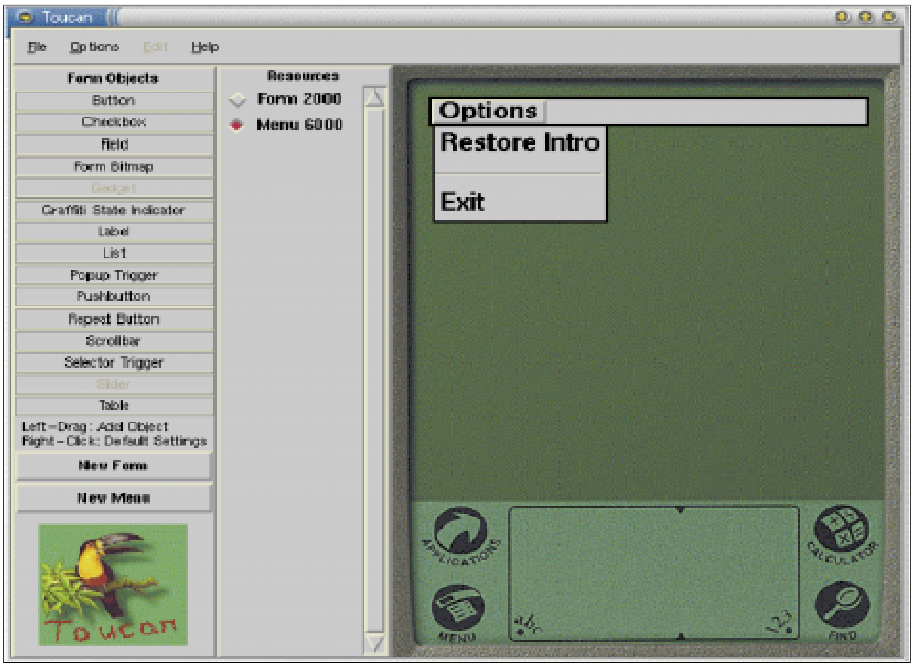


Figure 6: Toucan a GUI developer interface for Palm programs. The IDE and any software created with it are based on Tcl.

The *callback* variable contains the name of the Tcl procedure responsible for authentication. The browser displays the content of *realm* as a text during the login dialog. This data is also passed as an argument to the callback procedure. Our example accepts users that supply the password *tickle*. A real application would access a user password entry in a file, database or LDAP directory. After logging in the user name is stored in the *::env(REMOTE_USER)* variable.

Sessions are a more advanced variant. Here, Tclhttpd uses its own interpreter for the individual sessions. This provides for data separation between the sessions, and so retaining the data for each of the individual sessions.

More Info

The capabilities described in this article cover only a small portion of this web server's total functionality. The sample files bundled with the Tclhttpd package additionally show you how to upload files and use image maps. You can obtain additional information, which is available from [2]. Amongst the other interesting snippets, you will note an excerpt from a book by Brent Welch, Tclhttpd's author. Sourceforge also offers a mailing list that provides competent answers to complex issues, but still finds time to deal with beginners' questions.

In some cases you may not need to go to all the trouble of developing a new application yourself. The Infocetera[12] site provides a complete Groupware application, including a calendar, address book, room planner, task planner, and other modules. The whole application is based on Tclhttpd.

Our next informative TCL article will soon be appearing on your pages of Linux Magazine from out of the murky depths of server applications. We will be looking into both Tk and the BWidget set. The image in Figure 4 should serve to whet your appetite for this widget. It provides capable new widgets by using only the Tk standard widgets and pure Tcl code. ■

Tcl News

Next release anticipated

Tcl 8.4 will be leaving beta in the Autumn. Although I have not noticed any errors for a long time now, the Tcl coreteam prefers to wait and deliver an absolutely perfect version. A whole bunch of new applications in and for Tcl are available right now. The BWidgets and additional GUI elements such as Tree and Combobox – which will be featured in the next issue of TCL – are now available in version 1.4 [3]. The demo in Figure 4 might whet your appetite.

Ora Tcl and Tcl XML

There is also news on these two packages which were covered in our last TCL article: The latest version of the Ora Tcl[9] database extensions supports the full range of Oracle 9i features and the Tcl XML package on Sourceforge[10] now comprises xmlgen, providing a new approach to generating XML and HTML. xmlgen provides a language map between Tcl and XML: Instead of using elements and attributes, you can now work on a higher level with application objects.

Lots of Little Tools

Apart from all these server oriented treats there is news on several smaller tools. Toucan, a developer interface for Palm programs[8] requires only a minimal hardware platform. Both the developer interface and any applications developed with it are based on Tcl (Figure 6). The Googbar (Figure 3), which can launch Google searches[7] has an extremely small memory footprint. Tkdnd, a drag & drop extension for Tk[11] needs even less space on screen. It runs on the XDND protocol used by Gnome and KDE applications and also on Windows. *dndspy* (Figure 5) is also included. This program shows data traffic in the DND protocol

INFO

[1] Tclhttpd home page: *http://sourceforge.➘net/projects/tclhttpd/*

[2] Informationen on Tclhttpd: *http://www.➘tcl.tk/software/tclhttpd/*

[3] Home pages for Tcllib and BWidgets: *http://sourceforge.net/projects/tcllib/*

[4] Medusa project: *http://ciheam.maich.➘gr/medusa/*

[5] Tcl Apache module: *http://tcl.apache.➘org/mod_tcl/mod_tcl.html*

[6] Websh: *http://websh.com*

[7] Googbar: *http://www.geddy.hpg.ig.com.➘br/software/googbar/*

[8] Toucan: *http://home.attbi.com/➘~maccody/*

[9] Ora Tcl: *http://oratcl.sourceforge.net*

[10] Tcl XML: *http://tclxml.sourceforge.net*

[11] Tk DND: *http://www.iit.demokritos.gr/➘~petasis/*

[12] Infocetera: *http://www.infocetera.com*

THE AUTHOR

Carsten Zerbst works for Atlantec on the PDM ship building system. He is also interested in Tcl/Tk usage and applications.

ORBIT/COMDEX EUROPE CONGRESS 2002

messe schweiz

Praxis-Lösungen für IT-Anwender.

		Information Security Vom Produkt zur Strategie – eine gesamtheitliche Betrachtungsweise	**IT for Finance** Das IT-Forum für den Finanzsektor in Deutschland und in der Schweiz	**Enterprise Mobility** Business-Gründe für mobile Verbindungen – überall und jederzeit	**Procurement im E-Business** Wie europäische Unternehmen ihre Einkaufsprozesse optimieren	**Content meets Business** Content und Knowledge Management als Teil des Geschäftsprozesses
		Hauptsponsor: www.ca.com		Hauptsponsor: www.orange.ch Partner: www.gigagroup.net	Sponsor: www.conextrade.com Partner: www.ecademy.ch www.softnet.ch	Partner: www.contentmanager.de www.gigagroup.net www.netzwoche.ch
Mittwoch, 25.9.2002	9.30 bis 10.00	k Keynote: IT Security: das Spektrum der Bedrohung David Love, Head of Security Strategy EMEA, Computer Associates. *Sprache:* Englisch				
	10.30 bis 12.00	s1 Management-Aspekte der IT-Sicherheit	f1 Potenziale für Kostenreduktion in der Banken-IT	e1 Gute Business-Gründe für mobile Unternehmensapplikationen	p1 E-Procurement für KMUs	c1 Die 10 Kernfragen im Content Management
	13.30 bis 15.00	s2 Strategische Informationssicherheit (Sprache: Englisch)	f2 Die Zukunft der Finanzmarktplätze im Internet	e2 Die standardbasierte Plattform «Mobile Office»	p2 Kostensenkung im Ersatzteilmanagement	c2 Content-Management-Strategien für KMUs
	15.30 bis 17.00	s3 IDS-Geschichte, Gegenwart und Zukunft	f3 Customer Relationship Management im Finanzsektor	e3 Sprachtechnologie – das nächste Benutzer-Interface für das Internet	p3 Beschaffungsoptimierung in Grossunternehmen	c3 CMS-Lösungen – Welche Lösung eignet sich für welches Problem?
Donnerstag, 26.9.2002	9.30 bis 10.00	k Keynote: Information Warfare: eine wirtschaftliche Betrachtung David Love, Head of Security Strategy EMEA, Computer Associates. *Sprache:* Englisch				
	10.30 bis 12.00	s4 Security in der Microsoft-Welt	f4 Internet Banking	e4 Wissen Carriers tatsächlich, woher ihr Wachstum kommen wird?	p4 Collaborative Buying	c4 Content im Business – Erfolgsberichte
	13.30 bis 15.00	s5 Macht und Ohnmacht von Grossmächten im Internet	f5 Internet Banking: Perspektiven	e5 Verbindung von verteilten Arbeitsplätzen – Work Wirefree®	p5 Procurement Service Providers für die öffentliche Hand	c5 Von Content über Media Asset Management zum Geschäftsprozess
	15.30 bis 17.00	s6 Mobile IT; klein und fein, darfs auch sicher sein?	f6 Versicherungs- und Bankentechnologie	e6 Wearable Computing – Das tragbare Büro	p6 Prozessoptimierung mit Lieferanten	c6 Was Sie über Webanalyse wissen sollten! Tool-Anbieter berichten

(Änderungen vorbehalten. Stand 20. Juni 2002)

Attraktive Kongress-Packages!

Beim Kauf einer Sessionkarte erhalten Sie die folgenden Leistungen

- Eintritt zur ausgewählten Session
- Pausengetränke
- Tageskarte Orbit/Comdex Europe 2002 (Messe)*
- 1 Buch «Procurement im E-Business» – E-Business Cases (2001)*
- 1 CD-ROM EITO 2002 (solange Vorrat)*
- 1 Kongress-Bag mit Dokumentation*

* Diese Leistungen sind nur bei Bestellungen von Kongresskarten à CHF 180.–/CHF 200.– inbegriffen.

Alle Seminare finden im Kongresszentrum Basel statt.

Vorverkauf (bis 23.9.2002)
Preis für eine Session: CHF 180.–
Peis für jede weitere Session (für die gleiche Person, verschiedene Session/s): CHF 130.–

Ticketverkauf vor Ort (24.–26.9.2002)
Preis für eine Session: CHF 200.–
Peis für jede weitere Session (für die gleiche Person, verschiedene Session/s): CHF 150.–

Basel, 24.–27. September 2002

Die Orbit/Comdex Europe 2002 bietet IT-Anwendern eine breite, praxisorientierte Informationsplattform an: Neben den fünf Kongressthemen Information Security, IT for Finance, Enterprise Mobility, Procurement im E-Business und Content meets Business präsentiert die Orbit/Comdex Europe unter anderem die folgenden Messehighlights: Information Security Park, Content Expo, Innovation leads Business und den Enterprise IT Buyer's Club. Zahlreiche Aussteller stellen die neusten Produkte und Dienstleistungen aus den Bereichen IT, Telekommunikation, Internet und E-Commerce vor.

Kongressanmeldungen und weitere Informationen unter www.orbitcomdex.com oder Tel. +41 58 200 20 20.

orbit® EUROPE 2002 COMDEX®

INFORMATION TECHNOLOGY – ONE STEP AHEAD