```
/* ---------------------------------------------------------------------- */
/* --                Main module (Calendar Service)                      -- */
/* --                This script handles all CGI requests                -- */
/* --                                                                    -- */
/* --                Thomas Jungmann thj@gmx.net                         -- */
/* --                last modified 03.12.2000                            -- */
/* --                                                                    -- */
/* ---------------------------------------------------------------------- */

call dbgout "Calendar CGI started "                     --for debug purposes only
                                                        --writes debug info to a file

call initVars                                           --inits all handles to the interface classes (DB, etc.)
                                                        --and puts the CGI-Variables into cgi.

html~printHeader('The Web Calendar')                    --print MIME-Type and HTML Header

select
    when cgi.action = 'logon'       then call logon
    when cgi.action = 'newuser'     then call newuser
    when cgi.action = 'logout'      then call logout
    when cgi.action = 'overview'    then call overview
    when cgi.action = 'goto'        then call gotoMonth
    when cgi.action = 'gotoform'    then call gotoForm
    when cgi.action = 'viewday'     then call viewDay
    when cgi.action = 'view'        then call viewEvent
    when cgi.action = 'create'      then call createEvent
    when cgi.action = 'newentry'    then call newentryForm
    when cgi.action = 'Edit'        then call editEventForm
    when cgi.action = 'update'      then call updateEvent
    when cgi.action = 'Delete'      then call deleteEvent
    otherwise call abort 'Unknown CGI-Action'
end

html~printTail                                          --complete the HTML page

DROP db                                                 --drop references to the interface classes
DROP html
```

```
39 DROP parser
40 call dbgout "CGI-Script successfully completed"
41 exit 0
42
43 -----------END OF MAIN PROGRAM-------------
44
45 ----------------------------------------------------------------
46 /*                          Subroutines                         */
47 ----------------------------------------------------------------
48
49 ---------------------------
50 /* Handles the logon request */
51 ---------------------------
52 logon:  PROCEDURE EXPOSE html db cgi.                  --cgi. keeps all CGI-Variables, e.g.
53                                                        --?name=peter -> is stored in cgi.name
54     call dbgout "logon()"
55     uid = db~getUserID(cgi.name)                       --find User ID for given name
56     select
57         when uid = -1 then call abort 'DB-Access error while retrieving UserID'
58         when uid = -2 then call abort 'Username not found'
59         otherwise nop
60     end
61     dbpwd = db~getUserPwd(uid)                         --get Users Password
62     select
63         when dbpwd = -1 then call abort 'DB-Access error while retrieving UserPassword'
64         when dbpwd = -2 then call abort 'UserID not found'
65         otherwise nop
66     end
67     if \(dbpwd == cgi.pwd) then                        --Pwd. does not match to Pwd. stored in Database
68         do
69             html~out(html~paragraph('Passwords must be entered exactly, as they are case sensitive!'))
70             call abort 'Password incorrect'
71         end
72     newSID = db~createSID(uid)                         --create a new session identifier
73     if newSID = -1 then                                --problem with the database?
74         call abort 'Could not create SID'
75     cgi.sid = newSID                                   --store SID in global variable,
76                                                        --for later user (for procedure overview)
```

```
 77     db~deleteAllExpiredSID()                              --delete old and expired SIDs - for cleanup only
 78
 79     call overview                                         --overview shows the month-overview (=start page)
 80     return
 81
 82 ----------------------------
 83 /* Handles the logoff request */
 84 ----------------------------
 85 logout:     PROCEDURE EXPOSE html db cgi.
 86     call dbgout "logout()"
 87     res = db~closeSession(cgi.sid)                        --delete entry for current session in the database
 88     if res \=0 then call abort 'SID could not be deleted'   --perhaps expired and then deleted in a cleanup process
 89     html~out(html~headline('You have successfully logged out, thank you for using THE WEB CALENDAR',3))
 90     return
 91
 92 ----------------------------
 93 /* Handles the input from 'newuser.html' */
 94 ----------------------------
 95 newuser:    PROCEDURE EXPOSE html db cgi.
 96     call dbgout "newUser()"
 97     if cgi.name == "CGI.NAME" then call abort 'Please provide a valid username'     --cgi.name undefined, no name given at all
 98     if symbol(cgi.name) == 'BAD' then call abort 'Please do not use any special characters in your username'
 99     if cgi.pwd  == "CGI.PWD" then call abort 'Please provide a valid password'      --cgi.pwd undef., nothing was entered
100     if symbol(cgi.pwd)  == 'BAD' then call abort 'Please do not use any special characters in your password'
101     res = db~createNewUser(cgi.name,cgi.pwd)              --insert values into databases 'user'-relation
102
103     if res < 0 then                                       --error? probably the username is already in the database,
104         do                                                --but could also indicate some other DB-Error...
105             html~out(html~paragraph('Username already exists. Please use another name.'))
106             html~out(html~hyperlink('../newuser.html','Try again...'))
107             call abort 'Username exists'
108         end
109     html~out(html~paragraph('You have successfully created a new user account, you may now log on to THE WEB CALENDAR'))
110     return
111
112 ----------------------------
113 /* This procedure displays the current month as overview */
114 ----------------------------
```

```
115 overview:   PROCEDURE EXPOSE html db cgi.
116     call dbgout "overview()"
117     uid = db~getUID4SID(cgi.sid)                        --Find userID for given session ID
118     if uid < 0 then call abort 'Session has expired, please log on again'
119     html~printNav(cgi.sid)                              --Print HTML-Head with links for the main actions
120
121     call viewMonth uid,date('S',,,'-')                  --show overview for given user and current date
122     return
123
124 ----------------------------
125 /* This procedure displays any month between 01.01.0001 and 31.12.9999
126    according to the gegorian calendar */
127 ----------------------------
128 viewMonth:  PROCEDURE EXPOSE html db cgi.
129     call dbgout "viewMonth()"
130     parse arg uid,dt
131     calendar. = getCalendar(uid,dt)                     --this array consists of 4 to 5 textlines
132                                                         --already formated as days of a specifiv month
133
134     dtLang = date('L',dt,'S',,'-')                      --convert date to english language format
135
136     parse var dtLang . month year                       --get month & year for the calendar headline
137     prevDt = getPrevMonth(dt)                           --get previous and next month for the hyperlinks
138     nextDt = getNextMonth(dt)                           --next to the headline...
139     parse var prevDt pYear '-' pMonth '-' .             --... and split the date
140     parse var nextDt nYear '-' nMonth '-' .
141
142                                                         --prepare the hyperlinks for previous and next month
143     pm = html~hyperlink('calserv.cgi?action=goto&year='||pYear||'&month='||pMonth||'&sid='||cgi.sid,'&lt')
144     nm = html~hyperlink('calserv.cgi?action=goto&year='||nYear||'&month='||nMonth||'&sid='||cgi.sid,'&gt;')
145     html~out('<table border="0" cellpadding="15"><tr>')
146     html~out('<tr><td bgcolor= "#FFFF99" valign="TOP">')
147     html~out(html~headline(pm||' 'month||' '||year||' '||nm,1,'CENTER'))    --headline output
148     html~out('</td></tr>')
149     html~out('<td bgcolor="#FFFFCC" valign="TOP" align="LEFT">')
150     html~out(html~fixedSpaceStart)                      --ensure that a fixed space font is used, so
151                                                         --things don't get messed up
152     html~out('<font size="4">')
```

```
153     html~out('Mon Tue Wed Thu Fri Sat Sun')
154     html~out('---------------------------')
155     do i=1 to calendar.0                                --output of the formerly prepared calendar array
156         html~out(calendar.i)
157         if i <> calendar.0 then html~out('<br>')
158     end
159     html~out('</font>')
160     html~out(html~fixedSpaceEnd)
161     html~out('</td></tr></table>')
162     call gotoForm                                       --display a small form for quick jump to other month/year
163     return
164
165 ----------------------------
166 /* Show the overview of all events on one day */
167 ----------------------------
168 viewDay:    PROCEDURE EXPOSE html db cgi.
169     call dbgout "viewDay()"
170     uid = db~getUID4SID(cgi.sid)                        --Find userID for given session ID
171     if uid < 0 then call abort 'Session has expired, please log on again'
172     html~printNav(cgi.sid)                              --Print HTML-Head with links for the main actions
173     events. = db~getEventsOfDay(uid,cgi.date)           --events. -array has now all events for the day
174     if events.eventID.0 = 0 then html~out(html~paragraph(date('L',cgi.date,'S',,'-')||': There are no events on schedule for
    this day: '))
175     else
176         do
177             html~out('<table border="0" cellpadding="5">')
178                                                                 --headline with date
179             html~out('<tr><td valign="TOP" align="CENTER"
    colspan="2">'||html~headline(date('L',cgi.date,'S',,'-'),2,'CENTER')||'</td>')
180
181             do i=1 to events.eventID.0                          --loop over all events
182                 html~out('<tr>')
183                 hl = 'calserv.cgi?action=view&eventid='||events.eventID.i||'&sid='||cgi.sid  --prepare hyperlink for event
184                 html~out(html~tabdata(events.eventTime.i,i//2))          --i mod 2 = alternating bgcolor
185                 html~out(html~tabdata(html~hyperlink(hl,events.description.i),i//2))
186                 html~out('</tr>')
187             end
188             html~out('</table>')
```

```
189         end
190     return
191
192 ----------------------------
193 /* Show an event with all details */
194 ----------------------------
195 viewEvent:      PROCEDURE EXPOSE html db cgi.
196     call dbgout "viewEvent()"
197     sq = d2c(39)
198     checkUID = db~getUID4SID(cgi.sid)                   --Find userID for given session ID
199     if checkUID < 0 then call abort 'Session has expired, please log on again'
200     html~printNav(cgi.sid)                              --Print HTML-Head with links for the main actions
201     event. = db~retrieveEvent(cgi.eventid)              --get event from the database and store it in event. array
202     if event.eventID.0 = 0 then call abort 'Event has not been found'
203     if event.uID.1 \= checkUID then call abort 'Event does not belong to user'      --if someone has tampered the
204                                                                                     --query string to get information from other users
205
206     html~out('<table border="0" cellpadding="5">')
207     html~out('<tr>')
208     html~out(html~tabdata('Date',0))
209     html~out(html~tabdata(event.eventDate.1,0))
210     html~out('</tr>')
211     html~out('<tr>')
212     html~out(html~tabdata('Time',1))
213     html~out(html~tabdata(event.eventTime.1,1))
214     html~out('</tr>')
215     html~out('<tr>')
216     html~out(html~tabdata('Description',0))
217     html~out(html~tabdata(event.description.1,0))
218     html~out('</tr>')
219     html~out('<tr>')
220     html~out(html~tabdata('Category',1))
221     html~out(html~tabdata(db~getCategory(event.categoryID.1),1))
222     html~out('</tr>')
223     html~out('<tr>')
224     html~out(html~tabdata('Duration',0))
225     html~out(html~tabdata(event.duration.1,0))
226     html~out('</tr>')
```

```
227     html~out('</table>')
228
229                                                 --this form displays the 'Edit' and 'Delete' buttons
230     html~out('<FORM ACTION='||sq||'calserv.cgi'||sq||' METHOD="GET">')
231     html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'eventid'||sq||' VALUE='||cgi.eventid||'>')
232     html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'sid'||sq||' VALUE='||cgi.sid||'>')
233     html~out('<INPUT TYPE='||sq||'SUBMIT'||sq||' NAME='||sq||'action'||sq||' VALUE='||sq||'Edit'||sq||'>')
234     html~out('<INPUT TYPE='||sq||'SUBMIT'||sq||' NAME='||sq||'action'||sq||' VALUE='||sq||'Delete'||sq||'>')
235     html~out('</FORM>')
236     return
237
238 ----------------------------
239 /* This procedures creates a HTML-Form for new entries or for
240    existing entries that are to be modified */
241 ----------------------------
242 newEntryForm:    PROCEDURE EXPOSE html db cgi.
243                                                 --in case of a modification of an existing
244                                                 --event, these default arguments get their values
245                                                 --from the procedure 'editEventForm'
246     use arg defEventID,defDay,defMonth,defYear,defHour,defMinute,defDHour,defDMinute,defDescription,defCategory
247     call dbgout "newEntry()"
248     sq = d2c(39)                                -- decimal to char, 39 = single quote
249
250     uid = db~getUID4SID(cgi.sid)                --Find userID for given session ID
251     if uid < 0 then call abort 'Session has expired, please log on again'
252     html~printNav(cgi.sid)                      --Print HTML-Head with links for the main actions
253
254
255     if defDay=='DEFDAY' then                    --not default values given, creating some
256         do
257             html~out(html~headline('Create a new entry',3))
258             dt = date('S')                                  --current date and time as the default
259             parse var dt defYear 5 defMonth 7 defDay
260             tm = time()
261             parse var tm defHour ':' defMinute ':' .
262             defDHour = 0
263             defDMinute = 0
264             defDescription = ''
```

```
265             defCategory = 1
266             action = 'create'                       --prepare 'hidden field' 'action'
267         end
268     else
269         do
270             html~out(html~headline('Edit event',3))
271             action = 'update'                       --prepare 'hidden field' 'action'
272         end
273
274     html~out('<FORM ACTION='||sq||'calserv.cgi'||sq||' METHOD="GET">')      --start form
275     html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'action'||sq||' VALUE='||sq||action||sq||'>')
276     if action='update' then                         --for update an eventID must be supplied
277         html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'eventid'||sq||' VALUE='||sq||defEventID||sq||'>')
278     html~out('Date: <SELECT NAME='||sq||'day'||sq||' SIZE=1>')  --single line select box (dropdown) for 'day'-value
279     do i=1 to 31
280         padi = right(i,2,'0')                               --ensures that numbers < 10 are expanded (padded)
281                                                             --with a preceding '0', so they are alway 2 chars long
282         if i=defDay then
283             html~out('<OPTION SELECTED VALUE='||sq||padi||sq||'>'||padi)        --this is the default
284         else
285             html~out('<OPTION VALUE='||sq||padi||sq||'>'||padi)             --these are all others
286     end
287     html~out('</SELECT> ')
288
289     html~out('<SELECT NAME='||sq||'month'||sq||' SIZE=1>')      --dropdown-field for month
290     do i=1 to 12
291         padi = right(i,2,'0')                               --i padded with preceding '0'
292
293         dt = date('L','00/'||padi||'/01','O')               --english names of the months
294         parse var dt . month .                              --is stored in 'month'
295         if i=defMonth then
296             html~out('<OPTION SELECTED VALUE='||sq||padi||sq||'>'||month)           --default
297         else
298             html~out('<OPTION VALUE='||sq||padi||sq||'>'||month)                    --other
299     end
300     html~out('</SELECT>')
301
302                                                             --textfield for the year
```

```
303     html~out(' <INPUT TYPE='||sq||'TEXT'||sq||' NAME='||sq||'year'||sq||' SIZE=4 VALUE='||sq||defYear||sq||'>')
304
305     html~out('<P>Time: <SELECT NAME='||sq||'hour'||sq||' SIZE=1>')  --dropdown for time/hour
306     do i=0 to 24
307         padi = right(i,2,'0')                                   --i padded with preceding '0'
308         if i=defHour then
309             html~out('<OPTION SELECTED VALUE='||sq||padi||sq||'>'||padi)
310         else
311             html~out('<OPTION VALUE='||sq||padi||sq||'>'||padi)
312     end
313     html~out('</SELECT> ')
314
315     html~out('<SELECT NAME='||sq||'minute'||sq||' SIZE=1>')     --dropdown 'minute'
316     do i=0 to 60
317         padi = right(i,2,'0')
318         if i=defMinute then
319             html~out('<OPTION SELECTED VALUE='||sq||padi||sq||'>'||padi)
320         else
321             html~out('<OPTION VALUE='||sq||padi||sq||'>'||padi)
322     end
323     html~out('</SELECT>')
324
325     html~out(' Duration: <SELECT NAME='||sq||'dhour'||sq||' SIZE=1>')  --dropdown 'duration'/hour
326     do i=0 to 24
327         padi = right(i,2,'0')
328         if i=defDHour then
329             html~out('<OPTION SELECTED VALUE='||sq||padi||sq||'>'||padi)
330         else
331             html~out('<OPTION VALUE='||sq||padi||sq||'>'||padi)
332     end
333     html~out('</SELECT> ')
334
335     html~out('<SELECT NAME='||sq||'dminute'||sq||' SIZE=1>')                    --dropdown 'duration'/minutes
336     do i=0 to 60
337         padi = right(i,2,'0')
338         if i=defDMinute then
339             html~out('<OPTION SELECTED VALUE='||sq||padi||sq||'>'||padi)
340         else
```

```
341             html~out('<OPTION VALUE='||sq||padi||sq||'>'||padi)
342     end
343     html~out('</SELECT>')
344
345     html~out('<P>Description:')                                     --textfield for event description
346     html~out('<TEXTAREA ROWS=4 COLS=25 NAME='||sq||'description'||sq||'>'||defDescription||'</TEXTAREA>')
347
348     maxCat = db~getMaxCategory()                                --number of existing categorys in database
349     html~out('<P>Category: <SELECT NAME='||sq||'category'||sq||' SIZE=1>')  --dropdown for event category
350     do i=1 to maxCat
351         if i=defCategory then
352             html~out('<OPTION SELECTED VALUE='||sq||i||sq||'>'||db~getCategory(i))
353         else
354             html~out('<OPTION VALUE='||sq||i||sq||'>'||db~getCategory(i))
355     end
356     html~out('</SELECT>')
357                                                                     --Save/Reset buttons
358     html~out('<P><INPUT TYPE='||sq||'SUBMIT'||sq||' VALUE='||sq||'Save'||sq||'>')
359     html~out('<INPUT TYPE='||sq||'RESET'||sq||' VALUE='||sq||'Reset'||sq||'>')
360     html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'sid'||sq||' VALUE='||cgi.sid||'>')
361     html~out('</FORM>')
362
363     html~out('<FORM ACTION='||sq||'calserv.cgi'||sq||' METHOD="GET">')      --Cancel button (back to overview)
364     html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'action'||sq||' VALUE='||sq||'overview'||sq||'>')
365     html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'sid'||sq||' VALUE='||cgi.sid||'>')
366     html~out('<INPUT TYPE='||sq||'SUBMIT'||sq||' VALUE='||sq||'Cancel'||sq||'>')
367     html~out('</FORM>')
368     return
369
370 ----------------------------
371 /* This procedure actually creates an event that has been
372    entered in the 'createEventForm' */
373 ----------------------------
374 createEvent:    PROCEDURE EXPOSE html db cgi.
375     call dbgout "createEvent()"
376     sq = d2c(39)                            -- decimal to char, 39 = single quote
377     uid = db~getUID4SID(cgi.sid)
378     if uid < 0 then call abort 'Session has expired, please log on again'
```

```
379     html~printNav(cgi.sid)
380
381     eventDate = cgi.year||'-'||cgi.month||'-'||cgi.day          --put date together...
382     eventTime = cgi.hour||':'||cgi.minute||':00'                --... and time. seconds are alway 00
383     eventDuration = cgi.dhour||':'||cgi.dminute||':00'
384     res = db~createEvent(uid,cgi.category,cgi.description,eventDate,eventTime,eventDuration)  --store data in DB
385     if res \= 0 then call abort "Error while creating event"
386
387     html~out(html~headline('Event successfully stored',3))
388     html~continueButton(cgi.sid)                                --display 'continue'-button (to main overview)
389     return
390
391 ----------------------------
392 /* This is executed when a user clicks the 'delete'-button in the event view */
393 ----------------------------
394 deleteEvent:    PROCEDURE EXPOSE html db cgi.
395     call dbgout "deleteEvent()"
396     checkUID = db~getUID4SID(cgi.sid)
397     event. = db~retrieveEvent(cgi.eventid)              --event is retrieved from the database because
398                                                         --we have to check if the eventID actually belongs
399                                                         --to the user (we need only the user ID)
400     if checkUID < 0 then call abort 'Session has expired, please log on again'
401     if event.eventID.0 = 0 then call abort 'Event has not been found'
402     if event.uID.1 \= checkUID then call abort 'Event does not belong to user'
403
404     html~printNav(cgi.sid)
405     res = db~deleteEvent(cgi.eventid)
406     if res \= 0 then call abort "Error while creating event"
407     html~out(html~headline('Event deleted'))
408     html~continueButton(cgi.sid)                                --display 'continue'-button (to main overview)
409     return
410
411 ----------------------------
412 /* Create a form to modify an existing event.
413    This just creates the default values for the 'newEventForm' procedure */
414 ----------------------------
415 editEventForm:      PROCEDURE EXPOSE html db cgi.
416     call dbgout "editEvent()"
```

```
417     uid = db~getUID4SID(cgi.sid)
418     call dbgout "uid: "||uid
419     if uid < 0 then call abort 'Session has expired, please log on again'
420
421     event. = db~retrieveEvent(cgi.eventid)                      --get the old event data for the default values
422     dt = event.eventDate.1
423     tm = event.eventTime.1
424     dr = event.duration.1
425     parse var dt year '-' month '-' day                         --split the event date into single variables
426     parse var tm hour ':' minute ':' .                          --...and the event time
427     parse var dr dhour ':' dminute ':' .                        --...and the event duration as well
428
429     call newEntryForm event.eventID.1,day,month,year,hour,minute,dhour,dminute,event.description.1,event.categoryID.1
430     return
431
432 ----------------------------
433 /* This procedure actually does the update-job
434    It gets its data from the 'newEntryForm'-procedure */
435 ----------------------------
436 updateEvent: PROCEDURE EXPOSE html db cgi.
437     call dbgout "updateEvent()"
438     sq = d2c(39)
439     checkUID = db~getUID4SID(cgi.sid)
440     event. = db~retrieveEvent(cgi.eventid)              --old data needed for check is a user
441                                                         --is allowed to change the data (i.e. belongs to him)
442     if checkUID < 0 then call abort 'Session has expired, please log on again'
443     if event.eventID.0 = 0 then call abort 'Event has not been found'
444     if event.uID.1 \= checkUID then call abort 'Event does not belong to user'
445     html~printNav(cgi.sid)
446
447     eventDate = cgi.year||'-'||cgi.month||'-'||cgi.day          --put date together
448     eventTime = cgi.hour||':'||cgi.minute||':00'                --...and time; seconds alway 00
449     eventDuration = cgi.dhour||':'||cgi.dminute||':00'
450     res = db~updateEvent(cgi.eventid,cgi.category,cgi.description,eventDate,eventTime,eventDuration)
451     if res \= 0 then call abort "Error while updating event"
452
453     html~out(html~headline('Event updated'))
454     html~continueButton(cgi.sid)                                --display 'continue'-button
```

```
455     return
456
457 ----------------------------
458 /* Shows the 'goto'-month-form (under the main calendar view) */
459 ----------------------------
460 gotoForm:   PROCEDURE EXPOSE html cgi.
461     call dbgout "gotoForm()"
462     sq = d2c(39)                            -- decimal to char, 39 = single quote
463     html~out(html~headline('Choose month and year to display',4))
464     html~out('<FORM ACTION='||sq||'calserv.cgi'||sq||' METHOD="GET">')
465     html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'action'||sq||' VALUE='||sq||'goto'||sq||'>')
466     html~out('<SELECT NAME='||sq||'month'||sq||' SIZE=1>')
467     dt = date('L')                                          --current date in english language
468     parse var dt . thisMonth .                              --e.g. thisMonth='January'
469     do i=1 to 12
470         padi = right(i,2,'0')                               --expand i<10 to 2 digits with preceding '0'
471         dt = date('L','00/'||padi||'/01','O')
472         parse var dt . month .                              --all monthnames in english language
473         if month=thisMonth then
474             html~out('<OPTION SELECTED VALUE='||sq||padi||sq||'>'||month)       --current month is default
475         else
476             html~out('<OPTION VALUE='||sq||padi||sq||'>'||month)
477     end
478     html~out('</SELECT>')
479
480     html~out(' <INPUT TYPE='||sq||'TEXT'||sq||' NAME='||sq||'year'||sq||' SIZE=4 VALUE="2000">')   --textfield 'year'
481
482     html~out(' <INPUT TYPE='||sq||'SUBMIT'||sq||' VALUE='||sq||'Go'||sq||'>')          --'Go'-button
483     html~out('<INPUT TYPE='||sq||'HIDDEN'||sq||' NAME='||sq||'sid'||sq||' VALUE='||cgi.sid||'>')
484     html~out('</FORM>')
485     return
486
487 ----------------------------
488 /* Navigate to a date given in the 'gotoForm' */
489 ----------------------------
490 gotoMonth:      PROCEDURE EXPOSE html db cgi.
491     call dbgout "gotoMonth()"
492     uid = db~getUID4SID(cgi.sid)
```

```
493     if uid < 0 then call abort 'Session has expired, please log on again'
494     html~printNav(cgi.sid)
495     if cgi.year <= 0 | cgi.year > 9999 then                     --range check for year
496         do
497             if length(cgi.year) <> 2 then                       --year=00 probably a century year meant
498             do
499                 html~out(html~headline(d2c(39)||cgi.year||d2c(39)||' is not a valid year!',2))
500                 return
501             end
502         end
503 --  cgi.year = trunc(cgi.year)              -- just in case someone enters year 2001.17
504
505     if length(cgi.year) = 2 then            -- nifty heuristic to guess which century the user
506         do                                  -- meant, when he entered only the last 2 digits of the year
507             thisYear = date()
508             parse var thisYear . . thisYear
509             diff = thisYear//100 - cgi.year                 --find the years within 50 years in the past
510             select                                          --and 49 years in the future
511                 when diff>0 & abs(diff)>50 then cgi.year=thisYear+100-diff
512                 when diff>=0 & abs(diff)<=50 then cgi.year=thisYear-diff
513                 when diff<=0 & abs(diff)>=50 then cgi.year=thisYear-(100-abs(diff))
514                 when diff<0 & abs(diff)<50 then cgi.year=thisYear+abs(diff)
515                 otherwise nop
516             end
517         end
518     dt = right(cgi.year,4,'0')||'-'||cgi.month||'-01'
519
520     call viewMonth uid,dt                       --show the month-overview for that date
521     return
522
523 ----------------------------
524 /* This returns an array with 4 to 6 textlines, already formated
525    like a calendar table with hyperlinks on days where the user
526    has at least one event on schedule */
527 ----------------------------
528 getCalendar:     PROCEDURE EXPOSE db html cgi.
529     call dbgout "getCalendar()"
530     parse arg uid,year '-' month '-' day
```

```
531     year  = right(year,4,'0')              --expanded to 4 digits (precedings '0')
532     month = right(month,2,'0')              --expanded to 2 digits (precedings '0')
533     day   = right(month,2,'0')
534     dt = year||'-'||month||'-01'         --put a date-string together, 'day' is the first day of a month
535
536     n = dayOfWeek(dt)                    --determine the weekday of the first day of a month (needed for heading spaces)
537
538     today = date('S',,,'-')              --current date (will be marked as 'bold' in the calendar)
539
540     out.1 = ' '||copies('    ',n)       --begin with the heading spaces (no spaces when monday)
541
542     do i=1 to (7-n)                                  --loop for the first calendar line
543         padi = right(i,2,'0')                        --expand to 2 digits, if necessary
544         testdate = year||'-'||month||'-'||padi       --put a date string together for test if it is today
545         boldS = ''                                   --clear the bold-start variable (empty=no bold tag on this day)
546         boldE = ''                                   --clear the bold-end variable
547         if testdate == today then
548             do
549                 boldS = '<b>'                        --if today, prepare for bold printing
550                 boldE = '</b>'
551             end
552         events. = db~getEventsOfDay(uid,testdate)       --any events on testdate?
553         if events.eventID.0 = 0 then
554             out.1 = out.1||boldS||padi||boldE||'  '     --print without hyperlink (put probably bold, if today)
555         else do
556                                                         --event for testdate exists. prepare the hyperlink
557                 hyperlink = html~hyperlink('calserv.cgi?action=viewday&date='||testdate||'&sid='||cgi.sid,boldS||padi||boldE)
558                 out.1 = out.1||hyperlink||'  '
559         end
560     end
561
562     line = 2                                         --initial value for other calendar lines
563     s = 7-n+1                                        --first day on second line
564     maxDays = daysInMonth(dt)                        --get number of days in a given month
565     do until e = maxDays                             --e = end of calendar line (until expression is checked at end of loop)
566         out.line = ' '                               --one heading space on each line
567         e = s+6                                      --max. 7 days on each line
568         if e > maxDays then e = maxDays              --...but limited to the maximum of days on this month
```

```
569
570         do i=s to e                                     --loop over one week
571             padi = right(i,2,'0')
572             testdate = year||'-'||month||'-'||padi      --the same comment as above applies to these lines....
573             boldS = ''
574             boldE = ''
575             if testdate == today then
576                 do
577                     boldS = '<b>'
578                     boldE = '</b>'
579                 end
580             events. = db~getEventsOfDay(uid,testdate)
581             if events.eventID.0 = 0 then
582                 out.line=out.line||boldS||padi||boldE||'  '
583             else do
584                     hyperlink =
    html~hyperlink('calserv.cgi?action=viewday&date='||testdate||'&sid='||cgi.sid,boldS||padi||boldE)
585                     out.line = out.line||hyperlink||'  '
586             end
587
588         end
589         line = line+1                                   --next line
590         s = e+1                                         --last day+1 is the start day for the next line
591     end
592     out.0 = line-1                          --Linecount of calendar page. Usually 5 lines, but sometimes
593                                             --in 28-Days-Februarys that start with a Monday only 4 (e.g. Feb.1999)
594     return out.
595
596 ----------------------------
597 /* This is called at the very beginning of this script
598    and initializes the handles for the interface objects
599    and reads the CGI-Variables from the query string */
600 ----------------------------
601 initVars:   PROCEDURE EXPOSE db html cgi.
602     call dbgout "initVars()"
603     db     = .sqlInterface~NEW
604     html   = .HTMLgen~NEW
605     parser = .cgiParse~NEW
```

```
606     cgi.   = parser~parseQueryString()
607     return
608
609 ----------------------------
610 /* Called when an error condition occurs and prints
611    the reason as the HTML-Output */
612 ----------------------------
613 abort:  PROCEDURE EXPOSE html
614     parse arg why
615     call dbgout "Abnormal program termination: "||why
616     html~out(html~headline(why,1))
617     html~printTail
618     exit 0
619
620 ----------------------------
621 /* This procedure is only for debugging purposes and
622    writes a file called 'debug.dat' to the disk */
623 ----------------------------
624 dbgout: PROCEDURE
625     use arg out
626     file = .stream~new('debug.dat')
627     file~open('both append')
628     file~lineout(date()||' '||time()||' '||out)
629     file~close()
630     return
631
632
633 ----------------------------
634 /* Returns the number of days a given month has */
635 ----------------------------
636 daysInMonth:    PROCEDURE
637     call dbgout "daysInMonth()"
638     parse arg year '-' month '-' day
639     year  = right(year,4,'0')
640     month = right(month,2,'0')
641     day   = '01'
642     dt = year||month||day
643     dt = date('B',dt,'S')               --date('B') returns the number of days since 01.01.0001
```

```
644     do until (month \= xmonth)          --this loop increases this value until it is switched to the next month
645         res = date('S',dt,'B','-')
646         parse var res . '-' xmonth '-'
647         dt = dt + 1
648     end
649     dt = dt-2
650     dt = date('S',dt,'B','-')           --retransformation to a 'normal' date format
651     parse var dt . '-' . '-' dt         --only the day is important
652     return dt
653
654 ----------------------------
655 /* Returns the weekday of a given date */
656 ----------------------------
657 dayOfWeek:      PROCEDURE
658     call dbgout "dayOfWeek()"
659     use arg dt
660     parse var dt year '-' month '-' day
661     year  = right(year,4,'0')
662     month = right(month,2,'0')
663     day   = right(day,2,'0')
664     dt = year||month||day
665     dt = date('B',dt,'S')//7            --date('B') returns the number of days since 01.01.0001, this
666                                         --modulo 7 is the weekday
667     return dt
668
669 ----------------------------
670 /* Finds the previous month (for the navigation hyperlinks) */
671 ----------------------------
672 getPrevMonth:      PROCEDURE
673     call dbgout "getPrevMonth()"
674     use arg dt
675     parse var dt year '-' month '-' .
676     year  = right(year,4,'0')
677     month = right(month,2,'0')
678     if month = '01' then                            --when month is January...
679         do
680             year = right(year-1,4,'0')
681             month = '12'                            --...previous month is December, the year before
```

```
682         end
683     else
684         month = right(month-1,2,'0')
685     return (year||'-'||month||'-'||'01')
686
687 ----------------------------
688 /* Finds the next month (for the navigation hyperlinks) */
689 ----------------------------
690 getNextMonth:      PROCEDURE
691     call dbgout "getNextMonth()"
692     use arg dt
693     parse var dt year '-' month '-' .
694     year  = right(year,4,'0')
695     month = right(month,2,'0')
696     if month = '12' then                            --when month is December...
697         do
698             year = right(year+1,4,'0')
699             month = '01'                            --next month is January, in the next year
700         end
701     else
702         month = right(month+1,2,'0')
703     return (year||'-'||month||'-'||'01')
704
705 ------------------------------------------------------------------
706 /*                          Directives                          */
707 ------------------------------------------------------------------
708 ::REQUIRES 'cgilib.cmd'
709 ::REQUIRES 'sqli.cmd'
710 ::REQUIRES 'htmlgen.cmd'
```