```
 1:     ///----------------------
 2:     /// prog mdlSjson
 3:     /// zig 0.13.0 dev
 4:     ///----------------------
 5:
 6:
 7:
 8: const std = @import("std");
 9:
10:
11: // keyboard
12: const kbd = @import("cursed").kbd;
13:
14: // panel
15: const pnl = @import("forms").pnl;
16: // button
17: const btn = @import("forms").btn;
18: // label
19: const lbl = @import("forms").lbl;
20: // flied
21: const fld = @import("forms").fld;
22: // line horizontal
23: const lnh = @import("forms").lnh;
24: // line vertival
25: const lnv = @import("forms").lnv;
26:
27: // grid
28: const grd = @import("grid").grd;
29: // menu
30: const mnu = @import("menu").mnu;
31:
32: // full delete for produc
33: const forms = @import("forms");
34:
35:
36: const allocator = std.heap.page_allocator;
37:
38:
39:
40: pub fn  SavJson(XPANEL: *std.ArrayList(pnl.PANEL),
41:                 XGRID: *std.ArrayList(grd.GRID),
42:                 XMENU: *std.ArrayList(mnu.DEFMENU),
43:                 nameJson: []const u8) !void {
44:
45:
46:
```

```
47:
48:     const cDIR = std.fs.cwd().openDir("dspf",.{})
49:     catch |err| {@panic(try std.fmt.allocPrint(allocator,"err Open DIR.{any}\n", .{err}));};
50:
51:     var fjson = cDIR.openFile(nameJson, .{.mode = .read_write}) catch |err| {
52:                 @panic(try std.fmt.allocPrint(allocator,"err Open FILE.{any}\n", .{err}));};
53:     defer fjson.close();
54:
55:     const out  = fjson.writer();
56:
57:     var w = std.json.writeStream( out, .{ .whitespace = .indent_2 });
58:
59: //-----------------------------------
60: // Panel JSON
61: //-----------------------------------
62:
63:     const Ipanel = std.enums.EnumIndexer(pnl.Epanel);
64:
65:
66:     w.beginObject()   catch |err| {
67:                 @panic(try std.fmt.allocPrint(allocator,"err Open FILE.{any}\n", .{err}));};
68:
69:
70:     try w.objectField("PANEL");
71:     const nbrPnl: usize = XPANEL.items.len;
72:     var np: usize = 0;
73:     while (np < nbrPnl) : (np += 1) {
74:         try w.beginArray();
75:         try w.beginObject();
76:         var p: usize = 0;
77:         while (p < Ipanel.count) : (p += 1) {
78:             switch (Ipanel.keyForIndex(p)) {
79:                 .name => {
80:                     try w.objectField(@tagName(pnl.Epanel.name));
81:                     try w.print("\"{s}\"", .{XPANEL.items[np].name});
82:                 },
83:                 .posx => {
84:                     try w.objectField(@tagName(pnl.Epanel.posx));
85:                     try w.print("{d}", .{XPANEL.items[np].posx});
86:                 },
87:                 .posy => {
88:                     try w.objectField(@tagName(pnl.Epanel.posy));
89:                     try w.print("{d}", .{XPANEL.items[np].posy});
90:                 },
91:                 .lines => {
92:                     try w.objectField(@tagName(pnl.Epanel.lines));
```

```
 93:                     try w.print("{d}", .{XPANEL.items[np].lines});
 94:                 },
 95:                 .cols  => {
 96:                     try w.objectField(@tagName(pnl.Epanel.cols));
 97:                     try w.print("{d}", .{XPANEL.items[np].cols});
 98:                 },
 99:                 .cadre => {
100:                     try w.objectField(@tagName(pnl.Epanel.cadre));
101:                     try w.print("\"{s}\"", .{@tagName(XPANEL.items[np].frame.cadre)});
102:                 },
103:                 .title => {
104:                     try w.objectField(@tagName(pnl.Epanel.title));
105:                     try w.print("\"{s}\"", .{XPANEL.items[np].frame.title});
106:                 },
107:                 .button => {
108:                     const Ibutton = std.enums.EnumIndexer(btn.Ebutton);
109:                     const nbrBtn: usize = XPANEL.items[np].button.items.len;
110:                     var bp: usize = 0;
111:                     try w.objectField("button");
112:                     try w.beginArray();
113:                     while (bp < nbrBtn) : (bp += 1) {
114:                         try w.beginObject();
115:                         var b: usize = 0;
116:                         while (b < Ibutton.count) : (b += 1) {
117:                             switch (Ibutton.keyForIndex(b)) {
118:                                 .name => {
119:                                     try w.objectField(@tagName(btn.Ebutton.name));
120:                                     try w.print("\"{s}\"", .{XPANEL.items[np].button.items[bp].name});
121:                                 },
122:                                 .key => {
123:                                     try w.objectField(@tagName(btn.Ebutton.key));
124:                                     try w.print("\"{s}\"", .{@tagName(XPANEL.items[np].button.items[bp].key)});
125:                                 },
126:                                 .show => {
127:                                     try w.objectField(@tagName(btn.Ebutton.show));
128:                                     if (@intFromBool(XPANEL.items[np].button.items[bp].show) == 1)
129:                                                 try w.print("true", .{})
130:                                     else try w.print("false", .{});
131:                                 },
132:                                 .check => {
133:                                     try w.objectField(@tagName(btn.Ebutton.check));
134:                                     if (@intFromBool(XPANEL.items[np].button.items[bp].check) == 1)
135:                                                 try w.print("true", .{})
136:                                     else try w.print("false", .{});
137:                                 },
138:                                 .title => {
```

```
139:                                     try w.objectField(@tagName(btn.Ebutton.title));
140:                                     try w.print("\"{s}\"", .{XPANEL.items[np].button.items[bp].title});
141:                                 },
142:                             }
143:                         }
144:                         try w.endObject();
145:                     }
146:
147:                     try w.endArray();
148:                 },
149:                 .label => {
150:                     const Ilabel = std.enums.EnumIndexer(lbl.Elabel);
151:                     var l: usize = 0;
152:                     const nbrLbl: usize = XPANEL.items[np].label.items.len;
153:
154:                     var lp: usize = 0;
155:                     try w.objectField("label");
156:                     try w.beginArray();
157:                     while (lp < nbrLbl) : (lp += 1) {
158:                         try w.beginObject();
159:                         l = 0;
160:                         while (l < Ilabel.count) : (l += 1) {
161:                             switch (Ilabel.keyForIndex(l)) {
162:                                 .name => {
163:                                     try w.objectField(@tagName(lbl.Elabel.name));
164:                                     try w.print("\"{s}\"", .{XPANEL.items[np].label.items[lp].name});
165:                                 },
166:                                 .posx => {
167:                                     try w.objectField(@tagName(lbl.Elabel.posx));
168:                                     try w.print("{d}", .{XPANEL.items[np].label.items[lp].posx});
169:                                 },
170:                                 .posy => {
171:                                     try w.objectField(@tagName(lbl.Elabel.posy));
172:                                     try w.print("{d}", .{XPANEL.items[np].label.items[lp].posy});
173:                                 },
174:                                 .text => {
175:                                     try w.objectField(@tagName(lbl.Elabel.text));
176:                                     try w.print("\"{s}\"", .{XPANEL.items[np].label.items[lp].text});
177:                                 },
178:                                 .title => {
179:                                     try w.objectField(@tagName(lbl.Elabel.title));
180:                                     if (@intFromBool(XPANEL.items[np].label.items[lp].title) == 1)
181:                                         try w.print("true", .{})
182:                                     else try w.print("false", .{});
183:
184:                                 },
```

```
185:                                 }
186:                             }
187:                             try w.endObject();
188:                         }
189:
190:                         try w.endArray();
191:                     },
192:                     .field => {
193:                         const Ifield = std.enums.EnumIndexer(fld.Efield);
194:                         var f: usize = 0;
195:                         const nbrFld: usize = XPANEL.items[np].field.items.len;
196:
197:                         var fp: usize = 0;
198:                         try w.objectField("field");
199:                         try w.beginArray();
200:                         while (fp < nbrFld) : (fp += 1) {
201:                             try w.beginObject();
202:                             f = 0;
203:                             while (f < Ifield.count) : (f += 1) {
204:                                 switch (Ifield.keyForIndex(f)) {
205:                                     .name => {
206:                                         try w.objectField(@tagName(fld.Efield.name));
207:                                         try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].name});
208:                                     },
209:                                     .posx => {
210:                                         try w.objectField(@tagName(fld.Efield.posx));
211:                                         try w.print("{d}", .{XPANEL.items[np].field.items[fp].posx});
212:                                     },
213:                                     .posy => {
214:                                         try w.objectField(@tagName(fld.Efield.posy));
215:                                         try w.print("{d}", .{XPANEL.items[np].field.items[fp].posy});
216:                                     },
217:                                     .reftyp => {
218:                                         try w.objectField(@tagName(fld.Efield.reftyp));
219:                                         try w.print("\"{s}\"", .{@tagName(XPANEL.items[np].field.items[fp].reftyp)});
220:                                     },
221:                                     .width => {
222:                                         try w.objectField(@tagName(fld.Efield.width));
223:                                         try w.print("{d}", .{XPANEL.items[np].field.items[fp].width});
224:                                     },
225:                                     .scal => {
226:                                         try w.objectField(@tagName(fld.Efield.scal));
227:                                         try w.print("{d}", .{XPANEL.items[np].field.items[fp].scal});
228:                                     },
229:                                     .text => {
230:                                         try w.objectField(@tagName(fld.Efield.text));
```

```
231:                                    try w.print("\"\"", .{});
232:                                },
233:                                .requier => {
234:                                    try w.objectField(@tagName(fld.Efield.requier));
235:                                    if (@intFromBool(XPANEL.items[np].field.items[fp].requier) == 1)
236:                                                try w.print("true", .{})
237:                                    else try w.print("false", .{});
238:                                },
239:                                .protect => {
240:                                    try w.objectField(@tagName(fld.Efield.protect));
241:                                    if (@intFromBool(XPANEL.items[np].field.items[fp].protect) == 1)
242:                                                try w.print("true", .{})
243:                                    else try w.print("false", .{});
244:                                },
245:                                .edtcar => {
246:                                    try w.objectField(@tagName(fld.Efield.edtcar));
247:                                    try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].edtcar});
248:                                },
249:                                .errmsg => {
250:                                    try w.objectField(@tagName(fld.Efield.errmsg));
251:                                    try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].errmsg});
252:                                },
253:                                .help => {
254:                                    try w.objectField(@tagName(fld.Efield.help));
255:                                    try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].help});
256:                                },
257:                                .procfunc => {
258:                                    try w.objectField(@tagName(fld.Efield.procfunc));
259:                                    try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].procfunc});
260:                                },
261:                                .proctask => {
262:                                    try w.objectField(@tagName(fld.Efield.proctask));
263:                                    try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].proctask});
264:                                },
265:                                .progcall => {
266:                                    try w.objectField(@tagName(fld.Efield.progcall));
267:                                    try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].progcall});
268:                                },
269:                                .typecall => {
270:                                    try w.objectField(@tagName(fld.Efield.typecall));
271:                                    try w.print("\"{s}\"", .{XPANEL.items[np].field.items[fp].typecall});
272:                                },
273:                                .parmcall => {
274:                                    try w.objectField(@tagName(fld.Efield.parmcall));
275:                                    if (@intFromBool(XPANEL.items[np].field.items[fp].parmcall) == 1)
276:                                                try w.print("true", .{})
```

```
277:                                    else try w.print("false", .{});
278:                                },
279:                                .regex=> {
280:                                    try w.objectField(@tagName(fld.Efield.regex));
281:                                    try w.print("\"\"", .{});
282:                                },
283:
284:                            }
285:                        }
286:                        try w.endObject();
287:                    }
288:
289:                    try w.endArray();
290:                },
291:                .linev => {
292:                    const Ilinev = std.enums.EnumIndexer(lnv.Elinev);
293:                    var lx: usize = 0;
294:                    const nbrLineh: usize = XPANEL.items[np].linev.items.len;
295:
296:                    var lv: usize = 0;
297:                    try w.objectField("linev");
298:                    try w.beginArray();
299:                    while (lv < nbrLineh) : (lv += 1) {
300:                        try w.beginObject();
301:                        lx = 0;
302:                        while (lx < Ilinev.count) : (lx += 1) {
303:                            switch (Ilinev.keyForIndex(lx)) {
304:                                .name => {
305:                                    try w.objectField(@tagName(lnv.Elinev.name));
306:                                    try w.print("\"{s}\"", .{XPANEL.items[np].linev.items[lv].name});
307:                                },
308:                                .posx => {
309:                                    try w.objectField(@tagName(lnv.Elinev.posx));
310:                                    try w.print("{d}", .{XPANEL.items[np].linev.items[lv].posx});
311:                                },
312:                                .posy => {
313:                                    try w.objectField(@tagName(lnv.Elinev.posy));
314:                                    try w.print("{d}", .{XPANEL.items[np].linev.items[lv].posy});
315:                                },
316:                                .lng => {
317:                                    try w.objectField(@tagName(lnv.Elinev.lng));
318:                                    try w.print("{d}", .{XPANEL.items[np].linev.items[lv].lng});
319:                                },
320:                                .trace => {
321:                                    try w.objectField(@tagName(lnv.Elinev.trace));
322:                                    try w.print("\"{s}\"", .{
```

```
323:                                         @tagName(XPANEL.items[np].linev.items[lv].trace)});
324:                                 },
325:                             }
326:                         }
327:                         try w.endObject();
328:                     }
329:                     try w.endArray();
330:                 },
331:                 .lineh => {
332:                     const Ilineh = std.enums.EnumIndexer(lnh.Elineh);
333:                     var ly: usize = 0;
334:                     const nbrLineh: usize = XPANEL.items[np].lineh.items.len;
335:
336:                     var lh: usize = 0;
337:                     try w.objectField("lineh");
338:                     try w.beginArray();
339:                     while (lh < nbrLineh) : (lh += 1) {
340:                         try w.beginObject();
341:                         ly = 0;
342:                         while (ly < Ilineh.count) : (ly += 1) {
343:                             switch (Ilineh.keyForIndex(ly)) {
344:                                 .name => {
345:                                     try w.objectField(@tagName(lnh.Elineh.name));
346:                                     try w.print("\"{s}\"", .{XPANEL.items[np].lineh.items[lh].name});
347:                                 },
348:                                 .posx => {
349:                                     try w.objectField(@tagName(lnh.Elineh.posx));
350:                                     try w.print("{d}", .{XPANEL.items[np].lineh.items[lh].posx});
351:                                 },
352:                                 .posy => {
353:                                     try w.objectField(@tagName(lnh.Elineh.posy));
354:                                     try w.print("{d}", .{XPANEL.items[np].lineh.items[lh].posy});
355:                                 },
356:                                 .lng => {
357:                                     try w.objectField(@tagName(lnh.Elineh.lng));
358:                                     try w.print("{d}", .{XPANEL.items[np].lineh.items[lh].lng});
359:                                 },
360:                                 .trace => {
361:                                     try w.objectField(@tagName(lnh.Elineh.trace));
362:                                     try w.print("\"{s}\"", .{
363:                                         @tagName(XPANEL.items[np].lineh.items[lh].trace)});
364:                                 },
365:                             }
366:                         }
367:                         try w.endObject();
368:                     }
```

```
369:                     try w.endArray();
370:                 },
371:             }
372:         }
373:         try w.endObject();
374:         try w.endArray();
375:     }
376:     const nbrMenu: usize = XMENU.items.len;
377:     const nbrGrid: usize = XMENU.items.len;
378:     if ( nbrGrid == 0 and nbrMenu == 0)try w.endObject();
379:
380: //----------------------------------
381: // Grid JSON
382: //----------------------------------
383:     if ( nbrGrid > 0 ) {
384:     const Igrid = std.enums.EnumIndexer(grd.Egrid);
385:     try w.objectField("GRID");
386:     var ng: usize = 0;
387:
388:     try w.beginArray();
389:     while (ng < nbrGrid) : (ng += 1) {
390:
391:         try w.beginObject();
392:         var g: usize = 0;
393:         while (g < Igrid.count) : (g += 1) {
394:             switch (Igrid.keyForIndex(g)) {
395:                 .name => {
396:                     try w.objectField(@tagName(grd.Egrid.name));
397:                     try w.print("\"{s}\"", .{XGRID.items[ng].name});
398:                 },
399:                 .posx => {
400:                     try w.objectField(@tagName(grd.Egrid.posx));
401:                     try w.print("{d}", .{XGRID.items[ng].posx});
402:                 },
403:                 .posy => {
404:                     try w.objectField(@tagName(grd.Egrid.posy));
405:                     try w.print("{d}", .{XGRID.items[ng].posy});
406:                 },
407:                 .pagerows => {
408:                     try w.objectField(@tagName(grd.Egrid.pagerows));
409:                     try w.print("{d}", .{XGRID.items[ng].pageRows});
410:                 },
411:                 .separator => {
412:                     try w.objectField(@tagName(grd.Egrid.separator));
413:                     try w.print("\"{s}\"", .{XGRID.items[ng].separator});
414:                 },
```

```
415:                 .cadre => {
416:                     try w.objectField(@tagName(grd.Egrid.cadre));
417:                     try w.print("\"{s}\"", .{@tagName(XGRID.items[ng].cadre)});
418:                 },
419:                 .cell => {
420:                     const Icell = std.enums.EnumIndexer(grd.Ecell);
421:                     var cx: usize = 0;
422:                     const nbrcell: usize = XGRID.items[ng].cell.items.len;
423:
424:                     var cv: usize = 0;
425:                     try w.objectField("cells");
426:                     try w.beginArray();
427:                     while (cv < nbrcell) : (cv += 1) {
428:                         try w.beginObject();
429:                         cx = 0;
430:                         while (cx < Icell.count) : (cx += 1) {
431:                             switch (Icell.keyForIndex(cx)) {
432:                                 .text => {
433:                                     try w.objectField(@tagName(grd.Ecell.text));
434:                                     try w.print("\"{s}\"", .{XGRID.items[ng].cell.items[cv].text});
435:                                 },
436:                                 .long => {
437:                                     try w.objectField(@tagName(grd.Ecell.long));
438:                                     try w.print("{d}", .{XGRID.items[ng].cell.items[cv].long});
439:                                 },
440:                                 .reftyp => {
441:                                     try w.objectField(@tagName(grd.Ecell.reftyp));
442:                                     try w.print("\"{s}\"", .{@tagName(XGRID.items[ng].cell.items[cv].reftyp)});
443:                                 },
444:                                 .posy => {
445:                                     try w.objectField(@tagName(grd.Ecell.posy));
446:                                     try w.print("{d}", .{XGRID.items[ng].cell.items[cv].posy});
447:                                 },
448:                                 .edtcar => {
449:                                     try w.objectField(@tagName(grd.Ecell.edtcar));
450:                                     try w.print("\"{s}\"", .{XGRID.items[ng].cell.items[cv].edtcar});
451:                                 },
452:                                 .atrcell => {
453:                                     try w.objectField(@tagName(grd.Ecell.atrcell));
454:                                     try w.print("\"{s}\"", .{@tagName(XGRID.items[ng].cell.items[cv].atrCell.foregr)});
455:                                 }
456:                             } // end switch
457:                         } // end wile field cell
458:                         try w.endObject();
459:                     } // end nbr cell
460:                     try w.endArray();
```

```
461:                     try w.endObject();
462:                 },
463:                 .data => {
464:                 },
465:             }
466:         }
467:     }
468:     try w.endArray();
469:     if ( nbrMenu == 0 )try w.endObject();
470: }
471:
472:
473: //-----------------------------------
474: // Menu JSON
475: //-----------------------------------
476:     if ( nbrMenu > 0 ) {
477:     const Imenu = std.enums.EnumIndexer(mnu.Emenu);
478:     try w.objectField("MENU");
479:     var ng: usize = 0;
480:
481:     try w.beginArray();
482:     while (ng < nbrMenu) : (ng += 1) {
483:         try w.beginObject();
484:         var m: usize = 0;
485:         while (m < Imenu.count) : (m += 1) {
486:             switch (Imenu.keyForIndex(m)) {
487:                 .name => {
488:                     try w.objectField(@tagName(mnu.Emenu.name));
489:                     try w.print("\"{s}\"", .{XMENU.items[ng].name});
490:                 },
491:                 .posx => {
492:                     try w.objectField(@tagName(mnu.Emenu.posx));
493:                     try w.print("{d}", .{XMENU.items[ng].posx});
494:                 },
495:                 .posy => {
496:                     try w.objectField(@tagName(mnu.Emenu.posy));
497:                     try w.print("{d}", .{XMENU.items[ng].posy});
498:                 },
499:                 .cadre => {
500:                     try w.objectField(@tagName(mnu.Emenu.cadre));
501:                     try w.print("\"{s}\"", .{@tagName(XMENU.items[ng].cadre)});
502:                 },
503:                 .mnuvh => {
504:                     try w.objectField(@tagName(mnu.Emenu.mnuvh));
505:                     try w.print("\"{s}\"", .{@tagName(XMENU.items[ng].mnuvh)});
506:                 },
```

```
507:
508:                 .xitems => {
509:                     const Iopt = std.enums.EnumIndexer(mnu.Eopt);
510:                     var cx: usize = 0;
511:                     const nbrcell: usize = XMENU.items[ng].xitems.len;
512:
513:                     var cv: usize = 0;
514:                     try w.objectField("xitems");
515:                     try w.beginArray();
516:                     while (cv < nbrcell) : (cv += 1) {
517:                         try w.beginObject();
518:                         cx = 0;
519:                         while (cx < Iopt.count) : (cx += 1) {
520:                             switch (Iopt.keyForIndex(cx)) {
521:                                 .text => {
522:                                     try w.objectField(@tagName(mnu.Eopt.text));
523:                                     try w.print("\"{s}\"", .{XMENU.items[ng].xitems[cv]});
524:                                 },
525:                             } // end switch
526:                         } // end wile field cell
527:                         try w.endObject();
528:                     } // end nbr cell
529:                     try w.endArray();
530:                 }
531:             }
532:         }
533:         try w.endObject();
534:     }
535:     try w.endArray();
536:     }
537:     if ( nbrMenu > 0 )  try w.endObject();
538: return ;
539: }
```