EasyQuill

Questa guida

Questa guida si riferisce al programma EasyQuill, programma per la impaginazione automatica di relazioni tecniche.

Questo manuale è aggiornato al Marzo 2019.

Accordo di licenza d'uso del software Softing Next

1. Licenza. A fronte del pagamento del corrispettivo della licenza, compreso nel prezzo di acquisto di questo prodotto, e all'osservanza dei termini e delle condizioni di questa licenza la Softing Next s.r.l., nel seguito Softing Next, cede all'acquirente, nel seguito Licenziatario, un diritto non esclusivo e non trasferibile di utilizzo di questa copia di programma software, nel seguito Software.

2. Proprietà del software. La Softing Next mantiene la piena proprietà di questa copia di programma Software e della documentazione ad essa allegata. Pertanto la Softing Next non vende alcun diritto sul Software sul quale mantiene ogni diritto.

3. Utilizzo del software. Questo Software contiene segreti commerciali. È espressamente proibito effettuare copie o modifiche o reingegnerizzazioni, sotto qualsiasi forma e con qualsiasi mezzo, anche parziali, del Software e della documentazione a esso allegata. Il Licenziatario è responsabile a tutti i fini legali per qualunque infrazione causata o incoraggiata dalla non osservanza dei termini di questa licenza. È consentito effettuare una sola copia del Software esclusivamente per installazione su un solo disco rigido.

4. Cessione del software. Il software viene ceduto in licenza unicamente al Licenziatario e non può essere ceduto a terzi. In nessun caso è consentito cedere, assegnare, affidare, affittare o disporre in altro modo del Software se non nei termini qui espressamente specificati.

5. Cessazione. Questa licenza ha la durata di anni dieci. Il Licenziatario può porvi termine in ogni momento con la completa distruzione del Software. Questa licenza si intende cessata, senza onere di comunicazione da parte di Softing Next, qualora v sia inadempienza da parte del Licenziatario delle condizioni della licenza.

6. Esonero della garanzia del software. Il Licenziatario si fa carico di ogni rischio derivante, dipendente e connesso all'uso de Software. Il Software e la relativa documentazione vengono forniti nello stato in cui si trovano. Softing Next si esonera espressamente da ogni garanzia espressa o implicita ivi inclusa, ma senza limitazioni, la garanzia implicita di commerciabilità e di idoneità del prodotto a soddisfare particolari scopi. Softing Next non garantisce che le funzioni contenute nel Software siano idonee a soddisfare le esigenze del Licenziatario né garantisce una operatività ininterrotta o immune da difetti del Software né che i difetti riscontrati nel software vengano corretti. Softing Next non garantisce l'uso o i risultati derivanti dall'uso del Software e della documentazione né la loro correttezza, affidabilità e accuratezza. Le eventuali informazioni oral o scritte di esponenti o incaricati di Softing Next non inficiano questo esonero di garanzia.

7. Limitazioni di responsabilità. Softing Next è espressamente sollevata da ogni responsabilità per qualsiasi danno, diretto o indiretto, di ogni genere e specie, derivante dall'uso o dal non uso del Software e della relativa documentazione. In ogni casc i limiti di responsabilità di Softing Next nei confronti del Licenziatario per il complesso dei danni, delle perdite, e per ogni altra causa, sarà rappresentato dall'importo dal Licenziatario corrisposto a Softing Next per il relativo Software.

8. Foro esclusivo. In caso di controversie relative a questo accordo, sarà esclusivamente competente a decidere l'Autorità Giudiziaria di Roma.

9. Obbligatorietà ed interezza dell'Accordo. Il Licenziatario, avendo letto il testo che precede ed avendo riscontrato che questa Licenza e la Garanzia Limitata che contiene sono accettabili, le accetta senza condizioni e conferma, con l'atto di accettare l'installazione del Software, la sua volontà di vincolarsi alla scrupolosa osservanza di questo Accordo. Il Licenziatari dà altresì atto che quanto precede costituisce la totalità delle intese intercorse e che pertanto esso annulla e sostituisce ogni eventuale precedente accordo o comunicazione tra le parti.

SOFTING NEXT NON GARANTISCE CHE LE FUNZIONI CONTENUTE NEL SOFTWARE SIANO IDONEE A SODDISFARE LE ESIGENZE DEL LICENZIATARIO NÉ GARANTISCE UNA OPERATIVITÀ ININTERROTTA O IMMUNE DA DIFETTI DEL SOFTWARE NÉ CHE I DIFETTI RISCONTRATI VENGANO CORRETTI. SOFTING Next NON GARANTISCE L'USO O I RISULTATI DERIVANTI DALL'USO DEL SOFTWARE E DELLA DOCUMENTAZIONE NÉ LA LORO CORRETTEZZA, AFFIDABILITÀ E ACCURATEZZA.

Le informazioni contenute in questo documento sono soggette a cambiamento senza preavviso e non costituiscono impegno alcuno da parte della Softing Next srl. Nessuna parte di questo manuale e per nessun motivo può essere utilizzata se non come aiuto all'uso del programma.

Nòlian è registrato presso il Registro Pubblico Speciale per i programmi per Elaboratore in data 14/07/2000 al progressivo 001629, ordinativo D002017; EasyBeam in data 14/05/96 al progressivo 000348, ordinativo D000409; EasySteel in data 14/05/96 al progressivo 000346, ordinativo D000407; EasyWall in data 14/05/96 al progressivo 000347, ordinativo D000408, MacSap in data 23/11/97 al progressivo 000222, ordinativo D000264, ArchiLink in data 14/07/2000 al progressivo 001630, ordinativo D002018.

Softing®, il logo Softing, il logo Softing Next, Nòlian®, il logo Nòlian®, Mac-Sap®, MacBeam®, CADSap®, EasyWall®, EasySteel®, EasyBeam®, EasyFrame®, EasyWorld®, HyperGuide®, Sap-Script®, FreeLite®, inMod®, sono marchi registrati di Softing Next s.r.l.

Novità

Le novità qui descritte sono quelle disponibili alla data della ultimazione di questa Guida rispetto all'edizione rilasciata con EWS 56.

- Aggiunte nuove funzioni per il supporto delle nuove funzionalità di EasyWorld e Quilian

  Aggiunta funzione per stampare i valori e graficizzare le forzanti e gli spettri nella funzione Forzanti di EarthquakeEngineering

Presentazione di EasyQuill

EasyQuill consente di compilare automaticamente un documento attraverso una combinazione programmata di elementi preconfigurati o generati opportunamente.

EasyQuill nasce dalla ricorrente necessità, nella stesura di relazioni tecniche, di mettere insieme delle parti fisse, ripetitive e personalizzate completate da un certo numero di dati variabili. Esso facilita il compito di organizzare e mettere insieme testi predefiniti da utilizzare in casi diversi e dati provenienti dal calcolo e dalla progettazione, riducendo la possibilità di confusione ed errori, attraverso uno schema preordinato, "intelligente" ed altamente personalizzabile.

Come opera EasyQuill

Il programma permette di definire una struttura gerarchica di blocchi. Durante la generazione questa struttura viene analizzata ordinatamente ed ogni blocco eseguito per ottenere il documento finale.

Un blocco può ad esempio contenere un brano di testo da inserire nel documento finale. Il testo può essere fisso, generato sul momento o letto da un file: funzionalità come queste assicurano un veloce adattamento per l'uso in più relazioni senza l'onere da parte dell'utente di mantenere aggiornate diverse revisioni che differiscono per pochissimi elementi dipendenti dal contesto della relazione.

EasyQuill comprende un potente word processor WYSIWYG utilizzabile sia per specificare il testo in un blocco, sia per modificare il documento generato prima del salvataggio o della stampa. Il testo formattato viene salvato e caricato in formato standard RTF e può quindi essere scambiato con i più comuni word processor in circolazione.

Il contenuto del documento può essere determinato automaticamente durante la generazione attraverso blocchi funzione specializzati (ad esempio per inserire un'immagine caricandola da un file) oppure attraverso l'esecuzione di script contenuti in blocchi di comandi.

Durante la generazione del documento può essere richiesto l'intervento interattivo dell'utente attraverso l'uso di dialoghi e moduli di inserimento. Attraverso il linguaggio di script è possibile configurare accuratamente il tipo di interazione con l'utente per ottenere ad esempio valori numerici, etichette o brani di testo, file da inserire o interpretare, parametri per l'interazione con gli altri software della suite EasyWorld.

Il linguaggio di scripting

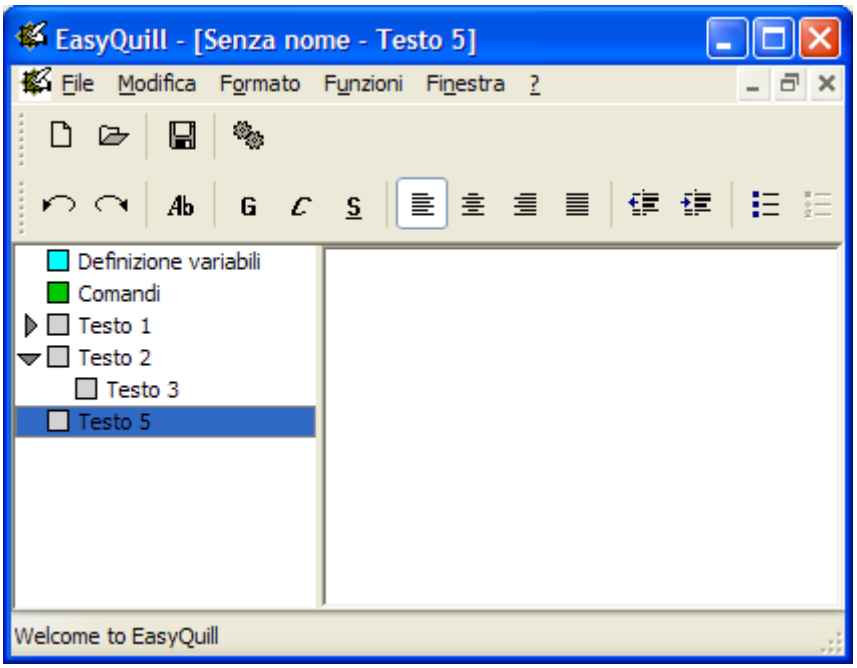
EasyQuill adotta un linguaggio di scripting che cerca di conciliare la semplicità con la potenza.

Il sofisticato motore di scripting si basa su Lua, uno standard industriale Open Source conosciuto a livello internazionale ed adottato con successo da molti altri software commerciali e non. Lua si distingue per la semplicità con cui può essere esteso ed adattato ad un software preesistente, per l'espressività e l'efficienza del linguaggio.

Il motore di scripting di EasyQuill permette tra le altre cose calcoli avanzati, manipolazione di stringhe, gestione di tabelle ed oggetti; inoltre offre funzionalità per l'interazione con l'utente, l'interrogazione dei dati nei software della suite EasyWorld, il disegno di grafici e tabelle e la generazione di testo formattato.

Lua è prodotto e distribuito da PUC-Rio, che ne detiene il copyright, presso http://www.lua.org/.

Panoramica sugli elementi di interfaccia



All'apertura, EasyQuill presenta un'interfaccia formata da:

- una barra dei menu in alto che offre accesso alle funzioni di caricamento, salvataggio, generazione del documento e a funzioni specifiche per il testo quando l'editor è attivo;
- una toolbar che dà accesso veloce ad alcune funzioni del menu File e a funzioni per la formattazione del testo;
- una finestra del documento divisa in due pannelli ridimensionabili: a sinistra è mostrata la struttura gerarchica (ad albero) del documento da generare; a destra appare l'editor relativo al blocco selezionato ed attivo nella struttura a sinistra.

La struttura del documento è formata da blocchi che offrono funzioni diverse identificate ognuna da un quadratino di colore diverso. Le funzioni dei blocchi sono descritte in dettaglio nel capitolo "I blocchi di struttura".

Un blocco può essere trascinato tenendo premuto il tasto sinistro del mouse per essere spostato in un punto diverso della struttura. Ogni blocco può contenere altri blocchi; dalla teoria dei grafi chiamiamo il blocco contenitore padre e i blocchi contenuti figli. Per inserire un blocco in un altro blocco è sufficiente trascinare il primo sopra al secondo oppure tra i figli già presenti. Quando un blocco ha dei figli alla sua sinistra appare una freccia: essa punta in basso quando i figli sono mostrati e verso destra quando essi sono nascosti; per mostrare o nascondere i figli di un blocco è sufficiente fare un click sulla freccia. figli di un blocco sono distinti da una posizione spostata verso destra rispetto al padre.

Su ogni blocco sono disponibili delle operazioni a cui si accede da un menu di contesto che si apre con un singolo click col tasto destro del mouse sul blocco stesso. Le prime operazioni sono specifiche del tipo di blocco e sono descritte in seguito. Alcune operazioni sono disponibili su tutti i blocchi:

Rinomina	Per cambiare nome al blocco.
Copia	Copia negli appunti il blocco corrente e tutti i figli per successive operazioni di Incolla.

Incolla	Inserisce nel blocco corrente i blocchi memorizzati negli appunti.
Duplica	Crea una copia del blocco selezionato e la inserisce alla fine del livello gerarchico corrente.
Elimina	Per eliminare il blocco e tutti gli eventuali figli.
Disabilitato	Per evitare che il blocco venga elaborato nella generazione del documento di uscita.
Aggiungi blocco	Per aggiungere un blocco di un certo tipo come figlio al blocco selezionato; se non c'è alcun blocco selezionato, il blocco viene aggiunto in fondo alla gerarchia.
Inserisci blocchi da file	Legge un file EasyQuill e ne copia la struttura all'interno del nodo selezionato.
Salva blocco su file	Memorizza il nodo selezionato e tutti gli eventuali figli su un file EasyQuill; le ultime due funzioni possono essere utilizzate per copiare un blocco da un file EasyQuill ad un altro.
Espandi tutto/Contrai tutto	Per espandere o contrarre la visualizzazione dei figli del nodo selezionato o di tutto l'albero se non c'è alcun nodo selezionato.

L'editor di testo

EasyQuill incorpora un editor WYSIWYG per il testo formattato che offre caratteristiche simili a quelle dei più comuni word processor. L'editor viene utilizzato sia per la modifica dei blocchi di testo della struttura, sia per la modifica degli script (in modalità solo testo), sia per la modifica del documento generato prima del salvataggio o della stampa.

L'editor supporta:

- Il salvataggio e il caricamento di testo RTF per lo scambio dei dati con i word processor più comuni.
- Lo scambio con altri programmi di testo formattato e di immagini attraverso gli appunti.
- L'inserimento di oggetti OLE per incorporare immagini e dati provenienti da software eterogenei.
- La generazione del sommario, delle intestazioni e dei piè di pagina personalizzati in stampa.
- La ricerca rapida nel testo.
- La memorizzazione di stili carattere personalizzati per una più rapida assegnazione delle caratteristiche del testo durante la digitazione.
- L'inserimento di semplici tabelle configurabili.
- L'impaginazione in stampa corretta con gestione di interruzioni di pagina, del controllo degli orfani e dei parametri di paragrafo.
- L'annullamento e il ripristino multipli delle operazioni.
- La colorazione della sintassi durante la modifica degli script e della descrizione dei form, utile per distinguere le

diverse parti che formano uno script: commenti, identificatori, parole chiave, chiamate a funzione, costanti, operatori...

I menu dedicati all'editor di testo

Le funzioni dell'editor sono accessibili attraverso i menu Modifica, Formato e Funzioni. Alcune funzioni dei menu sono disponibili anche nella toolbar, nel menu contestuale e attraverso tasti modificatori come scorciatoie.

Nel menu Modifica sono presenti i comandi:

Annulla e Ripristina	Per annullare o ripristinare le più recenti modifiche.
Taglia, Copia, Incolla, Duplica, Elimina	Per le comuni operazioni sul testo selezionato di interazione con la clipboard.
Seleziona tutto	Per selezionare l'intero contenuto dell'editor.
Trova	Per cercare nel contenuto dell'editor una o più parole.
Trova successivo	Per cercare ancora con le impostazioni di ricerca più recentemente utilizzate.
Imposta testo da cercare	Per impostare come testo di ricerca quello correntemente selezionato.
Vai alla linea	Per andare velocemente alla linea di testo specificata.

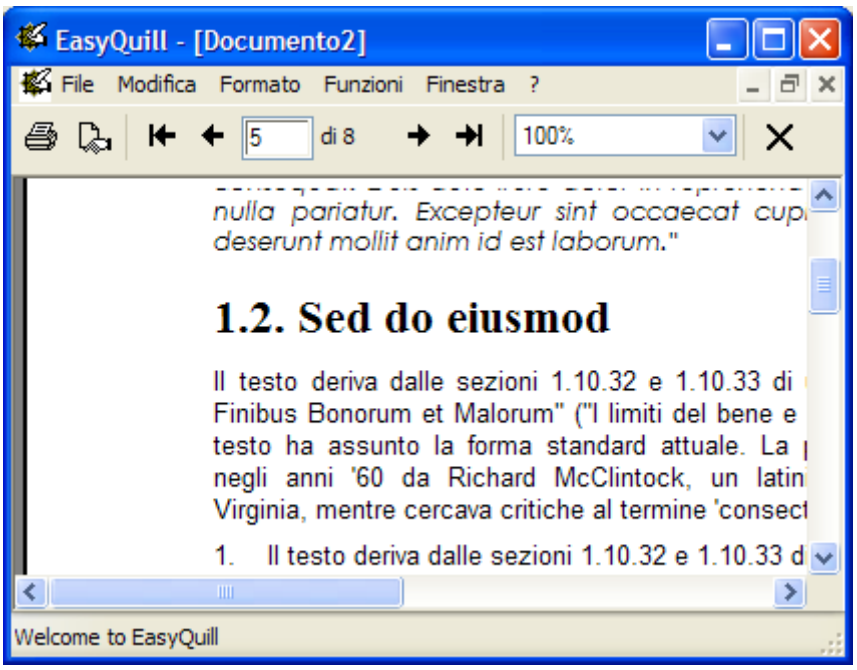
Dal menu Formato è possibile accedere a funzioni per specificare gli attributi del testo (attributi di carattere o di paragrafo), l'allineamento, il tipo di numerazione e così via. Dal documento EasyQuill è possibile modificare anche il formato di default per i nuovi blocchi di testo e il formato di intestazione e piè di pagina del documento da generare (ved. capitolo dedicato).

Il menu Funzioni contiene alcune operazioni speciali sul testo, come l'inserimento di una tabella, di un oggetto OLE o di un'interruzione di pagina. È anche possibile inserire un file (RTF, di testo o un'immagine) o salvare il testo selezionato su file RTF. Durante la modifica di un blocco di testo sono anche disponibili funzioni per l'inserimento di un blocco funzionale (ved. capitolo "I blocchi funzionali").

Quando si sta modificando un documento generato si aggiungono le funzioni per il salvataggio e il caricamento di file RTF, per impostare la stampante, visualizzare l'anteprima di stampa ed avviare la stampa. Tutte queste funzioni sono accessibili dal menu File. Si aggiunge inoltre una voce nel menu Formato per accedere alle opzioni per la generazione del sommario (ved. capitolo relativo).

L'anteprima di stampa

L'anteprima di stampa è disponibile nella modifica dei documenti generati da EasyQuill. Viene attivata attraverso l'omonima voce del menu File e si controlla interamente dalla toolbar mostrata in figura:



La prima icona avvia la stampa, la seconda imposta la pagina; seguono i controlli per la navigazione tra le pagine; quindi il fattore di zoom e il bottone per la chiusura dell'anteprima.

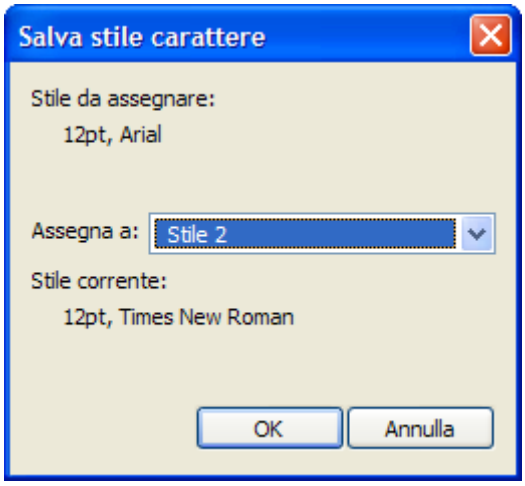
Durante l'anteprima di stampa la modifica del documento non è permessa ma rimangono attive nel menu Formato le voci relative alle opzioni delle intestazioni e piè di pagina e della generazione del sommario.

Stili di carattere personalizzati

Nell'editor di testo formattato è possibile impostare fino a sei stili personalizzati di carattere, ovvero fotografare gli attributi assegnati al testo per poi poterli riutilizzare in ogni momento su una qualunque selezione.

Le funzioni per memorizzare ed utilizzare gli stili sono accessibili attraverso il menu "Stili definiti dall'utente" del menu Formato oppure attraverso la toolbar "Stili caratteri". In entrambi i casi è possibile scegliere una voce del tipo "Stile *N*", dove *N* è un numero tra 1 e 6, per assegnare alla selezione di testo corrente lo stile indicato. Se non c'è alcuna selezione attiva lo stile viene applicata ai caratteri digitati da quel momento in poi.

Un'altra funzione sugli stili è "Assegna da selezione", che assegna lo stile di carattere della selezione corrente ad uno dei sei stili disponibili, mostrando il seguente dialogo:



Premendo OK sarà chiesto se si desidera assegnare il nuovo stile a tutto il testo del documento corrente a cui sono assegnati gli attributi dello stile che verrà sostituito. Nel caso del documento EasyQuill, la sostituzione avviene in tutti i blocchi di testo del documento.

L'ultima funzione, "Modifica stile corrente", mostra il classico dialogo per la modifica degli attributi del carattere per cambiare lo stile corrente. Confermando le modifiche allo stile, verrà chiesto se applicare le modifiche a tutto il testo del documento corrente a cui gli attributi precedenti sono assegnati.

I blocchi di struttura

Un documento EasyQuill è formato da una struttura gerarchica di blocchi elementari. Ogni blocco assolve ad una particolare funzione al momento della generazione del documento di uscita.

Sono disponibili diversi tipi di blocco, ognuno con caratteristiche e funzionalità peculiari che vengono descritte in dettaglio nei successivi capitoli.

Blocco di testo

Il blocco di testo è l'elemento più semplice della struttura di un documento EasyQuill. Esso rappresenta un brano di testo da inserire nel documento di uscita in relazione alla sua posizione nella gerarchia. Il brano può essere nella maggior parte dei casi considerato come un capitolo o un paragrafo del documento finale.

Nella struttura ad albero un blocco di testo è identificato dal colore grigio. Al momento della creazione viene chiesto il nome del blocco: il nome viene normalmente riportato nel documento finale come titolo di un capitolo o di un paragrafo. Per modificare il titolo del brano, si può fare doppio clic sul blocco oppure scegliere Rinomina dal menu contestuale (ved. capitolo "Panoramica sugli elementi di interfaccia"). Per evitare che il titolo venga inserito nel documento finale è sufficiente togliere la spunta dalla voce "Aggiungi titolo" del menu contestuale del blocco.

Se un blocco si trova al primo livello della gerarchia del documento e ha l'opzione "Aggiungi titolo" attivata, nella generazion del documento finale esso creerà in uscita un titolo di capitolo basato sul nome del blocco e una relativa voce nel sommario. Se il blocco ha come figli blocchi di testo con l'opzione "Aggiungi titolo" attivata, essi saranno considerati paragrafi di quel capitolo.

In generale per ogni blocco di testo che ha l'opzione "Aggiungi titolo" attiva lungo un percorso della gerarchia aumenta il livello corrente del testo che viene inserito. Usando ad esempio una numerazione comune per un libro, un blocco al primo livello sarà considerato il "Capitolo V", un blocco di testo suo figlio sarà il paragrafo "V.3", un blocco ancora più interno sarà "V.3.2" e così via.

Selezionando un blocco di testo nell'albero della struttura, viene mostrato nel pannello destro l'editore di testo formattato con il contenuto del blocco. Nel testo possono essere contenuti dei blocchi funzionali, ovvero degli oggetti che vengono interpretati nel momento della generazione per modificare in modo dinamico il contenuto del testo. I blocchi funzionali sonc discussi in un successivo capitolo.

Espressioni di scripting nel titolo

Nel nome del blocco possono essere inserite espressioni introdotte dal simbolo di cancelletto '#': queste espressioni vengon interpretate al momento della generazione del titolo dal motore di scripting e sostituite col valore calcolato.

Le espressioni di scripting valide sono di due tipi: semplici identificatori oppure espressioni complesse. Un'espressione semplice può essere il nome di una variabile e rispetta la convenzione dei nomi vista nel capitolo "Convenzioni lessicali": in

questo caso viene semplicemente sostituito il valore di quella variabile. Un'espressione complessa va racchiusa tra parentesi tonde e può contenere una o più istruzioni, riferimenti a tabella, chiamate a funzione ecc.; viene sostituita con il primo valor che essa restituisce.

Ad esempio l'espressione semplice "#var1" viene sostituita con il valore della variabile var1. L'espressione complessa "#(math.sin(math.pi/2))" viene sostituita col valore del seno a 90°. Si noti che è necessario utilizzare un'espressione complessa per accedere a valori di una tabella: è un'espressione corretta "#(v[3])", ma non "#v[3]".

Blocco di comandi

Il blocco di comandi, identificato dal colore verde, contiene una sequenza qualsiasi di comandi di scripting da eseguire durante il flusso di generazione del documento di uscita.

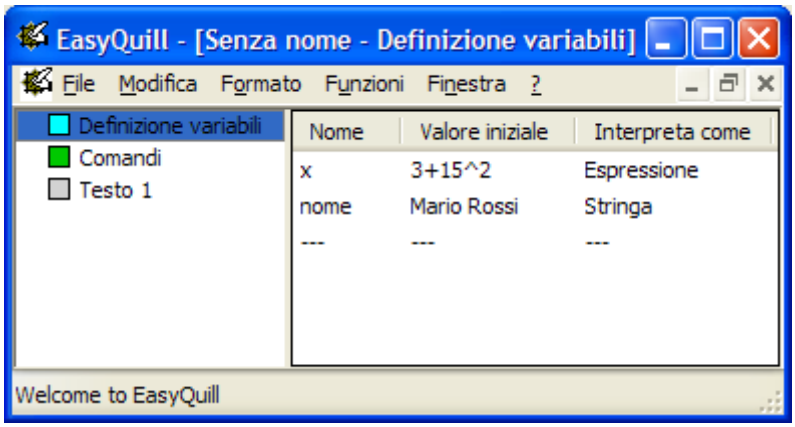
Ogni blocco comandi ha due "fasi": una fase "pre" ed una "post". L'utente passa da una fase all'altra facendo doppio clic col tasto sinistro del mouse sul blocco oppure scegliendo la fase dal menu contestuale del blocco. La fase "pre" contiene il codic da eseguire prima dei figli del blocco. La fase "post" contiene il codice da eseguire dopo aver eseguito tutti i figli. Sebbene normalmente sia sufficiente e più logico utilizzare esclusivamente la fase "pre", la fase "post" può essere utile in alcune occasioni: ad esempio per ripristinare lo stato della computazione ad uno stato memorizzato nella fase "pre".

L'input dello scripting avviene attraverso una versione solo testo dell'editore WYSIWYG mostrato nel pannello destro.

Blocco variabili

Il blocco variabili, distinto dal colore azzurro, permette di assegnare dei valori iniziali alle variabili di scripting da utilizzare nel seguito del documento.

Selezionando un blocco variabili, nel pannello destro viene mostrata una lista a quattro colonne:



La prima colonna indica il nome della variabile da assegnare, la seconda una espressione e la terza il tipo di espressione. Indicando come tipo "Espressione", l'espressione viene valutata ed il risultato assegnato alla variabile indicata. Indicando come tipo "Stringa", l'espressione viene assegnata direttamente come valore stringa. Una quarta colonna, non mostrata in figura, può contenere un commento opzionale per descrivere la variabile all'utente.

Per creare una nuova variabile è possibile modificare direttamente un campo dell'ultima riga identificata dai trattini oppure accedere al menu contestuale cliccando col tasto destro del mouse su una riga già esistente e scegliendo "Aggiungi variabile'

Dal menu contestuale è possibile anche creare una nuova variabile assegnandole il risultato di una interrogazione all'utente al momento della generazione. Queste voci sono semplicemente scorciatoie per ottenere le interrogazioni più comuni con minore sforzo. Una volta generate le assegnazioni esse possono essere ulteriormente personalizzate intervenendo

manualmente sul campo "Valore iniziale" della riga. Le voci disponibili sono:

Aggiungi richiesta si/no: scegliendo questa voce viene richiesto il testo della domanda da porre all'utente a cui può rispondere affermativamente o negativamente. Il risultato viene assegnato ad una variabile booleana vero/falso. Ved. funzione **ask_yesno**.

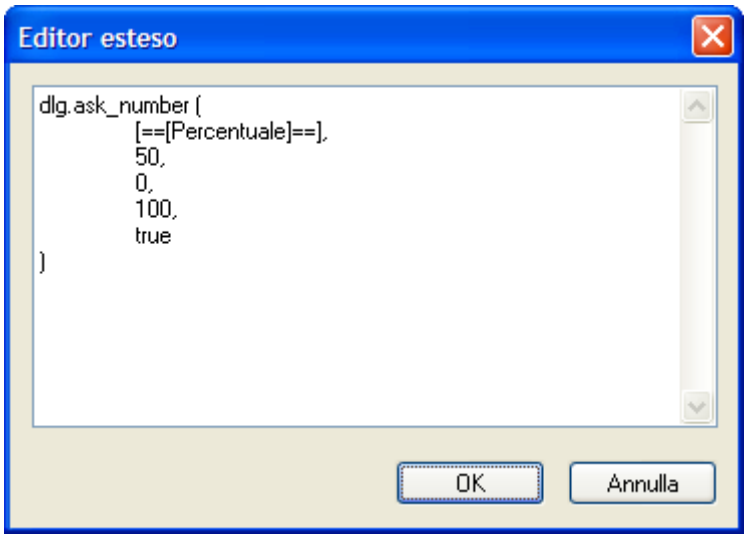
Aggiungi richiesta testo breve o **testo:** scegliendo queste voci viene richiesto il testo della domanda da porre all'utente per l richiesta di inserimento di un testo breve o di un paragrafo. Il risultato viene assegnato ad una variabile come stringa di testc Ved. funzioni **ask_text** e **ask_text_block**.

Aggiungi richiesta numero: scegliendo questa voce viene richiesto il testo della domanda da porre all'utente per la richiesta di inserimento di un valore numerico. È possibile specificare anche il valore di default, il valore minimo, il valore massimo e s il numero deve essere intero oppure può avere cifre decimali. Il risultato viene controllato in base ai parametri ed assegnato ad una variabile numerica intera o in virgola mobile. Ved. funzione **ask_number**.

Aggiungi richiesta file: scegliendo questa voce viene richiesto il titolo del dialogo per la richiesta di un file all'utente. È possibile specificare anche il filtro sui file da utilizzare a scelta tra alcuni comuni (file di testo, immagini, documenti EasyWorld...). Il nome del file scelto dall'utente viene assegnato ad una variabile come stringa di testo. Ved. funzione **ask_fil** .

Se il menu contestuale viene aperto su una riga alla prima colonna, è possibile copiare o incollare il nome della variabile. Se esso viene aperto sulla seconda colonna è possibile copiare o incollare l'espressione della variabile. Se esso viene aperto, infine, sulla quarta colonna è possibile copiare o incollare il commento della variabile.

Per tutte le colonne tranne quella del tipo di espressione è possibile accedere dal menu contestuale all' "Editore esteso", ovvero ad un dialogo che permette una modifica più comoda delle espressioni e dei commenti, con la possibilità addizionale di estendere su più righe una espressione:



Le variabili assegnate in un blocco variabili hanno sempre valore globale e le espressioni vengono interpretate dall'alto verso il basso: una variabile quindi può essere assegnata più volte e all'uscita dal blocco avrà assunto il valore assegnato nell'ultima riga che la riguarda.

Un blocco variabili può avere figli ma non ha alcun senso gerarchico: per la generazione è come se i figli fossero semplicemente blocchi da eseguire successivamente, come succederebbe se seguissero il blocco variabili allo stesso livello gerarchico.

Blocco form

Il blocco form, identificato dal colore giallo, contiene la descrizione di un modulo da presentare all'utente per assegnare interattivamente dei valori alle variabili durante l'elaborazione del documento.

La definizione del modulo avviene attraverso una versione solo testo dell'editore WYSIWYG mostrato nel pannello destro ch permette l'inserimento della descrizione in stile HTML secondo quanto spiegato nel capitolo "Sintassi per i moduli". All'uscita del modulo la variabile globale **form_exit_button** viene modificata in base alla definizione del modulo e al bottone premuto dall'utente per chiudere il dialogo.

Durante la modifica di un form è possibile verificare in ogni momento il risultato corrispondente a quanto scritto, scegliendo dal menu Funzioni la voce "Verifica finestra di dialogo" o premendo il tasto F10. Se per il test si desidera inizializzare le variabili di scripting utilizzate dal modulo (come spiegato nel capitolo "Sintassi per i moduli"), è possibile specificare (normalmente all'inizio) delle istruzioni di scripting che vengono eseguite solo durante il test all'interno di un blocco di commento HTML che comincia con l'espressione "<!--testWith:". Ad esempio se c'è una variabile *label* utilizzata nel modulo va inizializzata per il test, la descrizione del form può cominciare con:

```
<!--testWith:label="Prova"-->
```

Come per il blocco variabili (ved. paragrafo precedente), il blocco form può avere figli ma non ha alcun senso gerarchico.

Blocco di controllo

Un blocco di controllo, identificato dal colore blu, permette di controllare l'elaborazione dei blocchi che ha come figli: ne permette l'esecuzione condizionale o multipla.

La configurazione del blocco avviene interamente dal suo dialogo Proprietà, accessibile con un doppio clic col mouse sul blocco oppure dal menu contestuale:

I tipi di controllo sono quattro:

Blocco **IF**: L'elaborazione dei figli avviene solo se la condizione booleana espressa è vera; conta come vera qualsiasi espressione che si risolve in un valore diverso da **false** e **nil** (in particolare il numero 0 e la stringa vuota contano come vere).

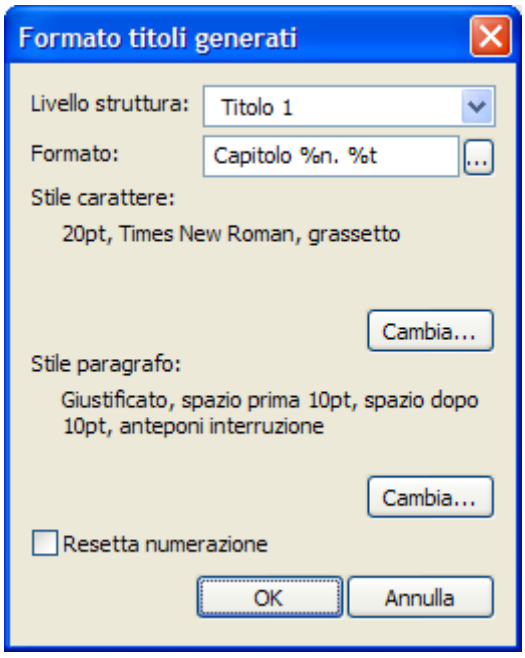
Blocco **WHILE**:	L'elaborazione dei figli viene ripetuta finché la condizione espressa rimane vera; la condizione viene controllata prima di ogni ciclo.
Blocco **REPEAT/UNTIL**:	L'elaborazione dei figli viene ripetuta finché la condizione espressa non diventa vera; la condizione viene controllata al termine di ogni ciclo, quindi i figli vengono elaborati almeno una volta.
Blocco **FOR**:	L'elaborazione dei figli avviene per il numero specificato di volte, in base ad un certo numero di iterazioni o ad un'espressione generale (descritta nel capitolo "Sintassi").

Blocco per generazione titoli

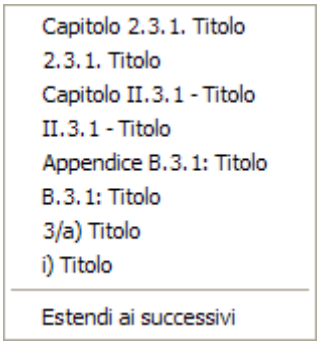
Un blocco per la generazione dei titoli, identificato dal colore rosso, permette di modificare i modelli di generazione dei titoli dei vari livelli della gerarchia nel documento finale oppure di reimpostare la numerazione adottata.

Se il blocco non ha figli, le opzioni di generazione vengono applicate all'intero documento da quel momento in avanti (indipendentemente dal suo livello nella gerarchia). Se invece il blocco ha figli, le opzioni di generazione vengono applicate solo ad essi: è quindi ad esempio possibile usare questo tipo di blocco come contenitore di una certa sezione del documento identificata da un certo modello per i titoli (introduzione, appendice...).

Le opzioni dei modelli sono impostabili attraverso il dialogo di Proprietà accessibile con un doppio clic del mouse sul blocco oppure dal menu contestuale del blocco:



Il primo controllo a tendina permette la scelta del livello gerarchico di cui modificare il modello. Segue la specifica del formato da adottare per il modello: esso è una stringa in cui alcuni gruppi di caratteri ("token") vengono sostituiti nella generazione. I formati più comuni sono disponibili nel menu a tendina che appare cliccando col mouse sul bottone con i tre punti accanto al campo di testo:



L'ultima voce del menu serve per estendere il modello del livello corrente a tutti i livelli successivi. I token vengono descritti in seguito.

Le voci successive nel dialogo sono gli stili di carattere e paragrafo da adottare per il titolo al livello correntemente selezionato. È possibile cambiare questi stili dagli appositi bottoni che danno accesso ai classici dialoghi utilizzati anche dall'editore di testo.

L'ultimo controllo nel dialogo è quello che reimposta la numerazione: se attivo la numerazione dei titoli viene reimpostata durante l'elaborazione nel punto in cui questo blocco si trova. Questa opzione può essere utile nei casi in cui il documento si diviso in diverse sezioni, ognuna con una numerazione indipendente: ad esempio per l'appendice la numerazione va senz'altro reimpostata.

I token utilizzabili nel formato del modello sono tutti introdotti da un simbolo di percentuale '%':

%t	Viene sostituito dal nome del blocco di testo: è il titolo vero e proprio.
%n	Viene sostituito dal valore in cifre arabe del livello corrente.
%R / **%r**	Viene sostituito dal valore in numeri romani (maiuscoli o minuscoli) del livello corrente.
%A / **%a**	Viene sostituito dal valore in lettere (A-Z maiuscole o minuscole) del livello corrente; dopo Z viene AA, AB e così via.

Nel caso dei valori del livello corrente è possibile specificare un formato più complesso per distinguere i valori del primo livello da quelli successivi. Il formato generale è "**%(*SL*)**", dove:

- le parentesi sono obbligatorie
- gli asterischi rappresentano un formato a scelta tra quelli descritti sopra: n, R, r, A o a; il primo rappresenta il formatc da assegnare al primo livello ed il secondo quello da assegnare agli altri; il primo formato può essere omesso per avere tutti i formati uguali
- S è un carattere qualsiasi (tranne una cifra decimale) che rappresenta il carattere di separazione tra i vari livelli; se omesso viene assunto per default il carattere punto '.'
- L è una cifra da 1 a 9 che indica il numero di livelli da considerare: ad esempio se è 1 viene scritto solo il valore del livello corrente, se è 2 viene scritto quello corrente e quello immediatamente superiore e così via; se omesso il valore di default è 9 (tutti i livelli)

Si consiglia di provare i comuni modelli di esempio proposti nel menu contestuale descritto in precedenza per prendere confidenza col sistema dei token.

I blocchi funzionali

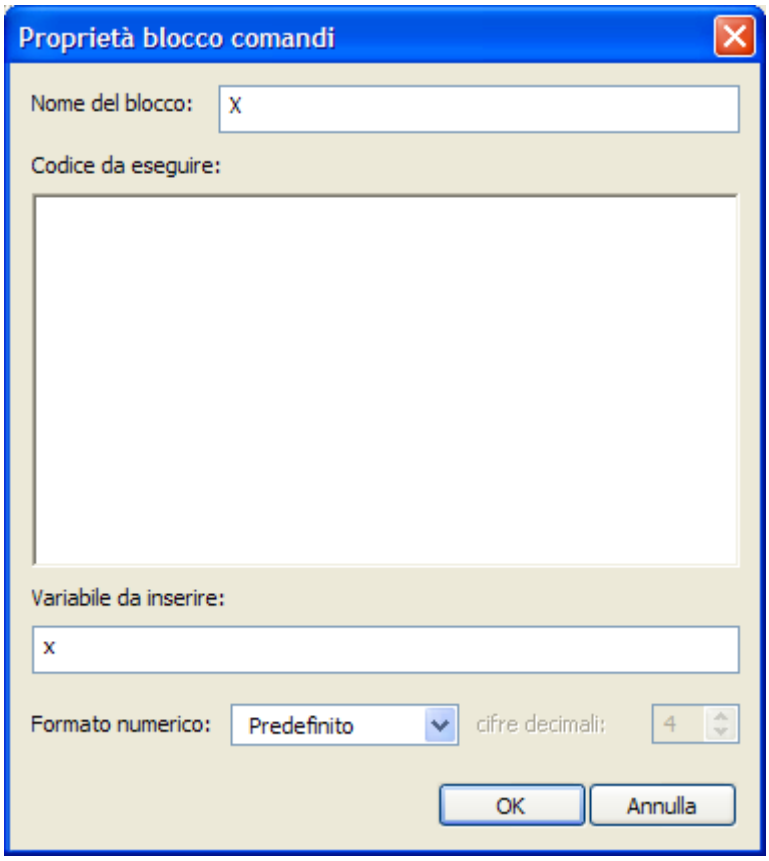
I blocchi funzionali sono oggetti inseriti all'interno del testo di un blocco di testo e che vengono eseguiti durante l'elaborazione del documento di uscita per influenzare il testo generato. Possono essere inseriti in un punto qualsiasi attraverso il menu Funzioni o dal menu contestuale dell'editore di testo. Poiché essi sono oggetti OLE a tutti gli effetti, essi possono essere trascinati, copiati ed incollati come qualsiasi parte del testo. Si avverte però che il loro uso al di fuori di EasyQuill è del tutto privo di senso e non supportato.

Ogni blocco funzionale opera in base a proprietà stabilite dall'utente nel dialogo accessibile con un doppio click del mouse o dalla voce "Proprietà oggetto" del suo menu contestuale. Tra le proprietà è sempre presente il nome del blocco, mostrato ne testo. Le altre proprietà dipendono dal tipo di blocco e sono descritte nei relativi paragrafi.

Blocco comandi

Il blocco funzionale comandi serve per inserire all'interno di un testo il valore di una variabile o del contenuto generato da uno script, in particolare quando il contenuto va immerso nel testo o lo script è particolarmente breve e non ha senso isolarl in un blocco di struttura specifico.

Il dialogo di proprietà del blocco si presenta come in figura:



Oltre al nome del blocco è possibile specificare del codice da eseguire ed una espressione da valutare (nel caso più semplice nome di una variabile) di cui inserire il valore nel testo durante l'elaborazione del testo. Entrambi i campi sono opzionali.

Il codice viene eseguito prima dell'inserimento del valore dell'espressione ed in ordine rispetto ad altri eventuali blocchi comandi nel testo.

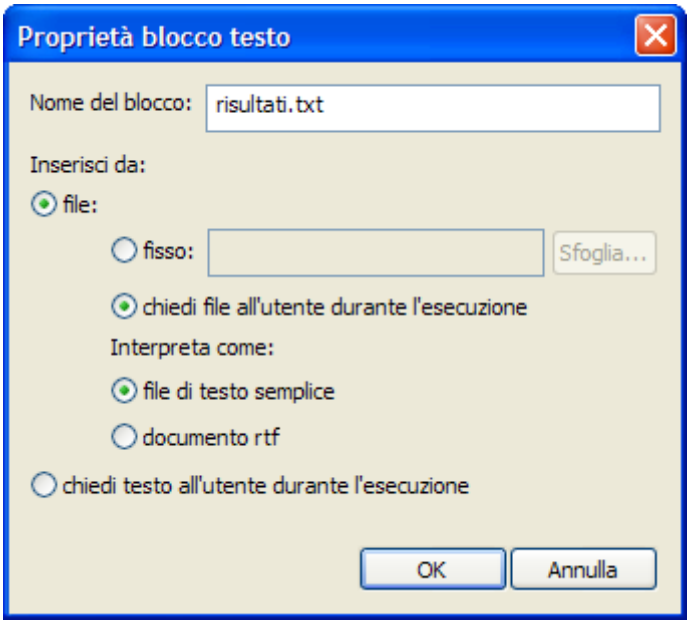
Il valore dell'espressione va a sostituire proprio il blocco comandi nel testo finale e assume lo stile carattere e paragrafo circostanti al blocco a meno che il codice del blocco non li modifichi esplicitamente (ved. capitolo "Funzioni di output").

Se il valore dell'espressione specificata è numerico, esso viene stampato con il formato specificato in fondo al dialogo, ovver esponenziale, esteso o il migliore tra i due in base al numero di cifre decimali specificato. Il formato predefinito (modificabile da script) è il migliore con sei cifre decimali.

Blocco di testo

Il blocco funzionale di testo serve per inserire nel documento di uscita del testo che provenga da un file di testo formattato c semplice oppure specificato direttamente dall'utente in fase di generazione.

Il dialogo di proprietà si presenta come in figura:

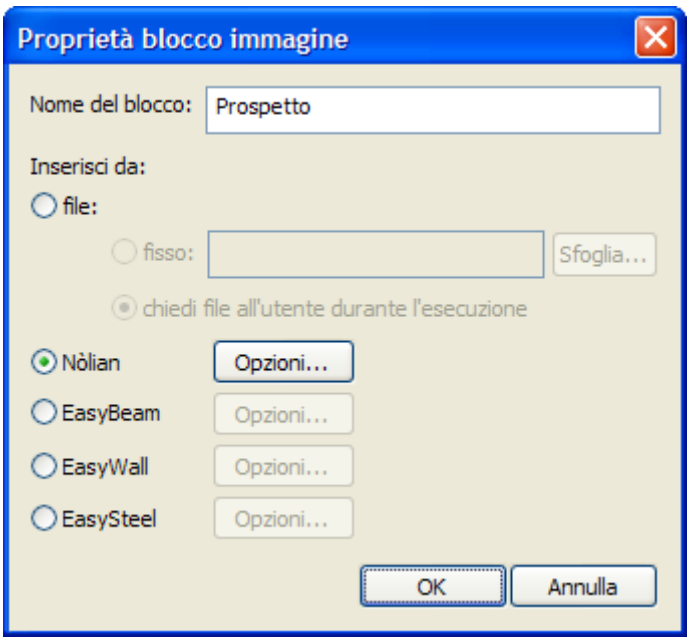


Oltre il nome del blocco si può specificare se il testo va letto da file o chiesto direttamente all'utente. Se il testo viene letto d file, si può indicare direttamente un percorso per il file oppure fare in modo che esso venga chiesto all'utente durante l'esecuzione; va inoltre indicato il formato in cui leggere il file: solo testo oppure come documento RTF.

Blocco immagine

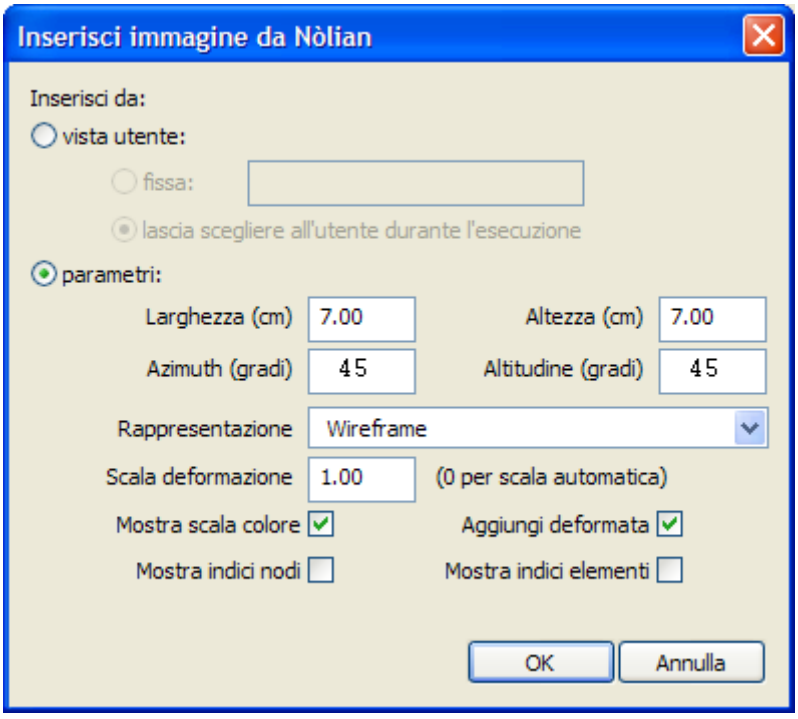
Il blocco funzionale immagine serve per inserire nel documento di uscita un'immagine. L'immagine può essere letta da un fil specificato nel dialogo di proprietà oppure chiesto all'utente durante l'esecuzione o generata da uno dei software della suite EasyWorld.

Il dialogo di proprietà si presenta come in figura:



Selezionando l'applicazione è possibile aprire un pannello per impostare i parametri di generazione dell'immagine in quella specifica applicazione. I parametri impostabili nei pannelli corrispondono ai parametri delle funzioni **append_param_view** e **append_user_view** descritte nelle librerie di funzioni di scripting più avanti.

L'aspetto generale dei pannelli di configurazione è come quello del seguente, relativo a Nòlian:



Nella prima parte è possibile specificare il nome di una vista utente memorizzata nel file corrente di Nòlian oppure lasciarla indicare all'utente al momento della generazione. Nella seconda parte è possibile costruire la vista da inserire a partire dai parametri descritti più avanti per la funzione **nolian.append_param_view**.

Generazione del documento di uscita

Una volta definita opportunamente la struttura del documento e le proprietà dei singoli blocchi, può essere avviata la generazione del documento di uscita: è sufficiente premere il tasto F5 oppure scegliere la voce Esegui del menu File.

Durante l'esecuzione i blocchi della struttura vengono elaborati ordinatamente uno per uno dall'alto verso il basso, secondo le seguenti semplici regole:

- se un blocco ha dei figli, essi vengono elaborati subito dopo il padre
- se un blocco di testo contiene dei blocchi funzionali, essi vengono elaborati nell'ordine in cui appaiono nel testo; quando tutti i blocchi funzionali sono stati elaborati, il testo finale viene aggiunto al documento di uscita
- se un blocco comandi ha una fase "post" essa viene elaborata subito dopo aver elaborato i figli e non prima
- se c'è un blocco di controllo l'esecuzione dei figli dipende dalle condizioni descritte nel capitolo "Blocco di controllo"

Il documento di uscita viene creato temporaneamente in un documento distinto da quello di EasyQuill, che permette la modifica del testo, il salvataggio come documento RTF o di testo non formattato e la stampa. Per una descrizione più approfondita dell'editor si rimanda ai capitoli "Panoramica sugli elementi di interfaccia" e "L'editor di testo".

Una istanza di EasyQuill può avere contemporaneamente aperto un solo documento EasyQuill e molti documenti RTF. Le finestre sono accessibili dal menu Finestra nel tipico stile delle applicazioni Windows.

Per la stampa dall'interno di EasyQuill è possibile specificare opzioni particolari per la generazione di un sommario e delle intestazioni e dei piè di pagina. Tali opzioni sono descritte in dettaglio nei successivi paragrafi.

Generazione del sommario

Una volta generato un documento con EasyQuill è possibile impostare delle opzioni per ottenere in stampa o in salvataggio anche un sommario dei contenuti del documento. Per accedere al dialogo per impostare tali opzioni è sufficiente accedere alla voce "Opzioni generazione sommario" del menu Formato dal documento generato:

Nella parte destra è visibile una anteprima del sommario aggiornata ad ogni variazione delle opzioni: l'anteprima tuttavia no dipende dal documento corrente ma da un documento fittizio.

Se il controllo "Aggiungi sommario in stampa" è attivo, il sommario verrà aggiunto in anteprima di stampa e in stampa. Se invece si desidera generare un sommario "statico" che appaia effettivamente nel testo per essere modificato manualmente che risulti nel salvataggio del file, è possibile scegliere il bottone "Genera ora": va indicato come sommario "statico" perché in questo caso esso non sarà aggiornato dinamicamente a seguito di modifiche al documento (ad esempio non aggiornerà titoli e numeri di pagina).

La generazione del sommario segue un certo modello, da selezionare tra quelli disponibili nel controllo a tendina, a partire d un certo stile di carattere impostabile.

Al sommario viene anteposta un'intestazione costruita da un titolo ("Sommario" ad esempio), da uno stile di carattere ed uno stile di paragrafo. Lo stile di paragrafo può essere ad esempio utilizzato per stabilire quanto spazio lasciare tra l'intestazione ed il sommario vero e proprio oppure per anteporre un'interruzione di pagina al sommario.

In fondo al dialogo appare il controllo per il numero di pagina in cui inserire il sommario (in stampa o con la funzione "Gener ora"): può essere specificato un numero di pagina preciso (ad esempio alla prima pagina o dopo un'introduzione o una copertina) oppure può essere indicato di inserire il sommario in fondo al documento.

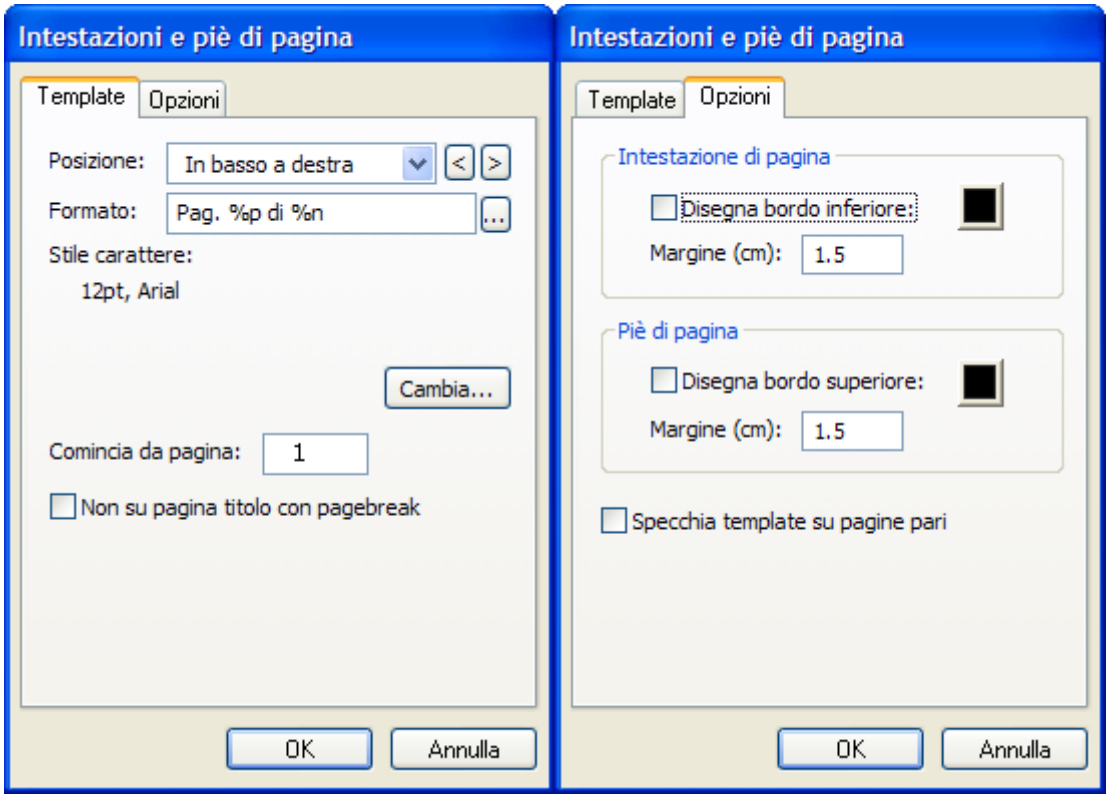
Intestazione e piè di pagina

In stampa le intestazioni e i piè di pagina vengono generate in base a dei modelli che sono impostabili sia dal documento generato e sia dal documento EasyQuill di partenza attraverso il dialogo accessibile dalla voce "Formato pagine generate" (dal documento EasyQuill) o "Intestazioni e piè di pagina" (dal documento generato) del menu Formato.

La differenza tra i due modi di accedere alle opzioni è che quello nel documento EasyQuill permette la modifica di opzioni

memorizzate col documento stesso, ripristinate nel caricamento e assegnate al documento di uscita ad ogni generazione; viceversa le opzioni del documento generato si perdono alla chiusura del documento e non possono essere memorizzate.

Il dialogo di opzioni si presenta con due schede mostrate in figura:



Nella prima scheda è possibile modificare il modello per la generazione delle intestazioni e dei piè di pagina selezionando da primo controllo a tendina la posizione e modificando il formato e lo stile del carattere da utilizzare. Inoltre è possibile far iniziare la stampa di un'intestazione da una certa pagina (per evitare ad esempio la stampa in una pagina di copertina o nelle pagine introduttive) oppure impedire la stampa su pagine che iniziano con un titolo (ad esempio evitare le intestazioni sulla prima pagina di un capitolo).

Nella seconda scheda è possibile abilitare la stampa di un bordo colorato per l'intestazione o per il piè di pagina, impostare i margini rispetto ai bordi del foglio ed impostare l'opzione per specchiare i modelli nelle pagine pari (i modelli a destra vengono usati a sinistra e viceversa).

Il formato del modello è una stringa che può contenere dei gruppi di caratteri ("token") da sostituire nel momento della generazione. Alcuni formati di uso comune sono disponibili nel menu a tendina che appare cliccando col mouse sul bottone coi tre punti accanto al campo di testo del formato. I token sono sempre introdotti da un simbolo di percentuale '%'; quelli riconosciuti sono:

%p Sostituito con il numero di pagina corrente.

%n Sostituito con il numero totale di pagine del documento.

%c Sostituito con il titolo dell'ultimo capitolo incontrato (titolo di primo livello).

%g Sostituito con l'ultimo titolo incontrato (di qualunque livello).

Consigliamo di provare i formati disponibili nel menu contestuale per prendere confidenza con i token disponibili.

Lo scripting

Come accennato nei capitoli introduttivi, il motore di scripting di EasyQuill si basa sullo standard industriale Lua.

Una presentazione completa del sistema di scripting Lua e delle caratteristiche avanzate del linguaggio esula dagli scopi di questo manuale. Nei prossimi capitoli esso si concentrerà invece su una semplice introduzione della sintassi e delle funzioni di base offerte dal linguaggio e da una dettagliata descrizione delle funzioni specifiche di EasyQuill.

Gli argomenti trattati qui trattati dovrebbero essere senz'altro sufficienti all'uso del linguaggio di scripting in EasyQuill per i comuni scopi per cui esso è stato progettato, ovvero per una generazione efficace e versatile di relazioni di calcolo e di documenti di testo più in generale.

Per ogni approfondimento riguardante caratteristiche avanzate del linguaggio di Lua si rimanda al sito web in lingua inglese della documentazione del prodotto:

http://www.lua.org/docs.html

Per ogni dubbio sul linguaggio e sull'uso dello scripting, la nostra assistenza tecnica rimane come di consueto a disposizione dei nostri clienti.

Sintassi

Vengono descritte nelle seguenti sezioni le regole fondamentali di sintassi che uno script Lua deve rispettare.

Vengono qui tralasciati gli aspetti del linguaggio più avanzati che si considerano come non strettamente necessari per la generazione di relazioni e di documenti di uso generale. L'utente interessato può approfondire la conoscenza del linguaggio sulla documentazione fornita dal sito internet:

http://www.lua.org/docs.html

Convenzioni lessicali

Uno script è formato da una o più linee di codice. Ogni linea di codice può contenere una o più espressioni formate da alcune particelle elementari: identificatori, valori, operatori e parole chiave.

Gli identificatori sono stringhe formate da lettere, cifre e sottolineature '_' che non iniziano con una cifra. Ad esempio `x`, `z0`, `_y` e `Progettista` sono tutti identificatori validi. Un identificatore può essere usato per riferirsi ad una variabile, ad una funzione o al campo di una tabella.

Nel linguaggio esistono alcune parole chiave che non possono essere utilizzate come identificatori:

```
and       break     do        else      elseif
end       false     for       function  if
in        local     nil       not       or
repeat    return    then      true      until     while
```

Il linguaggio è case-sensitive: le lettere maiuscole sono distinte dalle minuscole; perciò le stringhe x e X sono riconosciute come due identificatori differenti.

Valori di tipo stringa possono essere definiti tra una coppia di apici singole o doppie. All'interno di una stringa possono esser usate delle sequenze di controllo per caratteri speciali, introdotte dal carattere backslash '\', tra cui: '`\n`' per un accapo, '`\t`' per una tablatura, '`\\`' per il backslash, '`\'`' per il singolo apice e '`\"`' per il doppio apice. Un carattere può essere anche specificato con la sequenza '`\ddd`', dove ddd è il valore ASCII del carattere espresso come numero con una, due o tre cifre: ad esempio '`\128`' rappresenta il simbolo dell'euro '€'. Una stringa letterale può contenere valori qualsiasi, compresi valori nulli rappresentati dalla sequenza '`\0`'.

Un altro modo per specificare una stringa letterale è quello di usare la forma estesa con parentesi quadre: la stringa va racchiusa tra due coppie di parentesi quadre; tra le parentesi possono essere inseriti uno o più segni di uguale, l'importante che ce ne sia lo stesso numero tra le parentesi aperte e tra quelle chiuse; il numero di uguali viene chiamato livello della forma estesa. Ad esempio la stringa '`[[frase di esempio]]`' ha livello 0, mentre la stringa '`[===[frase di esempio]===]`' ha livello 3. Una stringa espressa in forma estesa può contenere qualunque carattere, non interpreta le sequenze di controllo e può estendersi per più linee: viene compreso tutto quello che viene trovato tra due parentesi dello stesso livello. Se subito dopo la parentesi aperta c'è un accapo, esso viene per comodità ignorato, in modo che la stringa possa cominciare direttamente su una riga vuota.

Una costante numerica può essere espressa con una parte decimale dopo un punto, con un esponente dopo una 'e' maiuscola o minuscola oppure in forma esadecimale quando introdotta da '0x'. Ad esempio le seguenti sono tutte costanti numeriche valide:

```
3      3.0    3.14   314.16e-2     0x2a
```

Un commento comincia con un doppio trattino '`--`' fuori da una stringa letterale e si estende fino alla fine della linea corrente. Se subito dopo i trattini c'è una parentesi quadra lunga aperta, il commento si estende fino alla successiva parentesi quadra lunga chiusa dello stesso livello.

Valori e tipi

Le variabili in Lua sono tipizzate dinamicamente. Questo vuol dire che le variabili non hanno un tipo fisso ma assumono quello dei valori che di volta in volta vengono loro assegnati.

I tipi più semplici per un valore sono: *nullo*, *booleano*, *numero*, *stringa*, *funzione* e *tabella.*

Il tipo nullo è quello della costante **nil** e serve in genere proprio per indicare che una variabile non ha alcun valore valido assegnato.

Il tipo booleano è quello dei valori **true** (vero) e **false** (falso). Qualunque valore che non sia false o nil è considerato come tru quando interpretato come booleano (compresi il valore numerico 0 e la stringa vuota).

Un numero è sempre inteso reale, ovvero rappresentato internamente con una notazione in virgola mobile a doppia precisione.

Una stringa è una qualunque sequenza di caratteri, ovvero valori numerici interi da 0 a 255 che hanno una corrispondenza nella mappa caratteri ASCII.

Una tabella è una mappa associativa tra due valori non nulli: un indice e un valore ad esso associato. Il tipo tabella può esser utilizzato per rappresentare i comuni array, mappe, insiemi, record, grafi, alberi, ecc.. Le tabelle sono descritte in maggiore dettaglio nel capitolo a loro dedicato.

La conversione tra tipi è in molti casi automatica: ad esempio i numeri vengono convertiti in stringhe al momento di una stampa oppure se si inserisce una stringa in una operazione aritmetica essa viene interpretata come un numero (ovvero viene letto il numero all'inizio della stringa se ce n'è uno, altrimenti viene utilizzato 0). Esistono poi funzioni esplicite di

conversione tra tipi come `string.format`, `tonumber` e`as_int`, per le quali si rimanda ai capitoli di riferimento delle funzioni più avanti nel manuale.

Variabili

Le variabili sono posizioni in memoria che mantengono un valore. Ci sono tre tipi di variabile: variabile locale, variabile globale e campo di tabella.

Tutte le variabili sono globali a meno che non siano definite esplicitamente come locali con la parola chiave `local`. I parametri di una funzione sono sempre variabili locali. La distinzione tra variabili locali e variabili globali è che mentre i valori delle seconde possono essere modificati e letti in qualunque punto dello script, in qualunque blocco sia di struttura che funzionale, l'accesso alle prime è possibile solo nel contesto lessicale in cui sono definite: ad esempio l'accesso all'argomento di una funzione ha senso solo all'interno della funzione stessa; oppure una variabile definita come locale all'interno di un blocco di struttura, può essere letta o modificata solo all'interno di quel blocco.

Se una variabile non è assegnata e si tenta di leggerne il valore, esso sarà pari a **nil**.

Il campo di una tabella può essere letto o modificato attraverso il suo indice tra parentesi quadre: ad esempio se abbiamo la tabella *table* un suo campo potrebbe essere indicizzato da `table[3]`, `table["nome"]` e così via. Nel caso di indici di tipo stringa un'alternativa è quella di usare la notazione col punto: nell'esempio precedente lo stesso campo può essere indicizzato da `table.nome`.

Istruzioni

Un'istruzione elementare di script è rappresentata da un'assegnazione, una struttura di controllo, una chiamata a funzione oppure una dichiarazione di variabile.

Un blocco di istruzioni è formato da singole istruzioni separate da accapo o punti e virgola ';'. Un blocco viene considerato lessicalmente come una singola istruzione quando delimitato dalle parole chiave `do` / `end`. Delimitare un blocco risulta utile per il controllo delle variabili locali e per uscire con un **break** o un **return** da un altro blocco.

Un'assegnazione è un'istruzione del tipo:

var_list = exp_list

Var_list è un elenco di variabili separate da virgole (ad esempio "`x, y, z`") ed *exp_list* è un elenco di espressioni separate da virgole (ad esempio "`0, "Nome", false`"). Nel caso più semplice in cui c'è una sola variabile ed una sola espressione, l'assegnazione assume la familiare forma:

```
variabile = valore
```

Nel caso di più espressioni e più variabili, si ha una forma del tipo:

```
a, b, c = exp1, f2(), 1
```

Prima dell'assegnazione tutte le espressioni vengono valutate e i valori vengono assegnati alle variabili. Se il numero di variabili è superiore a quello dei valori, i valori vengono assegnati alle prime variabili mentre le ultime restanti vengono impostate a **nil**. Se il numero di variabili è inferiore a quello dei valori, i valori in eccesso vengono scartati.

Se la lista di espressioni termina con una chiamata a funzione, tutti i valori restituiti dalla funzione vengono a far parte della lista di valori da assegnare. Ad esempio se la funzione restituisce due valori, vengono aggiunti due valori alla lista di quelli da

assegnare:

```
a, b, c = 1, f()
```

Se `f()` restituisce due valori, sarà assegnato 1 ad `a e i due valori di f()` rispettivamente a `b` e `c`. Se `f()` restituisce solo un valore, esso viene assegnato a `b`, mentre a `c` viene assegnato il valore **nil**.

Una dichiarazione di variabile locale consiste in un'istruzione del tipo:

local var_list [= exp_list]

Var_list è l'elenco delle variabili da dichiarare come locali e l'assegnazione dei valori iniziali da *exp_list* è opzionale. Nel caso in cui i valori iniziali non siano espressi, tutte le variabili assumono valore iniziale **nil**. Le variabili locali definite in un blocco di istruzioni sono accessibili solo in quel blocco e nei blocchi in esso eventualmente contenuti.

Strutture di controllo

Le strutture di controllo permettono l'esecuzione condizionale di un blocco di istruzioni.

Il controllo **while** assume la forma:

```
while condizione do blocco end
```

Le istruzioni di *blocco*vengono eseguite finché *condizione*è vera. Una condizione è una espressione che si considera vera quando non ha valore **false** o **nil** (il numero 0 e la stringa vuota contano come vere).

Il controllo **repeat** assume la forma:

```
repeat blocco until condizione
```

Le istruzioni di *blocco* vengono eseguite finché *condizione* non diventa vera. La condizione in questo caso viene controllata nel contesto lessicale del blocco, quindi può essere controllata anche su variabili locali definite nel blocco. Notare che nel **repeat** la condizione viene controllata *dopo* aver eseguito il blocco almeno una volta, al contrario del **while** dove invece vien controllata *prima* di ogni esecuzione.

Il controllo **if** assume la forma:

```
if condizione1 then blocco1
{elseif condizione2 then blocco2}
[else blocco3]
end
```

Il blocco `elseif` è opzionale e può essercene più di uno. Il blocco `else` è opzionale e può essercene solo uno. Le istruzioni c *blocco1* vengono eseguite solo se *condizione1* è vera; se non lo è, vengono controllate in sequenza le condizioni dei blocchi `elseif` ed eseguite le istruzioni del primo blocco la cui condizione è vera; infine, se nessuna condizione precedente è risultata vera, vengono eseguite le istruzioni di *blocco3*.

Il ciclo **for** ha due forme, una numerica ed una generale. La forma generale ha usi particolari che esulano dagli scopi di quest trattazione e che verrà qui utilizzata solo nel riferimento della funzione *pairs* / *ipairs*. Il modello della forma numerica è:

```
for v = e1, e2, e3 do blocco end
```

e1, e2, e3 sono espressioni numeriche che vengono valutate all'inizio del ciclo e sono rispettivamente il valore iniziale, il valore finale e l'incremento ad ogni passo che la variabile *v* assume. L'incremento è opzionale e ha valore supposto 1. La variabile *v* è locale al ciclo e non può essere letta o modificata al termine dello stesso: se il suo valore è utile al di fuori del ciclo, è necessario copiarlo in una variabile globale ausiliaria.

L'esecuzione ciclica delle istruzioni nei blocchi **while**, **repeat** e **for** può essere interrotta dall'istruzione **do break end, che porta il controllo fuori dal ciclo più interno dove si trova.**

Espressioni

Un'espressione è un componente elementare di un'istruzione: ogni istruzione è formata da una o più espressioni concatenate. Un'espressione può essere ad esempio rappresentata da un valore (un numero, una stringa, **nil**, vero o falso, ecc.), una chiamata a funzione, un'operazione binaria tra due espressioni o un'operazione unaria su una espressione.

Le operazioni binarie tra espressioni riconosciute sono:

le operazioni algebriche: **+** (somma), **-** (sottrazione), ***** (moltiplicazione), **/** (divisione), **%** (modulo), **^** (elevazione a potenza);

le operazioni relazionali, che restituiscono sempre un valore booleano **true** o **false**: **==** (uguali in tipo e valore), **~=** (diversi), **<** (minore), **>** (maggiore), **<=** (minore o uguale), **>=** (maggiore o uguale);

le operazioni logiche: **and** (restituisce il valore a sinistra se è **false** o **nil**, quello a destra altrimenti) e **or** (restituisce il valore a sinistra se non è né **false** né **nil**, quello a destra altrimenti);

l'operatore di concatenazione tra stringhe **..** che restituisce la stringa sinistra seguita da quella a destra: **"x".."y"** restituisce **"xy"**.

Gli operatori unari comprendono il meno unario **-**, che cambia segno ad un numero; il **not** che restituisce **true** se il valore a cui è applicato è **false** o **nil** e restituisce **false** in tutti gli altri casi; l'operatore **#** che restituisce la lunghezza di una stringa o di un array (nel caso dell'array restituisce l'indice precedente al primo valore **nil** trovato a partire da 1).

La precedenza tra operatori è, nell'ordine dalla minore alla maggiore: `or`, `and`, `<`, `>`, `<=`, `>=`, `~=`, `==`, `..`, `+`, `-`, `*`, `/`, `%`, `not`, `#`, `-` (unario), `^`. Gli operatori binari sono tutti associativi a sinistra tranne la concatenazione e l'elevazione a potenza che sono associativi a destra.

Tabelle

Una tabella è una mappa associativa formata da coppie di due valori non nulli: un indice e un valore. Ogni coppia prende il nome di campo. Il tipo tabella può essere utilizzato per rappresentare i comuni array, mappe, insiemi, record, grafi, alberi, ecc..

Il campo di una tabella può essere letto o modificato attraverso il suo indice tra parentesi quadre: ad esempio se abbiamo la tabella *table* un suo campo potrebbe essere indicizzato da `table[3]`, `table["nome"]` e così via. Nel caso di indici di tipo stringa un'alternativa è quella di usare la notazione col punto: nell'esempio precedente lo stesso campo può essere indicizzato da `table.nome`.

Le tabelle possono essere eterogenee: il valore di ogni campo può avere un tipo differente. Il tipo per indice e valore può essere uno qualsiasi tra quelli disponibili tranne il tipo nullo. Ad esempio un campo può avere come valore una funzione per realizzare un sistema di programmazione ad oggetti.

È possibile costruire una tabella *table* vuota con l'istruzione:

```
table = {}
```

Se si vuole costruire una tabella con tre valori di indice da 1 a 3 invece:

```
table = { "val1", "val2", "val3" }
```

Se invece degli indici numerici automatici se ne vogliono specificare di espliciti:

```
table = { [5] = "val1", x = "val2",
["y"] = "val3", [f(1.2)] = "val4" }
```

In questo caso il primo campo ha indice 5: vi si accede con *table[5]*. Il secondo è associato all'indice "x" ed il terzo all'indice "y" e sono modalità equivalenti per utilizzare una stringa come indice: entrambi possono essere acceduti attraverso la notazione *table["x"]* oppure dalla notazione *table.x*. Infine l'ultimo campo ha come indice il risultato della chiamata a funzione *f(1.2)*, qualunque esso sia (viene segnalato errore se il valore è nullo).

Le notazioni si possono infine mischiare:

```
table = { "val1", x = "val2", "val3", [f(1.2)] = "val4" }
```

In quest'ultimo caso al primo e al terzo campo vengono assegnati gli indici numerici consecutivi liberi, ovvero 1 al primo e 2 a secondo.

Nella costruzione di una tabella la virgola tra i campi può essere indifferentemente sostituita con il punto e virgola. Se l'elenco dei campi termina con una chiamata a funzione, vengono aggiunti alla tabella tutti i risultati della funzione.

I campi possono essere aggiunti ad una tabella in ogni momento con una normale assegnazione, ad esempio la seguente espressione aggiunge o sostituisce a *table* un campo con indice "x" e valore 2:

```
table["x"] = 2
```

Si deve fare attenzione nella specifica degli indici di una tabella attraverso variabili: il tipo di variabile utilizzato è importante. Se ad esempio si scrive `table[x]`, al cambiare del tipo della variabile *x* cambia anche l'elemento della tabella a cui si sta facendo riferimento: in particolare x=5 (valore intero) e x="5" (stringa) sono indici *diversi*. Se la variabile viene ad esempio dall'input di un utente (attraverso una finestra di dialogo) e ci si aspetta un numero, è sempre bene forzare la conversione con funzioni come *tonumber()*.

Funzioni

Un blocco di istruzioni può essere contrassegnato come una funzione da poter chiamare in ogni momento dello script con ur identificatore opportuno. EasyQuill rende già disponibili decine di funzioni che sono descritte nel seguito. L'utente può definire facilmente le proprie.

Una funzione generalmente prende dei parametri in input, li elabora e restituisce in output un risultato. La forma elementar per definire una funzione è:

```
function nome ( [argomenti] ) blocco end
```

Gli argomenti sono opzionali e sono una lista di variabili locali alla funzione che assumono i valori passati come parametri nella chiamata alla funzione stessa. Anteponendo la parola chiave `local` alla definizione si specifica che la funzione ha solo valore locale al blocco corrente. Per restituire dei risultati, il blocco di istruzioni della funzione deve terminare con un'istruzione `return` *`exp_list`* dove exp_list è una lista di espressioni separata da virgole.

Una semplice definizione di funzione può essere ad esempio:

```
function media (a, b)
    return (a + b) / 2
end
```

Per chiamare una funzione è sufficiente usare l'identificatore seguito dagli eventuali argomenti tra parentesi. Ad esempio la seguente istruzione assegna alla variabile globale c il valore numerico 3.5:

```
c = media(3, 4)
```

Una funzione può avere un numero variabile di argomenti se la lista di argomenti nella dichiarazione termina con tre punti (`'...'`). Ad esempio la funzione **string.format** è dichiarata come :

```
function string.format ( f, ... )
```

Per accedere ai parametri corrispondenti alla lista di argomenti variabili, si può utilizzare la stessa espressione `'...'` oppure la variabile `arg` che è una tabella contenente tutti gli argomenti non espliciti più un campo `n` che indica il numero di parametri nella lista.

Ad esempio considerando la funzione string.format:

- con una chiamata del tipo `string.format("a")`, il parametro f avrà valore "a" e arg sarà la tabella `{ n=0 }`
- con la chiamata `string.format("a%d", 2)`, il parametro f avrà valore "a%d" e arg sarà la tabella `{ 2, n=1 }`
- con la chiamata `string.format("a%d%s", 2, "b")`, il parametro f avrà valore "a%d%s" e arg sarà la tabella `{ 2 "b", n=1 }`

Come specificato in precedenza, una funzione può restituire più di un risultato. Per accedere ai risultati successivi al primo è possibile assegnare i risultati ad un gruppo di variabili (ved. **Istruzioni**) o ad una tabella. Ad esempio, se *f()* è una funzione ch restituisce tre risultati, è possibile scrivere:

```
a, b, c = f()
```

In alternativa i risultati possono essere inseriti in una tabella:

```
t = { f() }
```

e acceduti attraverso le espressioni proprie delle tabella: il secondo valore restituito può essere letto come `t[2]`. Vedere anche la funzione **select(v, ...)**.

Funzioni di base

Il motore di scripting mette a disposizione alcune funzioni di utilità generale di seguito descritte.

as_int(v)

Prende in ingresso un numero qualsiasi e ne restituisce la sola parte intera come stringa, in modo che l'output non dipenda dal formato attualmente utilizzato per i numeri.

date()

Restituisce una tabella con i seguenti campi inizializzati:

hour, **min**, **sec**	ora, minuto e secondo attuali
day	numero del giorno del mese corrente tra 1 e 31
month	numero del mese dell'anno corrente tra 1 e 12
month_name	nome del mese corrente, tutto minuscolo: gennaio, febbraio...
year	anno corrente a quattro cifre, es. 2006
wday	numero del giorno della settimana corrente da 1 a 7 cominciando dalla Domenica
wday_name	nome del giorno della settimana corrente, tutto minuscolo: lunedì, martedì...
yday	numero del giorno dell'anno corrente da 1 a 365
isdst	valore **true** o **false** che indica se si è attualmente nell'ora legale o meno

exit()

Permette di interrompere prematuramente l'elaborazione del documento.

Se non viene specificato alcun argomento, l'uscita interrompe semplicemente l'elaborazione del documento: si indica che si è conclusa (con successo) la generazione del documento di uscita.

Se viene specificato un parametro di tipo stringa, esso viene interpretato come il testo dell'errore da presentare all'utente e la generazione del documento di uscita viene annullata.

tonumber(v)

Prende in ingresso una stringa e la interpreta per restituire un valore numerico reale. Se v non può essere convertito ad un numero, restituisce **nil**.

Può essere specificato un parametro aggiuntivo *base*, che indica la base su cui interpretare la stringa: *base* deve essere un numero tra 2 e 36. Se *base* non viene specificato, si intende pari a 10.

Per basi maggiori di 10 le lettere maiuscole o minuscole vengono interpretate come cifre essendo A = 10 e Z = 35.

Il numero può avere una parte decimale ed un esponente solo se viene specificata una base 10.

tostring(v)

Prende in ingresso un valore qualsiasi e lo converte in una rappresentazione letterale opportuna. Nel caso di valori numerici in particolare si consiglia di utilizzare **string.format(v)** per avere maggior controllo sui parametri di conversione.

type(v)

Prende in ingresso un valore qualsiasi e restituisce il nome del tipo del valore.

Possibili valori restituiti sono "NIL", "NUMBER", "STRING", "BOOLEAN", "TABLE" e "FUNCTION".

assert (v)

Se il parametro *v* ha valore **false** o **nil**, termina l'esecuzione con un errore; altrimenti restituisce *v*.

Un secondo parametro opzionale può indicare il messaggio di errore da mostrare.

Questa funzione può essere utile per controllare che una chiamata a funzione abbia avuto successo. Ad esempio la funzione **dlg.ask_yesno** (ved.) restituisce **false** quando l'utente risponde no ad una domanda. La funzione **assert** può essere utilizzata per interrompere subito l'esecuzione:

```
      assert(dlg.ask_yesno("Si desidera continuare?"), "L'utente ha interrotto
l'esecuzione")
```

Questa riga di codice mostrerà all'utente un dialogo che gli chiede se continuare. Se l'utente risponde no, l'esecuzione si interrompe con un messaggio d'errore opportuno.

Si noti che se l'esecuzione termina con un messaggio di errore, la generazione del documento di uscita viene annullata. Per evitare di perdere il contenuto generato fino a quel momento, si può utilizzare la funzione **exit** (ved.).

select (v, ...)

Se *v* è un numero, restituisce la lista degli altri argomenti dal *v*-esimo in poi. Altrimenti *v* può essere la stringa "#" e la funzione restituisce il numero degli altri argomenti della funzione.

Questa funzione può essere utile per scartare un certo numero di valori restituiti da una funzione o per saperne il numero. Ad esempio la funzione **math.modf** (ved.) restituisce la parte intera e la parte frazionaria di un numero. Se ci interessa solo l

parte frazionaria possiamo scrivere:

```
f = select(2, math.modf(a))
```

Funzioni matematiche

Il motore di scripting offre diverse funzioni matematiche di supporto, oltre a quelle accessibili attraverso i comuni operatori algebrici discussi nel capitolo "Espressioni".

Tutti i valori numerici vengono rappresentati come reali in virgola mobile a doppia precisione. Il valore più grande rappresentabile è circa 10^{308}, quello più piccolo circa 10^{-324}. La precisione massima è su circa 15-16 cifre decimali.

Per il supporto alle funzioni matematiche sono definite alcune costanti:

math.huge: un valore maggiore di qualunque altro valore numerico;

math.pi: il valore di pi greco.

math.abs(v)

Restituisce il valore assoluto di v.

math.acos(v)

Restituisce l'arcocoseno di v in radianti.

math.asin(v)

Restituisce l'arcoseno di v in radianti.

math.atan(v)

Restituisce l'arcotangente di v in radianti.

math.atan2(x, y)

Restituisce l'arcotangente di (x / y) in radianti e usa il segno di entrambi i parametri per scegliere correttamente il quadrante del risultato, gestendo anche correttamente i casi per y = 0.

math.ceil(v)

Restituisce il più piccolo intero maggiore o uguale a v.

math.cos(v)

Restituisce il coseno di v, interpretato come angolo in radianti.

math.cosh(v)

Restituisce il coseno iperbolico di v.

math.deg(v)

Restituisce il corrispondente angolo in gradi di v, interpretato come angolo in radianti.

math.exp(v)

Restituisce il valore *ev*.

math.floor(v)

Restituisce il più grande intero minore o uguale a v.

math.fmod(x, y)

Restituisce il resto della divisione (x / y).

math.frexp(v)

Restituisce una coppia di valori *m* ed *e* tali che *v = m*2e*, con *e* intero ed il valore assoluto di *m compreso nell'intervallo [0.5, 1) o zero se v = 0.*

math.ldexp(m, e)

Restituisce il valore *m*2e* (*e dovrebbe essere un numero intero).*

math.log(v)

Restituisce il logaritmo naturale di v.

math.log10(v)

Restituisce il logaritmo in base 10 di v.

math.max(v, ...)

Restituisce il valore massimo tra tutti i suoi argomenti. Il numero di argomenti deve essere almeno uno.

math.min(v, ...)

Restituisce il valore minimo tra tutti i suoi argomenti. Il numero di argomenti deve essere almeno uno.

math.modf(v)

Restituisce due valori: la parte intera di v e la parte frazionaria di v.

math.pow(x, y)

Restituisce il valore *xy* (assolutamente equivalente all'espressione **x^y**).

math.rad(v)

Restituisce il corrispondente angolo in radianti di v, interpretato come angolo in gradi.

math.random( )

Restituisce un valore pseudo-casuale compreso nell'intervallo [0, 1).

Se alla funzione viene passato un parametro aggiuntivo *m*, il valore restituito è compreso nell'intervallo [1, m]. Se vengono passati due parametri *m* ed *n*, il valore restituito è compreso nell'intervallo [m, n].

math.randomseed(v)

Imposta a v il seme per la generazione pseudo-casuale dei numeri: per un certo seme, la sequenza è sempre la stessa.

math.sin(v)

Restituisce il seno di v, interpretato come angolo in radianti.

math.sinh(v)

Restituisce il seno iperbolico di v.

math.sqrt(v)

Restituisce la radice quadrata di v.

math.tan(v)

Restituisce la tangente di v, interpretato come angolo in radianti.

math.tanh(v)

Restituisce la tangente iperbolica di v.

Funzioni su stringa

Il motore di scripting permette una gestione molto potente e versatile del testo, nella forma di valori stringa che possono essere manipolati attraverso le funzioni qui di seguito elencate oltre che attraverso l'operatore di concatenazione (. .) discusso nel capitolo "Espressioni".

Una stringa è una sequenza di caratteri qualsiasi di lunghezza definita. Ogni carattere è un numero da 0 a 255 che ha una corrispondenza nella mappa di caratteri del sistema (ASCII). I caratteri che formano una stringa di lunghezza *n* sono indicizzabili: il primo carattere ha indice 1, l'ultimo ha indice *n*. Possono essere utilizzati anche indici negativi per indicare una posizione a partire dalla fine della stringa: -1 indica l'ultimo carattere della stringa.

string.byte(s)

Restituisce il valore numerico del primo carattere della stringa s.

Se viene specificato un parametro aggiuntivo *i*, esso viene interpretato come l'indice del carattere della stringa di cui restituire il valore numerico. Es.: `string.byte( s, 3 )` restituisce il valore numerico del terzo carattere della stringa s.

Se vengono specificati due parametri aggiuntivi *i* e *j*, vengono restituiti i valori numerici di tutti i caratteri con indice compreso tra *i* e *j* inclusi. Es.: `string.byte( s, 2, 5 )` restituisce i valori numerici del secondo, terzo, quarto e quinto carattere della stringa s.

string.char(...)

Restituisce una stringa formata da tutti i valori numerici passati come parametri alla funzione.

Es.: `string.char( 10, 11, 12 )` restituisce una stringa di tre caratteri con valore numerico rispettivamente 10, 11 e 12.

string.find(s, p)

Restituisce gli indici di s in cui comincia e termina la prima occorrenza del pattern p. Se p non appare in s viene restituito **nil**.

Può essere specificato come terzo parametro l'indice di s da cui cominciare la ricerca. Il valore di default è 1.

Il parametro p viene considerato un pattern di ricerca a meno che non sia specificato **true** come quarto parametro opzionale in tal caso p viene interpretato come stringa letterale.

Se p è un pattern di ricerca con blocchi di cattura e viene trovata una occorrenza in s, i caratteri catturati vengono restituiti dopo gli indici di inizio e fine.

Vedere "Pattern di ricerca" per una descrizione dei pattern.

string.format(f, ...)

Restituisce una versione formattata dei suoi parametri secondo la stringa di formato f.

Il formato f è una stringa che può contenere dei codici che indicano il formato del relativo parametro della funzione. Tutti i codici cominciano col simbolo di percentuale %, seguito dal tipo di valore da inserire:

%c	Per un carattere espresso numericamente.
%d o %i	Per un numero intero con segno, %u per un numero intero senza segno; può essere specificato un numero minimo di caratteri da utilizzare, ad es. %8d utilizza almeno otto caratteri, anteponendo spazi se necessario; se il numero di caratteri è specificato dopo un meno (-), gli spazi vengono messi dopo se necessario; se prima del numero c'è uno zero (0), al numero vengono anteposti degli zeri anziché degli spazi.
%o	Per un numero intero da stampare in forma ottale.
%x o %X	Per un numero intero da stampare in forma esadecimale con lettere minuscole o maiuscole a seconda del caso della x.
%e o %E	Per un numero reale espresso in notazione scientifica: ad es. 1.3e-5 oppure 1.3E-5 a seconda che la e sia stata specificata minuscola o maiuscola; si può opzionalmente specificare il numero di cifre prima del punto (ad es. %3e indica tre cifre prima del punto), quelle dopo il punto (ad es. %.5e indica cinque cifre decimali dopo il punto), oppure entrambe: %3.5e indica un punto con tre cifre prima del punto e cinque cifre dopo.
%f	Per un numero reale espresso sempre in forma estesa, senza esponente; anche %f accetta lo stesso tipo di formato per il numero di cifre dopo il punto di %e, mentre la specifica di cifre prima del punto viene ignorata.
%g o %G	Per un numero reale da formattare in base ai numeri di cifre specificati (come per %e) ma scegliendo la rappresentazione più compatta tra quella scientifica e quella estesa.
%s	Per un valore di tipo stringa, %q per un valore di tipo stringa con caratteri di escape.

string.gsub(s, p, r)

Restituisce una copia della stringa s in cui tutte le occorrenze di p sono state sostituite da r.

Il parametro p è un pattern di ricerca. Ved. "Pattern di ricerca" per maggiori informazioni.

Il parametro r può essere una stringa, una tabella o una funzione.

Se r è una stringa, per ogni occorrenza di p essa viene interpretata sostituendo ai codici del tipo %*n* con n una cifra da 1 a 9 l
catture definite da p e sostituendo i codici %0 con l'intera occorrenza trovata in s; la stringa finale viene quindi sostituita all'occorrenza trovata in s.

Se r è una tabella, ogni occorrenza di p trovata in s viene sostituita con il valore della tabella indicizzato dalla prima cattura definita in p o dall'intera occorrenza se non ci sono catture definite.

Se r è una funzione, essa viene chiamata per ogni occorrenza con tutte le catture come argomenti oppure con l'intera occorrenza come unico argomento se non ci sono catture definite.

Se la ricerca nella tabella o la funzione restituiscono **false** o **nil**, non viene effettuata alcuna sostituzione nella stringa; altrimenti l'occorrenza corrente viene sostituita dal valore restituito (che dev'essere una stringa o un numero).

Può essere specificato un numero come quarto parametro opzionale per limitare il numero di sostituzioni da effettuare: ad esempio specificando 1 viene eseguita solo la prima sostituzione.

string.len(s)

Restituisce la lunghezza della stringa s.

string.lower(s)

Restituisce una copia della stringa s dove tutti i caratteri maiuscoli sono stati tradotti in caratteri minuscoli.

string.match(s)

Cerca il pattern p nella stringa s. Se il pattern viene trovato, vengono restituiti i caratteri catturati dai suoi blocchi di cattura oppure tutta la parte di s corrispondente al pattern se non ci sono blocchi di cattura definiti. Se p non viene trovato in s, viene restituito **nil**.

Può essere specificato un parametro aggiuntivo per indicare l'indice di s da cui iniziare la ricerca.

Vedere "Pattern di ricerca" per una descrizione dei pattern.

string.rep(s)

Restituisce una stringa formata da n ripetizioni di s.

string.reverse(s)

Restituisce una copia girata della stringa s.

string.sub(s)

Restituisce la parte di s che comincia alla posizione *i* e si estende fino alla fine di s.

Può essere specificato un parametro aggiuntivo *j* per indicare l'indice dell'ultimo carattere di s da restituire.

Gli indici *i* e *j possono essere anche negativi per indicare posizioni relative alla fine della stringa.*

Es.: `string.sub( s, 1, 5 )` restituisce una stringa formata dai primi cinque caratteri di s; `string.sub( s, -3 )` restituisce una stringa formata dagli ultimi tre caratteri di s.

string.upper(s)

Restituisce una copia della stringa s dove tutti i caratteri minuscoli sono stati tradotti in caratteri maiuscoli.

Pattern di ricerca

Un pattern di ricerca è un tipo di espressione regolare utilizzato per le funzioni di ricerca come **string.find** e **string.match**.

Un pattern è formato da una sequenza di elementi base che può cominciare con un accento circonflesso ^ per indicare che il pattern va cercato all'inizio di una stringa e può terminare con un simbolo di dollaro $ per indicare che il pattern va cercato

alla fine di una stringa.

Gli elementi base di un pattern possono essere:

- una singola classe di caratteri, che corrisponde ad un singolo carattere della classe stessa
- una classe di caratteri seguita dall'asterisco *, per indicare da zero ad infinite corrispondenze della classe, prendendo sempre il maggior numero possibile di caratteri
- una classe di caratteri seguita dal segno -, per indicare da zero ad infinite corrispondenze della classe, prendendo sempre il minor numero possibile di caratteri
- una classe di caratteri seguita dal segno +, per indicare da una ad infinite corrispondenze della classe, prendendo sempre il maggior numero possibile di caratteri
- una classe di caratteri seguita dal punto interrogativo ?, per indicare zero od una corrispondenza della classe
- la forma %b*xy*, dove *x* e *y* sono due caratteri distinti; questa forma corrisponde ad una stringa che comincia col carattere *x* e termina col carattere *y* bilanciati, ovvero se vengono trovate altre occorrenze di *x* prima della fine, vann trovate anche un numero corrispondente di *y* prima della fine; ad es. %b() cerca un'espressione tra parentesi bilanciate

Una classe di caratteri è un elemento rappresentato da:

x	un carattere qualsiasi non compreso tra i caratteri `^$()%.[]*+-?;` rappresenta il carattere *x* stesso
. (un punto)	un qualsiasi carattere
%w	rappresenta una qualsiasi lettera o cifra decimale
%a	rappresenta una lettera dell'alfabeto qualsiasi
%l (elle)	rappresenta una qualsiasi lettera minuscola
%u	rappresenta una qualsiasi lettera maiuscola
%d	rappresenta una qualsiasi cifra decimale da 0 a 9
%x	rappresenta una qualsiasi cifra esadecimale da 0 a 9 e da A ad F (maiuscoli o minuscoli)
%z	rappresenta un qualsiasi carattere con rappresentazione 0
%s	rappresenta un qualsiasi carattere di spazio (spazio, accapo, tablatura...)
%c	rappresenta un qualsiasi carattere di controllo (accapo, tablatura...)
%p	rappresenta un qualsiasi carattere di punteggiatura

%x	dove x è un carattere non alfanumerico: rappresenta il carattere *x* stesso; questa classe va utilizzata per i caratteri speciali e può essere utilizzata per i segni di punteggiatura
[*gruppo*]	rappresenta una classe unica composta dal gruppo di classi contenute tra tutte quelle sopra descritte; inoltre possono essere specificate classi di caratteri compresi tra due caratteri separati da un segno meno: es. [0-7] rappresenta la classe delle cifre dei numeri ottali

[^*gruppo*]: rappresenta la classe complementare rispetto a [*gruppo*].

Un pattern non può contenere il carattere zero (0): utilizzare la classe %z al suo posto.

Un pattern può definire dei blocchi di cattura. I blocchi di cattura definiscono delle sotto-stringhe che vengono isolate e restituite dalle funzioni di ricerca. Per definire un blocco di cattura è sufficiente racchiudere tra parentesi la parte del patterr che lo definisce. Se un blocco di cattura è vuoto (cioè "()"), viene catturato l'indice corrente nella stringa in cui cercare.

Funzioni su tabella

Le tabelle sono un tipo di dato molto importante per il motore di scripting poiché permettono l'implementazione di molte strutture dati di uso comune, dall'array alla mappa fino agli oggetti. Per questo motivo sono fornite alcune potenti funzioni d manipolazione delle tabelle, qui descritte.

Per una descrizione più dettagliata del tipo tabella, si rimanda al capitolo "Tabelle".

ipairs(t)

È una funzione che può essere utilizzata per analizzare tutti gli elementi di una tabella indicizzati da numeri consecutivi da 1 ad *n*, dove *n*+1 è il primo indice per cui la tabella restituisce **nil**. **ipairs(t)** va utilizzata all'interno di un ciclo for generico. La forma in cui va usata è:

```
for i, v in ipairs( t ) do istruzioni end
```

All'interno del corpo istruzioni si potrà accedere al valore v corrispondente all'indice i della tabella t. I nomi delle variabili sono ovviamente indicativi e possono essere modificati a piacere.

All'interno del corpo istruzioni non può essere creato alcun nuovo campo della tabella. Possono invece essere modificati o eliminati campi esistenti.

I campi della tabella vengono visitati in ordine crescente a partire da 1 fino all'ultimo elemento non nullo consecutivo.

pairs(t)

È una funzione che può essere utilizzata per analizzare ogni elemento di una tabella all'interno di un ciclo for generico. La forma in cui va usata è:

```
for k, v in pairs( t ) do istruzioni end
```

All'interno del corpo istruzioni si potrà accedere alla chiave k ed al valore v del campo corrente della tabella t. I nomi delle variabili sono ovviamente indicativi e possono essere modificati a piacere.

All'interno del corpo istruzioni non può essere creato alcun nuovo campo della tabella. Possono invece essere modificati o eliminati campi esistenti.

L'ordine in cui vengono processati i campi della tabella non è definito ma dipende dalla rappresentazione interna dei dati. Se è necessaria una visita ordinata si consiglia l'uso della funzione **ipairs(t)**.

table.concat(t)

Restituisce una stringa formata dalla successione di tutti i valori stringa o numerici indicizzati da 1 all'ultimo valore non nullo di indice consecutivo, ovvero quello che ha indice restituito dall'operatore #.

Come secondo parametro può essere specificata una stringa che viene interposta ai valori dell'array nella concatenazione. D norma tale stringa è vuota.

Un terzo parametro opzionale può indicare l'indice da cui iniziare la lettura dei valori nella tabella. Un quarto parametro può infine indicare l'indice dell'ultimo elemento da leggere nella tabella.

table.insert(t, v)

Inserisce nella tabella t un campo di valore v ed indice pari ad *n*+1, dove *n* è il valore restituito dall'operatore **#** applicato a t.

La forma **table.insert(t, pos, v)** inserisce nella tabella t un campo di valore v ed indice pos. Se la tabella ha valori con indice maggiore di pos, essi vengono preventivamente spostati verso l'alto di una posizione.

table.maxn(t)

Restituisce il più grande indice numerico positivo trovato nella tabella t o zero se non esistono indici positivi.

La differenza tra **table.maxn** e l'operatore **#** è che quest'ultimo restituisce l'indice precedente al primo che corrisponde ad u valore nullo. Se ad esempio t è definita come:

```
t = { 1="a", 3="b" }
```

allora `table.maxn(t)` restituisce 3, mentre `#t` restituisce 1.

table.remove(t)

Elimina il campo di indice *n* dalla tabella t, dove *n* è il valore restituito dall'operatore **#**, e restituisce il valore del campo eliminato.

È possibile specificare come parametro aggiuntivo l'indice numerico dell'elemento da eliminare. In questo caso se esistono campi di indice maggiore a quello specificato, dopo l'eliminazione essi vengono spostati verso il basso a coprire il buco.

table.sort(t)

Ordina in modo crescente gli elementi della tabella t da 1 ad *n*, dove *n* è il valore restituito dall'operatore **#**.

Come secondo parametro opzionale può essere specificata una funzione che prende in ingresso due valori e restituisce **true** se il primo è minore del secondo. Di norma la funzione utilizzata è l'operatore **<**.

L'algoritmo di ordinamento non è stabile: se due elementi sono considerati uguali, applicare più volte l'ordinamento potrebbe non mantenere il loro ordine relativo nella tabella uguale.

unpack(t)

Restituisce tutti i valori della tabella t che hanno indice da 1 all'ultimo valore non nullo di indice numerico consecutivo, ovvero quello restituito dall'operatore **#** (ved. capitolo "Tabelle").

È possibile specificare un indice come secondo parametro per indicare il primo campo di cui restituire il valore. È anche possibile specificare un altro indice come terzo parametro per indicare l'ultimo campo di cui restituire il valore. Ad es. `unpack( t, 3, 5 )` restituisce i valori dei campi di t indicizzati da 3, 4 e 5, come in:

```
a, b, c = unpack(t, 3, 5)
```

Funzioni ausiliarie

Vengono di seguito descritte funzioni di utilità più specifica rispetto alle funzioni di base ma non per questo meno generali.

calc_spectrum(t)

Prende in ingresso una tabella di parametri da interpretare per generare uno spettro di risposta. Viene restituita una tabella avente gli stessi parametri della tabella d'ingresso oltre ai valori calcolati dello spettro.

La tabella in ingresso può essere generata manualmente o dalla funzione **dlg.ask_spectrum**

La tabella in uscita può essere utilizzata ad esempio per la stampa del fattore di struttura Q o per il grafico dei valori attraverso la funzione **out.append_plot**

La tabella ha i seguenti campi (gli indici sono tutti valori stringa):

Campo	Tipo	Descrizione
intervalli	in	il numero di valori da calcolare per la funzione dello spettro
durata	in	la durata in secondi su cui calcolare la funzione
tipo_spettro	in	Un valore tra "Inelastico" o "Elastico".
azione_sismica	in	il tipo di componente dell'azione sismica: "Orizzontale" o "Verticale"
classe_dutt	in	un valore tra "Alta", "Bassa" o "Non dissipativa" che indica la classe di duttilità della struttura
terreno	in	la categoria del suolo, tra "A" e "E", in base alla seguente tabella:
		A formazioni litoidi o suoli omogenei molto rigidi

B depositi di sabbie o ghiaie molto addensate o argille consistenti

C depositi di sabbie e ghiaie mediamente addensate o di argille di media consistenza

D depositi di terreni granulari da sciolti a poco addensati oppure poco coesivi da poco a mediamente consistenti

E profili di terreno costituiti da strati superficiali alluvionali

desc_terreno	out	la descrizione della categoria del suolo corrispondente al tipo della voce precedente; ignorata in input
smorzamento	in	il valore di smorzamento da utilizzare per la funzione
fattore_kw	in	il fattore di riduzione del fattore di comportamento q per strutture con pareti in calcestruzzo armato
ampl_topogr	in	un valore tra "T1", "T2", "T3" o "T4" che indica l'amplificazione topografica
perc_quota_pendio	in	la percentuale di quota sul pendio
ag	in	indica il valore del parametro "Ag(/g)"
f0	in	indica il valore del parametro "f0"
Tc	in	indica il valore del parametro "Tc*"
materiale	in	il tipo di materiale tra "Calcestruzzo", "Acciaio", "Acciaio e calcestruzzo", "Legno" o "Muratura".
tipo_struttura	in	il codice della tipologia di struttura: in base alla seguente tabella per strutture in calcestruzzo:

1 strutture a telaio di un piano

2 strutture a telaio con più piani ed una sola campata

3 strutture a telaio con più piani e più campate

4 strutture con solo due pareti non accoppiate per direzione orizzontale

5 altre strutture a pareti non accoppiate

6 strutture a pareti accoppiate o miste equivalenti a pareti

7 strutture deformabili torsionalmente

8 strutture a pendolo inverso

9 strutture a pendolo inverso intelaiate monopiano

in base alla seguente tabella per strutture in acciaio o miste in acciaio e calcestruzzo:

1 strutture a telaio di un piano

2 strutture a telaio con più piani ed una sola campata

3 strutture a telaio con più piani e più campate

4 strutture con controventi eccentrici a un piano

5 strutture con controventi eccentrici a più piani

6 strutture con controventi concentrici a diagonale tesa attiva

7 strutture con controventi concentrici a V

8 strutture a mensola o a pendolo inverso

9 strutture a telaio con controventi concentrici a un piano

10 strutture a telaio con controventi concentrici a più piani ed una sola campata

11 strutture a telaio con controventi concentrici a più piani e più campate

12 strutture a telaio con tamponatura in murature

in base alla seguente tabella per strutture in legno:

1 pannelli a telaio chiodati con diaframmi incollati

2 strutture reticolari iperstatiche con giunti chiodati

3 portali iperstatici con mezzi di unione a gambo cilindrico

4 pannelli a telaio chiodati con diaframmi chiodati

5 pannelli di tavole incollate a strati incrociati

6 strutture reticolari

7 strutture miste con telaio in legno e tamponature non portanti

8 strutture isostatiche in genere

in base alla seguente tabella per strutture in muratura:

1 costruzioni in muratura ordinaria

2 costruzioni in muratura armata

3 costruzioni in muratura armata con progettazione in capacità

4 costruzioni in muratura confinata

5 costruzioni in muratura confinata con progettazione in capacità

desc_struttura	out	la descrizione della tipologia di struttura corrispondente al tipo della voce precedente; ignorata in input
regolarita_alt	in	la regolarità in altezza della struttura: "Regolare" o "Non regolare"
regolarita_pnt	in	la regolarità in pianta della struttura: "Regolare" o "Non regolare"
stato_limite	in	Un valore tra "Definibile", "Operatività SLO", "Danno SLD", "Salvaguardia vita SLV" o "Collasso SLC", che indica lo stato limite
prob_superamento	in	indica la probabilità di eccedenza
vita	in	indica il periodo di riferimento in anni

longitudine	in	indica la longitudine del punto geografico
latitudine	in	indica la latitudine del punto geografico
fattore_q	out	il fattore di struttura "q", calcolato in uscita in base ai parametri di ingresso
fattore_n	out	il fattore di struttura "η", calcolato in uscita in base ai parametri di ingresso
calcolo_q	in	un valore tra "Si" e "No" per attivare il calcolo automatico del fattore di struttura q
valori	out	tabella di valori della funzione calcolati in uscita: ogni valore della funzione è rappresentato da una tabella col valore x in posizione 1 e il valore y in posizione 2

convert_length(value, from, to)

restituisce il valore della lunghezza **value**, espressa nell'unità di misura **from**, convertito nell'unità di misura **to**.
Le unità di misura utilizzabili per **from** e **to** sono le seguenti:

length_mm	millimetri
length_cm	centimetri
length_m	metri

convert_force(value, from, to)

restituisce il valore della forza **value**, espressa nell'unità di misura **from**, convertito nell'unità di misura **to**.
Le unità di misura utilizzabili per **from** e **to** sono le seguenti:

force_kg	kilogrammi
force_N	Newton
force_kN	kiloNewton
force_dN	decaNewton

convert_pressure(value, from, to)

restituisce il valore della pressione **value**, espressa nell'unità di misura **from**, convertito nell'unità di misura **to**.
Le unità di misura utilizzabili per **from** e **to** sono le seguenti:

pressure_kg_mmq	kilogrammi per millimetro quadrato
pressure_kg_cmq	kilogrammi per centimetro quadrato
pressure_kg_mq	kilogrammi per metro quadrato
pressure_N_mmq	Newton per millimetro quadrato
pressure_N_cmq	Newton per centimetro quadrato
pressure_N_mq	Newton per metro quadrato
pressure_kN_mmq	kiloNewton per millimetro quadrato
pressure_kN_cmq	kiloNewton per centimetro quadrato
pressure_kN_mq	kiloNewton per metro quadrato
pressure_dN_mmq	decaNewton per millimetro quadrato
pressure_dN_cmq	decaNewton per centimetro quadrato
pressure_dN_mq	decaNewton per metro quadrato

parse_text_file(path)

Legge un file di testo dal percorso specificato e lo interpreta per restituire una struttura tabellare: ogni riga non vuota del file corrisponde ad un campo della tabella restituita; ogni campo contiene una tabella riempita con tutte le espressioni separate da spazi trovate nella riga.

Ad esempio consideriamo un file contenente il testo:

```
0.0    3.0
x      y      z
```

Esso viene interpretato per formare una tabella corrispondente alla seguente:

```
{ { 0.0, 3.0 }, { "x", "y", "z" } }
```

Un parametro opzionale di tipo stringa può indicare la stringa di separazione tra i campi di una riga di testo. Ad esempio specificando ",", i campi vengono interpretati come separati da una virgola.

Questa funzione permette ad esempio di importare nei propri script un tabulato generato al di fuori dell'applicazione, come ad esempio uno spettro di risposta, dati di sagomario o altro.

verify_file_name(path)

Verifica che il percorso specificato corrisponda ad un file effettivamente esistente. Il percorso può essere sia assoluto che relativo. Il file viene cercato nei percorsi noti all'applicazione fino a quel momento e se non viene trovato viene richiesto all'utente. Dal momento in cui l'utente specifica un percorso per il file, il percorso viene associato a quel file e la cartella a cu fa riferimento aggiunta a quelle considerate per le successive ricerche. La funzione restituisce il percorso assoluto del file oppure il valore nullo se il file non viene trovato e l'utente sceglie di non specificarlo (chiudendo il dialogo di richiesta con il tasto 'Annulla').

Questa funzione viene chiamata automaticamente ogni volta che si sceglie di specificare in un blocco funzionale nel testo un percorso fisso per un file da inserire.

Si noti che se il parametro non corrisponde ad un file trovato immediatamente nel percorso di ricerca, l'utente può scegliere un file qualunque, anche con nome o estensione diversi da quelli specificati nel parametro. Da quel punto dell'elaborazione in poi il nome specificato viene associato sempre al file scelto e non viene più richiesto.

Funzioni di output

Attraverso lo scripting è possibile manipolare direttamente il documento di uscita attraverso le funzioni della libreria **out**.

Le funzioni descritte in seguito possono essere utilizzate in blocchi di comandi sia di struttura che funzionali, con due scopi principali: per controllare il formato del testo (ad esempio salvandone ed impostandone lo stato prima o dopo un blocco di testo o un altro tipo di output); per aggiungere al documento di uscita elementi che vanno dal semplice testo alle immagini e alle tabelle.

La libreria di output definisce alcuni valori enumerati di comodo:

out.endl corrisponde ad una stringa contenente un singolo accapo

out.tab corrisponde ad una stringa contenente una singola tablatura

out.append(s)

Aggiunge all'output il valore v.

Il parametro deve essere un numero od una stringa. Se v è una stringa, il suo valore viene aggiunto direttamente. Se v è un numero, il suo valore viene formattato secondo il formato numerico correntemente impostato. Ved. **out.set_number_format**.

out.append_number(v)

Aggiunge all'output il valore numerico v utilizzando il formato correntemente impostato (ved. **out.set_number_format**).

È possibile specificare un secondo parametro per indicare il formato numerico desiderato, a scelta tra i valori:

out.format_noexp	per ottenere sempre una rappresentazione in forma estesa, non esponenziale
out.format_exp_e o **out.format_exp_E**	per ottenere sempre una rappresentazione in forma esponenziale con la e minuscola o maiuscola rispettivamente
out.format_best_e o **out.format_best_E**	per ottenere la rappresentazione più compatta tra quella esponenziale e quella estesa in base al numero di cifre decimali desiderato

È possibile specificare come terzo parametro il numero di cifre decimali da utilizzare.

out.append_chapter_title(s)

Aggiunge all'output un titolo di paragrafo in base al livello e alle opzioni di generazione correnti e con s come valore del titolo.

Può essere specificato un valore numerico (positivo o negativo) che a valle della generazione del titolo viene aggiunto al numero di livello corrente: ad esempio specificando 1 i titoli generati successivamente apparterranno al livello successivo, mentre specificando -1 i titoli generati successivamente apparterranno al livello precedente. Si raccomanda di fare attenzione nell'uso di questi valori per tenere sempre sotto controllo la generazione corretta dei titoli: in particolare un incremento di livello dovrebbe sempre essere bilanciato da un successivo decremento.

out.append_page_break( )

Aggiunge all'output un'interruzione di pagina esplicita.

out.append_text_file(path)

Aggiunge all'output il contenuto del file di testo specificato. Il percorso passa attraverso la funzione **verify_file_path** (ved.) e non è quindi necessario specificare un percorso assoluto.

Poiché il testo viene interpretato come non formattato, esso assume lo stile di carattere e paragrafo attivi nel documento di uscita al momento dell'inserimento.

out.append_rtf_file(path)

Aggiunge all'output il contenuto del documento RTF specificato. Il percorso passa attraverso la funzione **verify_file_path** (ved.) e non è quindi necessario specificare un percorso assoluto.

Il contenuto del file viene inserito senza alcuna modifica allo stile di carattere e paragrafo in esso definiti.

out.append_image_file(path)

Aggiunge all'output un'immagine letta dal file specificato. Il percorso passa attraverso la funzione **verify_file_path** (ved.) e non è quindi necessario specificare un percorso assoluto.

out.append_plot(w, h, t)

Genera un grafico e lo aggiunge all'output come immagine.

I primi due parametri indicano la larghezza e l'altezza dell'immagine da generare espresse in centimetri.

Il terzo parametro della funzione indica una tabella di punti. Ogni punto è rappresentato da un array in cui il primo elemento è la X, il secondo la Y, il terzo il tipo di punto, il quarto lo spessore della curva da quel punto in poi ed il quinto il colore da quel punto in poi. I valori dal terzo al quinto sono opzionali.

Il tipo di punto è uno tra i seguenti valori enumerati:

out.point_none	standard, la curva passa per questo punto
out.point_quotes	la curva passa per questo punto e vengono scritte le coordinate del punto sul grafico
out.point_circlet	la curva passa per questo punto ed esso viene cerchiato
out.point_triangle	la curva passa per questo punto ed esso viene circondato da un triangolino
out.point_nodraw	il punto non viene considerato per il disegno della curva ma solo per estendere i confini del sistema di riferimento
out.point_newcurve	il punto segna l'inizio di una nuova curva e non viene connesso al punto precedente

Lo spessore è un valore intero che di norma vale 1, ovvero il minor valore per cui la curva è visibile.

Il colore va specificato come array di tre valori numerici compresi tra 0 e 255 che sono rispettivamente la componente rossa, quella verde e quella blu del colore (ad es. "`{ 255, 0, 0 }`" rappresenta il rosso pieno, "`{ 255, 255, 0 }`" rappresenta il giallo e così via).

Può essere specificato per la funzione un quarto parametro di tipo tabella che indica le opzioni per la generazione del grafico. Ogni opzione è opzionale e specificata come campo avente indice di tipo stringa opportuno, tra i seguenti:

draw_grid	opzione per il disegno della griglia, **true** o **false**
draw_sub_grid	opzione per il disegno della griglia più fine, **true** o **false**
show_origin	opzione per fare in modo che l'origine degli assi sia sempre nel quadro, **true** o **false**

show_axes	opzione per mostrare sempre gli assi anche con l'origine fuori quadro, **true** o **false**
step_x	passo esplicito da utilizzare sull'asse delle ascisse, di norma viene scelto automaticamente
step_y	passo esplicito da utilizzare sull'asse delle ordinate, di norma viene scelto automaticamente
label_x	etichetta di testo per descrivere l'asse delle ascisse
label_y	etichetta di testo per descrivere l'asse delle ordinate
pen_color e **pen_width**	colore e spessore di default della curva, da specificare con le stesse modalità delle proprietà del punto sopra descritte

La griglia, se richiesto, viene disegnata in corrispondenza dei passi di ascissa ed ordinata specificati o calcolati automaticamente in base all'estensione del grafico. La griglia più fine disegna linee aggiuntive tra i passi della griglia.

out.append_table_row(w, t)

Aggiunge all'output una riga di tabella larga w e contenente una cella per ogni valore di t.

I valori di t di tipo stringa vengono aggiunti così come sono; i valori di tipo numerico sono invece convertiti secondo il formato numerico corrente (ved. **set_number_format**). Il formato del carattere utilizzato è quello corrente nel documento d uscita.

La larghezza w, espressa in centimetri, viene divisa equamente per il numero di valori per determinare la larghezza della singola cella.

Se si desidera generare un'intera tabella, può essere utilizzata questa funzione per ogni riga della tabella: le righe aggiunte consecutivamente formeranno un'unica tabella.

Il formato delle celle può essere opzionalmente specificato come terzo parametro: può essere specificato un solo formato per tutte le celle oppure un array di formati da applicare per ogni cella; se il numero di celle è maggiore dei formati specificati, per quelle in eccesso viene utilizzato l'ultimo formato presente.

Ogni formato di cella va espresso come una tabella di attributi. Ogni attributo può avere un indice numerico o di tipo stringa Se un attributo non viene specificato, il suo valore viene ereditato dalla cella precedente. Gli attributi modificabili sono:

width	(1)	larghezza in centimetri della cella o **out.cell_autowidth** per specificare che la cella prenderà una parte dello spazio libero
style	(2)	stile del testo della cella, un singolo valore o un array; ved. **out.set_font_style**
align	(3)	allineamento del testo nella cella; ved. **out.set_alignment**
fontface	(4)	nome del carattere da utilizzare per la cella o **out.cell_autofont** per specificare che la cella utilizzerà il carattere utilizzato intorno alla tabella; ved. **out.set_font**

fontsize (5) dimensione in punti del carattere da utilizzare per la cella oppure **out.cell_autofontsize** per indicare che la cella utilizzerà la dimensione del carattere utilizzato intorno alla tabella

fontcolor (6) colore del carattere da utilizzare per la cella, espresso come array di tre valori: rosso, verde e blu rispettivamente; ved. **out.set_font_color**

La possibilità di usare sia l'indice che il nome per un attributo permette di definire agevolmente alcuni attributi lasciandone vuoti altri e rendere la definizione intuitiva: ad es. "`{ 3.6, fontsize = 12, fontface = "arial" }`" definisce la larghezza della cella, il nome e la dimensione del carattere da utilizzare; gli altri parametri vengono ereditati dalla cella precedente o dal contesto in cui è inserita la tabella.

Un quarto parametro opzionale permette di specificare le opzioni relative ai bordi delle celle della tabella. Il valore può essere un singolo valore enumerato oppure un array di valori enumerati a scelta tra i seguenti:

out.border_default questo è il valore di default e comprende tutti gli altri: i bordi sono tutti visibili

out.border_top se presente viene disegnato il bordo superiore delle celle

out.border_bottom se presente viene disegnato il bordo inferiore delle celle

out.border_left se presente viene disegnato il bordo sinistro della riga

out.border_right se presente viene disegnato il bordo destro della riga

out.border_sides comprende sia il bordo destro che quello sinistro della riga

out.border_between_cells se presente viene disegnato il bordo tra le celle della riga

Ad esempio per mostrare tutti i bordi tranne quelli interni si può specificare "`{ out.border_top, out.border_bottom, out.border_sides }`". Il risultato sarà simile alla seguente tabella:

a	b	c

La tabella rappresentata è ottenuta con un blocco di comandi funzionale, col codice:

```
local w = 6
local t = { "a", "b", "c" }
local o = { align=out.align_center, fontsize=10 }
local b = { out.border_top, out.border_bottom, out.border_sides }
out.append_table_row(w, t, o, b)
```

out.set_alignment(v)

Imposta l'allineamento del testo a v. Il parametro v deve essere un valore enumerato tra i seguenti:

out.align_center	allineamento centrale del testo
out.align_left	allineamento a sinistra del testo
out.align_right	allineamento a destra del testo
out.align_justified	allineamento giustificato del testo

out.get_alignment( )

Restituisce il tipo di allineamento del testo corrente. Il valore restituito è uno tra quelli enumerati descritti nella funzione **out.set_alignment**.

out.set_paratype(v)

Imposta il tipo di paragrafo corrente a v. Il parametro v può essere uno a scelta tra i seguenti valori enumerati:

out.para_normal	paragrafo normale
out.para_bullet	paragrafo da elenco puntato
out.para_numbers	paragrafo da elenco con numerazione araba
out.para_letters	paragrafo da elenco con numerazione in lettere minuscole
out.para_letters_capital	paragrafo da elenco con numerazione in lettere maiuscole
out.para_roman	paragrafo da elenco con numerazione in numeri romani minuscoli
out.para_letters_capital	paragrafo da elenco con numerazione in numeri romani maiuscoli

Se il tipo di paragrafo specificato non è normale o da elenco puntato è possibile specificare con un secondo parametro opzionale lo stile di numerazione, con un valore enumerato tra i seguenti:

out.numbers_parenthesis	parentesi tonda chiusa dopo il numero

out.numbers_both_parentheses il numero è racchiuso tra parentesi tonde

out.numbers_period punto dopo il numero

out.numbers_alone solo il numero

out.get_paratype( )

Restituisce il tipo di paragrafo e lo stile di numerazione impostati nel paragrafo corrente. La coppia di risultati può assumere valori enumerati descritti per la funzione **out.set_paratype**.

out.set_paraspacing(a, b)

Imposta lo spazio in punti rispettivamente da anteporre e da posporre al paragrafo corrente.

out.get_paraspacing( )

Restituisce una coppia di valori che indicano lo spazio in punti che viene anteposto e posposto al paragrafo corrente e che può essere impostato con la funzione **out.set_paraspacing**.

out.set_font(f)

Imposta il carattere corrente a quello indicato col nome f. Il parametro f può assumere uno dei nomi di font disponibili nel sistema e visibili normalmente nei dialoghi di scelta dei caratteri, come "Times New Roman", "Arial" e così via.

Un secondo parametro opzionale può indicare la dimensione in punti del carattere da utilizzare.

Un terzo parametro opzionale può indicare il set di caratteri da utilizzare per il carattere. Questo parametro può essere necessario ad esempio per l'utilizzo di font di simboli e può assumere uno tra i seguenti valori enumerati:

out.charset_ansi il set di caratteri predefinito per le lingue occidentali

out.charset_default il set di caratteri predefinito per il sistema in uso, in genere coincidente con **out.charset_ansi**

out.charset_symbol il set di caratteri specifico per i simboli

out.charset_oem il set di caratteri specifico del sistema in uso

Tutti i parametri possono assumere il valore **nil** se non si intende specificarli. Ad esempio se si vuole impostare solo la dimensione del carattere senza cambiare il tipo si può scrivere:

```
out.set_font( nil, 10 )
```

out.get_font( )

Restituisce tre valori che indicano rispettivamente il tipo di carattere, la dimensione ed il set di caratteri correntemente utilizzati dal testo. Questi valori possono essere impostati con la funzione **out.set_font**.

out.set_font_color(t)

Imposta il colore corrente del testo. Il parametro t deve essere una tabella con tre valori numerici compresi tra 0 e 255 che indicano rispettivamente la componente rossa, quella verde e quella blu del colore da utilizzare.

out.get_font_color( )

Restituisce una tabella di tre valori che descrive il colore corrente del testo nel documento di uscita. I tre valori numerici son compresi tra 0 e 255 e corrispondono rispettivamente alla componente rossa, verde e blu del colore utilizzato. Il colore può essere impostato con la funzione **out.set_font_color**.

out.set_font_style(v)

Imposta lo stile corrente del carattere del documento di uscita. Il parametro v può essere un singolo valore o una tabella di valori enumerati a scelta tra i seguenti:

out.style_bold	effetto grassetto
out.style_italic	effetto italico
out.style_underline	effetto sottolineato
out.style_strikeout	effetto sbarrato
out.style_subscript	testo in pedice; non può essere utilizzato con **out.style_superscript**
out.style_superscript	testo in apice; non può essere utilizzato con **out.style_subscript**

out.get_font_style( )

Restituisce una tabella contenente tutti gli effetti di carattere correntemente attivi nel testo del documento di uscita. Se non è attivo alcun effetto la tabella è vuota. I valori che la tabella di uscita può contenere sono i valori enumerati descritti per la funzione **out.set_font_style**.

out.set_tab_space(v)

Imposta la dimensione in centimetri di ogni interruzione di tablatura, ovvero la distanza tra due tablature successive.

out.get_tab_space( )

Restituisce lo spazio in cm correntemente impostato tra due tablature successive del documento di uscita. Se per il paragrafo corrente gli spazi non sono assegnati in modo uniforme per tutte le tablature viene restituito 0.

out.set_tab_stops(t)

Imposta le tablature del paragrafo corrente. Il parametro t deve essere una tabella. Ogni valore di t deve essere a sua volta una tabella di due valori: posizione in centimetri rispetto al margine sinistro e tipo della tablatura.

Il tipo della tablatura può essere composto dalla somma di valori enumerati di due gruppi.

I valori enumerati del primo gruppo indicano l'allineamento del testo rispetto alla posizione di tablatura; va scelto un valore tra i seguenti (se non ne viene specificato alcuno viene usato quello predefinito):

out.tab_align_default	il valore predefinito, che indica che il testo sarà a sinistra della tablatura
out.tab_align_center	indica che il testo sarà centrato sulla posizione della tablatura
out.tab_align_right	indica che il testo sarà a destra della tablatura

I valori enumerati del secondo gruppo indicano il tipo di collegamento grafico con il testo della tablatura precedente; se non ne viene specificato alcuno, viene usato il valore predefinito che indica che non si desidera alcun collegamento grafico. I valori possibili sono:

out.tab_lead_dotted	collegamento con puntini
out.tab_lead_dashed	collegamento con trattini
out.tab_lead_underlined	collegamento con linea di sottolineatura
out.tab_lead_thickline	collegamento con linea continua
out.tab_lead_doubleline	collegamento con uguali (doppia linea continua)

Ad esempio per ottenere una tablatura a 10 cm dal margine sinistro, allineamento a destra e collegata alla precedente con puntini va specificato "`{ 10, out.tab_lead_dotted + out.tab_align_right }`".

out.get_tab_stops( )

Restituisce una tabella con le impostazioni di tutte le tablature definite per il paragrafo corrente. Ogni valore della tabella è una tabella di due elementi dove il primo valore indica la posizione in centimetri dal margine sinistro e il secondo valore indica il tipo di tablatura come descritto per la funzione **out.set_tab_stops**.

out.indent( )

Sposta l'inizio del paragrafo corrente sulla tablatura successiva all'attuale.

Può essere specificato come parametro aggiuntivo il numero di tablature di cui spostare in avanti il paragrafo.

out.outdent( )

Sposta l'inizio del paragrafo corrente sulla tablatura precedente all'attuale, se essa non ha posizione 0.

Può essere specificato come parametro aggiuntivo il numero di tablature massime di cui spostare indietro il paragrafo. Se il numero specificato è maggiore dell'indice di tablatura corrente, il testo viene spostato in 0.

out.set_number_format(f, n)

Imposta il formato da utilizzare per la stampa dei valori numerici attraverso le funzioni **out.append** e derivate.

Il primo parametro deve assumere un valore enumerato tra quelli disponibili:

out.format_noexp	per ottenere sempre una rappresentazione in forma estesa, non esponenziale
out.format_exp_e o **out.format_exp_E**	per ottenere sempre una rappresentazione in forma esponenziale con la *e* minuscola o maiuscola rispettivamente
out.format_best_e o **out.format_best_E**	per ottenere la rappresentazione più compatta tra quella esponenziale e quella estesa in base al numero di cifre decimali desiderato

Il secondo parametro indica il numero di cifre decimali da mostrare.

out.set_char_style(index)

Imposta lo stile corrente nel documento in uscita. Il parametro *index* deve essere compreso tra 1 e 6 ed identifica uno degli stili di carattere personalizzati.

Funzioni di interazione

Attraverso lo scripting è possibile gestire l'interazione con l'utente durante la generazione attraverso le funzioni della libreria **dlg**.

Le funzioni descritte di seguito possono essere utili per influenzare il valore di variabili di scripting o il flusso dell'elaborazione, in modo che il documento di uscita rispecchi parametri indicati dall'utente direttamente durante la generazione, senza dover modificare il documento EasyQuill di partenza.

L'interazione con l'utente avviene sempre tramite finestre di dialogo. La maggior parte delle funzioni qui descritte mostrano all'utente un dialogo preconfigurato per l'inserimento di un certo tipo di valore (un numero o un breve testo, oppure una

risposta di tipo Sì / No). È poi possibile generare dei moduli personalizzati da presentare all'utente per indicare più parametr contemporaneamente o semplicemente per mostrare un'interfaccia diversa da quella delle funzioni standard (ved. **dlg.do_form**).

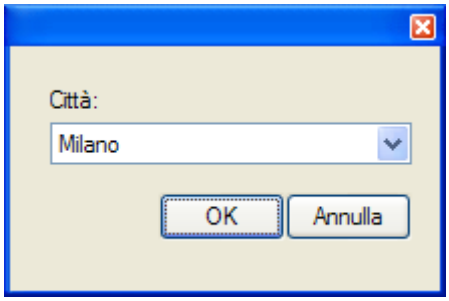
dlg.ask(s, t)

Mostra all'utente un dialogo con etichetta *s* per la scelta di una tra le opzioni disponibili nella tabella di stringhe *t*. Se l'utente sceglie OK, viene restituita la stringa di *t* correntemente selezionata. Se l'utente esce con Annulla, viene restituito **nil**.

Ad esempio con la chiamata:

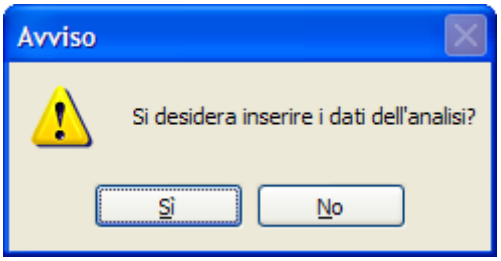
```
dlg.ask('Città:', { 'Roma', 'Milano', 'Napoli' })
```

viene mostrato il seguente dialogo, in cui le tre scelte specificate sono disponibili nel menu a tendina centrale:



dlg.ask_yesno(s)

Mostra all'utente un dialogo di avviso con il testo del parametro *s* e permette la scelta tra un bottone Sì e un bottone No. Viene restituito **true** se l'utente sceglie Sì, **false** se sceglie No.

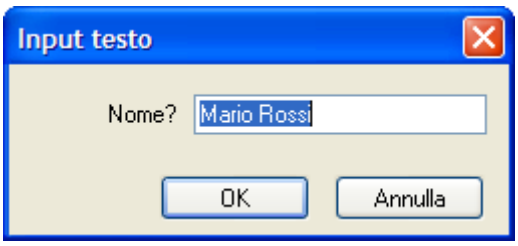


dlg.ask_text( )

Mostra all'utente un dialogo in cui può essere inserita una breve stringa di testo. Il valore restituito è la stringa inserita se l'utente preme il tasto OK, il valore **nil** se preme Annulla.

Come parametro opzionale può essere specificato un breve testo come domanda per l'inserimento.

Un secondo parametro opzionale può indicare il valore predefinito della risposta.

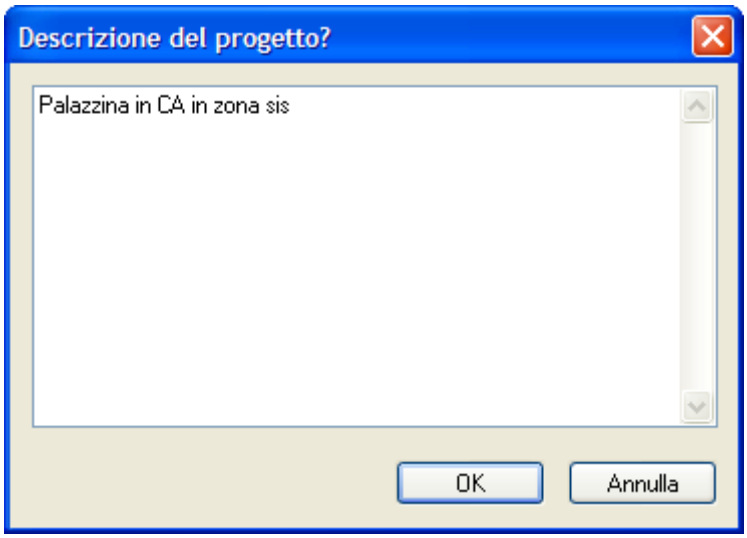


dlg.ask_text_block( )

Mostra all'utente un dialogo in cui può essere inserito un testo di lunghezza arbitraria. Il valore restituito è la stringa inserita se l'utente preme il tasto OK, il valore **nil** se preme Annulla.

Come parametro opzionale può essere specificato un breve testo come titolo del dialogo.

Un secondo parametro opzionale può indicare il valore predefinito della risposta.



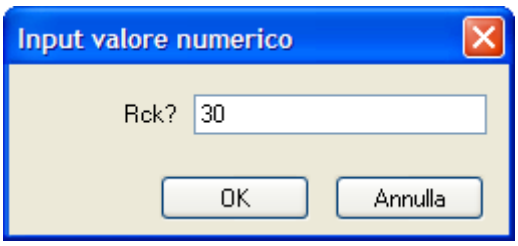
dlg.ask_number(s)

Mostra all'utente un dialogo in cui può essere inserito un valore numerico. Il parametro *s* può essere un breve testo usato come domanda per l'inserimento. Il valore restituito è il numero inserito se l'utente preme il tasto OK, il valore **nil** se preme Annulla.

Come secondo parametro opzionale può essere specificato il valore predefinito della risposta.

Un terzo e un quarto parametro opzionali possono indicare i valori rispettivamente minimo e massimo per cui il numero è considerato valido. Se il numero inserito dall'utente è al di fuori dei limiti consentiti, il tasto OK viene reso grigio.

Un quinto parametro opzionale booleano indica se il valore deve essere intero (quando è **true**) o può essere un qualunque numero reale (quando il parametro è **false**).



dlg.ask_file( )

Mostra il dialogo standard di apertura file affinché l'utente ne scelga uno. Il valore restituito è il percorso del file scelto se l'utente preme il tasto OK, il valore **nil** se preme Annulla.

Come parametro opzionale può essere specificato un breve testo come titolo del dialogo.

Un secondo parametro opzionale può essere una tabella contenente la descrizione dei filtri sui file da utilizzare nel dialogo. Ogni filtro è descritto da un valore enumerato tra quelli predefiniti oppure da una tabella di due valori in cui il primo valore è una descrizione testuale ed il secondo è un filtro wildcard del tipo "*.ext". I filtri predefiniti disponibili come valori enumerat sono:

dlg.filter_allfiles	filtro su tutti i file, andrebbe di norma sempre inserito come ultimo elemento della tabella dei filtri ed è quello predefinito quando non viene specificato alcun filtro
dlg.filter_textfiles	filtro su tutti i file con estensione TXT, ovvero lo standard per file di testo non formattato
dlg.filter_rtffiles	filtro su tutti i file con estensione RTF, ovvero lo standard per i file di testo formattato che possono essere letti ad esempio con **out.append_rtf_file**
dlg.filter_imagefiles	filtro su tutti i file con estensione BMP, JPG e PNG, ovvero gli standard più comuni per file contenenti immagini e che possono essere lette ad esempio con **out.append_image_file**
dlg.filter_nolian	filtro su tutti i file con estensione SAP, ovvero lo standard per i file creati da Nòlian e dai suoi post-processori

dlg.ask_spectrum( )

Mostra il dialogo per la generazione di uno spettro di risposta.

In uscita i parametri vengono inseriti in una tabella che viene restituita dalla funzione se l'utente preme il bottone OK. Se l'utente preme invece Annulla viene restituito il valore **nil**.

Un parametro di tipo stringa opzionale può indicare il testo da utilizzare per il titolo della finestra.

La struttura restituita può essere utilizzata ad esempio dalla funzione **calc_spectrum** alla cui documentazione si rimanda per la descrizione della struttura stessa.

dlg.do_form(f)

Interpreta lo script passato come parametro secondo la sintassi descritta nel paragrafo seguente "Sintassi per i moduli" e genera un dialogo che contiene il modulo di inserimento generato dall'interpretazione.

Questa funzione interpreta lo script nello stesso modo del blocco di struttura form (ved.). La differenza sta nel fatto che questa funzione restituisce un valore che può indicare quale bottone è stato premuto in uscita dal modulo mentre il blocco d struttura modifica sempre la variabile globale **form_exit_button**.

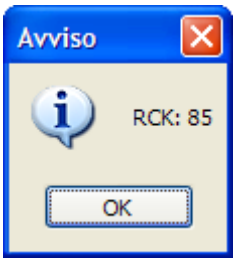
dlg.show_msg(s)

Mostra all'utente il messaggio di avviso *s* durante l'elaborazione. Può essere utile per dare la possibilità di verificare durante l'elaborazione il valore di una o più variabili.

Ad esempio il codice:

```
dlg.show_msg('RCK: ' .. rck )
```

produce durante l'elaborazione un dialogo come quello in figura:



Sintassi per i moduli

Un modulo (o form) può essere utilizzato in EasyQuill per gestire una interazione con l'utente in cui possano essere definiti più valori contemporaneamente o possa essere meglio curato l'aspetto grafico rispetto ai dialoghi standard per la richiesta d valori.

I moduli possono essere utilizzati attraverso blocchi di struttura form appositi oppure direttamente dallo scripting attraversc la funzione **dlg.do_form**. In entrambi i casi va scritta una definizione del modulo attraverso una particolare sintassi molto simile a quella HTML per i form e che viene descritta di seguito.

La descrizione di un modulo è formata da una combinazione di testo semplice e controlli. Il testo semplice viene inserito direttamente nel modulo; i controlli vengono inseriti nel flusso del testo attraverso regole empiriche generali per la generazione dello schema finale del modulo. Ai fini dello schema finale contano in particolare alcuni controlli (ad esempio
 e <TAB>, ved. sotto) e le dimensioni richieste per i controlli attraverso la definizione dei loro attributi. La dimensione totale del modulo viene calcolata automaticamente in modo che esso contenga tutti i controlli definiti.

I controlli vanno definiti tra parentesi angolari, in stile HTML:

```
<tipo attributo1="valore1" ...>
```

Tipo è una parola chiave che definisce qual è il tipo del controllo da aggiungere. Può seguire un numero imprecisato di

attributi a cui vengono assegnati dei valori per specificare delle opzioni per quel controllo. La sintassi dei moduli non è sensibile alle maiuscole: tutti gli identificatori di tipo e attributo possono essere indifferentemente formati da lettere maiuscole o minuscole.

I tipi di controllo riconosciuti sono:

INPUT definisce un controllo in base all'attributo *type*. I tipi disponibili sono:

static	per creare un testo statico; rispetto al testo semplice ha la possibilità di avere specificata una dimensione fissa, più righe e di prendere il valore ad ogni aggiornamento da una variabile
text	per creare un semplice campo di testo
button	per creare un bottone
check	per creare un check box
radio	per creare un radio button
textarea	per creare un box di testo

Per i controlli di INPUT sono disponibili i seguenti attributi:

name	il nome del controllo; può essere il nome di una variabile a cui viene assegnato il valore inserito; nel caso dei radio button più radio button devono avere lo stesso nome per poter formare un gruppo: la variabile nominata alla fine verrà impostata coll'attributo *value* del radio button selezionato
value	il valore iniziale per i campi di testo oppure quello che viene assegnato alla variabile definita in *name* quando un controllo button, check o radio viene selezionato; per questi controlli il valore viene usato anche come etichetta quando è assente l'attributo *caption*
caption	il valore dell'etichetta dei controlli button, check e radio
checked	se presente questo attributo, senza valore assegnato, un checkbox o un radio button ha inizialmente il valore selezionato
command	il valore di questo attributo, se presente, viene eseguito come fosse una espressione di scripting quando viene cliccato un controllo button, check o radio oppure viene modificato il testo di un controllo text o textarea; l'espressione può contenere qualsiasi istruzione valida, comprese chiamate a funzione, e può contenere più istruzioni concatenate separate da punto e virgola
exit	se presente questo attributo, senza valore assegnato, un bottone viene considerato come di uscita dal modulo; il suo valore viene assegnato alla variabile d'uscita (quella predefinita è **form_exit_button**)
cancel	se presente questo attributo, senza valore assegnato, un bottone viene considerato come di annullamento dell'operazione; il valore assegnato alla variabile d'uscita è **nil**

size	la dimensione orizzontale del controllo, espressa in caratteri, usata per static, text, button e textarea
height	la dimensione verticale del controllo, espressa in caratteri, usata solo per static e textarea

SELECT / LIST definiscono rispettivamente un controllo di tipo combo box o di tipo lista. Le voci disponibili vanno definite attraverso l'attributo *list* oppure attraverso dei controlli interni di tipo **OPTION**. Gli attributi disponibili per questi controlli sono:

name	il nome della variabile a cui viene assegnato il valore selezionato
value	il valore inizialmente selezionato
index	se presente e senza valori assegnati, indica che il valore restituito deve essere l'indice della voce selezionata e non il valore dell'etichetta
list	definisce le voci disponibili per il controllo in un unica stringa di espressioni divise da un simbolo di backslash '\': ad esempio il valore "a\b\c" aggiunge tre voci al controllo
command	il valore di questo attributo, se presente, viene eseguito come fosse una espressione di scripting ogni volta che viene selezionata una voce nel combo box o nella lista; l'espressione può contenere qualsiasi istruzione valida, comprese chiamate a funzione, e può contenere più istruzioni concatenate separate da punto e virgola
size	la dimensione orizzontale del controllo, espressa in caratteri
height	la dimensione verticale del controllo, espressa in caratteri, usata solo per la lista

Le voci del controllo possono essere definite attraverso controlli contenuti di tipo **OPTION**. La forma diventa:

```
<SELECT ...><OPTION>a</OPTION><OPTION>b</OPTION></SELECT>
```

Il controllo OPTION può avere, oltre all'attributo *value* che definisce l'etichetta e il valore assegnato alla variabile de controllo, un attributo aggiuntivo *selected*, senza valori assegnati, che indica che il valore è quello inizialmente selezionato.

BR è un controllo fittizio per indicare un'interruzione di linea; non ha bisogno di attributi.

TAB è un controllo fittizio per indicare che il controllo successivo va allineato alla tablatura seguente; una tablatura è qui definita come uno spazio orizzontale corrispondente ad un testo statico di quattro lettere 'M' maiuscole; può essere specificato l'attributo *size* per indicare il numero di caratteri da utilizzare invece dei quattro predefiniti.

TITLE è un controllo fittizio che imposta il titolo del modulo al valore dell'attributo *value*.

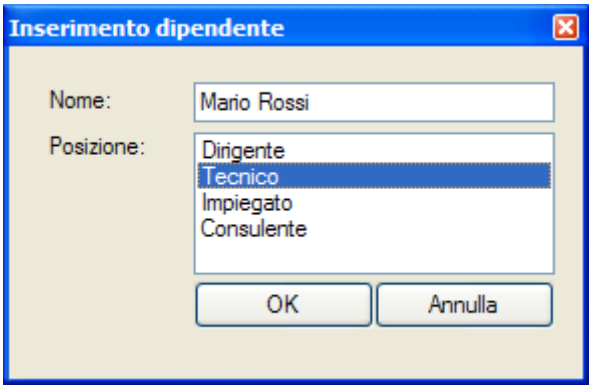
Alcuni caratteri sono riservati per la definizione dei controlli e non dovrebbero essere utilizzati nel testo semplice e nel valore degli attributi: al loro posto vanno utilizzate le corrispondenti entità, esattamente come avviene in HTML. Le entità sono parole chiave sempre racchiuse tra una e commerciale (&) e un punto e virgola (;). I caratteri incriminati (e le corrispondenti entità tra parentesi) sono: **&** (*&*), **<** (*<*), **>** (*>*), **"** (*"*), **'** (*'*).

Nei valori degli attributi dei controlli e nel testo semplice possono essere inserite espressioni introdotte dal simbolo di cancelletto '#': queste espressioni vengono interpretate dal motore di scripting e sostituite col valore calcolato. La sostituzione è la stessa descritta nel capitolo "Espressioni di scripting nel titolo", riferito ai blocchi strutturali di testo.

Le espressioni di scripting valide sono di due tipi: semplici identificatori oppure espressioni complesse. Un'espressione semplice può essere il nome di una variabile e rispetta la convenzione dei nomi vista nel capitolo "Convenzioni lessicali": in questo caso viene semplicemente sostituito il valore di quella variabile. Un'espressione complessa va racchiusa tra parentesi tonde e può contenere una o più istruzioni, riferimenti a tabella, chiamate a funzione ecc.; viene sostituita con il primo valor che essa restituisce.

Ad esempio l'espressione semplice "`#var1`" viene sostituita con il valore della variabile *var1*. L'espressione complessa "`#(math.sin(math.pi/2))`" viene sostituita col valore del seno a 90°. Si noti che è necessario utilizzare un'espressione complessa per accedere a valori di una tabella: è un'espressione corretta "`#(v[3])`", ma non "`#v[3]`".

Per l'attributo *name* è consentito solo il tipo di espressione semplice per indicare la variabile su cui viene memorizzato il valore finale del controllo. Se il nome non inizia con un cancelletto, allora il valore non viene memorizzato su alcuna variabile del motore di scripting.



Il semplice modulo rappresentato in figura è definito dalla seguente descrizione:

```
<TITLE value="Inserimento dipendente">
Nome:<tab size=8>
<INPUT type="text" name="#nome"><br />
Posizione:<tab size=8>
<LIST name="#posizione">
       <OPTION value="Dirigente">        <OPTION value="Tecnico">
<OPTION value="Impiegato">        <OPTION value="Consulente"> </LIST><br />
<tab size=8><INPUT type="button" value="OK" exit>
<INPUT type="button" value="Annulla" cancel>
```

In uscita dal modulo col bottone OK, la variabile "nome" avrà valore "Mario Rossi" e la variabile "posizione" avrà valore "Tecnico".

Interazione con All In One

Le funzioni della libreria **aio** qui descritta consentono di interrogare l'applicazione All In One e sono naturalmente disponibili solo qualora lo script stia funzionando al suo interno.

aio.has_ambient(ambient)

Restituisce **true** se l'ambiente specificato è disponibile.

Il parametro *ambient* indica l'ambiente richiesto e può assumere uno tra i seguenti valori enumerati:

aio.ambient_inmod

aio.ambient_ee

aio.ambient_easybeam

aio.ambient_easysteel

aio.ambient_easywall

aio.ambient_exsys

aio.ambient_exsys_frp

aio.ambient_quarmon

aio.ambient_donjon

aio.ambient_wallverine

aio.ambient_wallverine_frp

aio.ambient_elwood

Accesso al progetto

Il motore di scripting di EasyQuill offre un certo numero di funzioni per l'accesso alle informazioni contenute nel progetto corrente, in modo da ottenere dati utili alla compilazione di una relazione tecnica.

L'accesso ai dati della struttura avviene sempre in base ad uno stato di computazione: ci sono funzioni per impostare lo stato (con prefisso **use_**, ad esempio l'indice dell'analisi o del gruppo di risultati da considerare) e funzioni che restituiscono risultati in base allo stato corrente (con prefisso **get_**, ad esempio quali gruppi di risultati sono disponibili per il file e l'analisi correntemente impostati).

Tutte le funzioni di questa libreria permettono solo la lettura dei dati contenuti nel progetto, senza eseguire alcun calcolo. Funzioni più complesse, che eseguono calcoli o creano immagini, prevedono l'accesso diretto a specifici ambienti e sono

descritte nei relativi capitoli successivi.

sap.use_analysis(v)

Imposta l'analisi corrente per le successive interrogazioni. Il parametro *v* può indicare un identificatore numerico di chiave o un nome tra quelli restituiti dalla funzione **sap.get_analyses**.

sap.use_mode(v)

Imposta l'indice del modo di vibrare da considerare nelle funzioni successive.

sap.use_displacements(k)

Imposta il set di risultati di tipo "spostamenti" da utilizzare nelle funzioni successive. Il parametro *k* deve essere la chiave numerica relativa al set voluto, ad esempio una di quelle restituite dalla funzione **sap.get_displacements**.

sap.use_stresses(k)

Imposta il set di risultati di tipo "sforzi" da utilizzare nelle funzioni successive. Il parametro *k* deve essere la chiave numerica relativa al set voluto, ad esempio una di quelle restituite dalla funzione **sap.get_stresses**.

sap.use_reactions(k)

Imposta il set di risultati di tipo "reazioni" da utilizzare nelle funzioni successive. Il parametro *k* deve essere la chiave numerica relativa al set voluto, ad esempio una di quelle restituite dalla funzione **sap.get_reactions**.

sap.use_buckling(k)

Imposta il set di risultati di tipo "buckling" da utilizzare nelle funzioni successive. Il parametro *k* deve essere la chiave numerica relativa al set voluto, ad esempio una di quelle restituite dalla funzione **sap.get_buckling_results**.

sap.use_capacity(k)

Imposta il set di risultati di tipo "capacity" da utilizzare nelle funzioni successive. Il parametro *k* deve essere la chiave numerica relativa al set voluto, ad esempio una di quelle restituite dalla funzione **sap.get_capacity_results**.

sap.get_phases( )

Restituisce una tabella contenente i nomi di tutte le fasi presenti nel file corrente, ordinate per indice di fase.

sap.get_loadcases( )

Restituisce una tabella contenente i nomi delle condizioni di carico definite nel file corrente.

sap.get_analyses( )

Restituisce una tabella che descrive tutti i risultati di analisi salvati nel file corrente.

Ogni valore della tabella è a sua volta una tabella che ha i seguenti campi:

key	chiave dell'analisi, da utilizzare ad esempio in **sap.use_analysis**
name	nome assegnato dall'utente all'analisi
phase	indice della fase a cui l'analisi si riferisce
comment	commento assegnato dall'utente per descrivere l'analisi
type	tipo di analisi, uno tra i seguenti valori enumerati:

sap.analysis_combination	combinazione di risultati definita dall'utente
sap.analysis_static_linear	analisi statica lineare
sap.analysis_static_NL_frame	analisi statica non lineare per telai
sap.analysis_static_NL_incremental	analisi statica elasto-plastica o incrementale
sap.analysis_dynamic_spectral	analisi dinamica spettrale
sap.analysis_dynamic_seismic_NL	analisi dinamica sismica non lineare con effetti del secondo ordine
sap.analysis_timehistory_wilson	analisi con time-history di Wilson
sap.analysis_timehistory_newmark	analisi con time-history di Newmark
sap.analysis_buckling	analisi di instabilità
sap.analysis_capacity	analisi di capacità
sap.analysis_ee_static	analisi Earthquake Engineering statica
sap.analysis_ee_incremental	analisi Earthquake Engineering incrementale

sap.analysis_ee_transient	analisi Earthquake Engineering in transitorio
sap.analysis_ee_capacity	analisi Earthquake Engineering di capacità
sap.analysis_ee_modal	analisi Earthquake Engineering modale
sap.analysis_ee_ida	analisi Earthquake Engineering IDA
sap.analysis_ee_vulnerability	analisi Earthquake Engineering di vulnerabilità

sap.get_analysis_info( )

Restituisce una tabella che descrive alcuni parametri dell'analisi corrente. I campi definiti sono:

halfband	larghezza della semibanda della matrice usata per il calcolo
equations	numero di equazioni, o di righe e colonne della matrice usata per il calcolo
blocks	numero di blocchi in cui la matrice è stata divisa
blocks_equations	numero di equazioni di ogni blocco in cui la matrice è stata divisa
norm	zero algoritmico trovato nel calcolo
time	tempo in secondi impiegato per l'analisi

sap.get_spectral_cases( )

Restituisce una tabella che descrive gli attributi delle condizioni dinamiche utilizzate dall'analisi corrente.

Ogni valore della tabella è a sua volta una tabella contenente i seguenti campi:

name	nome della condizione di carico
spectrum	nome dello spettro di riferimento
x, **y**, **z**	tre campi per le tre componenti dell'accelerazione spettrale

sap.get_spectrum(nome_spettro)

Restituisce una tabella che descrive gli attributi dello spettro richiesto. I campi definiti sono:

type il tipo dello spettro, uno tra i seguenti valori enumerati:

- **sap.spectrum_type_generic** spettro non riconducibile ad un impiego specifico
- **sap.spectrum_type_slo** spettro per la valutazione dello stato limite di operatività
- **sap.spectrum_type_sld** spettro per la valutazione dello stato limite di danno
- **sap.spectrum_type_slv** spettro per la valutazione dello stato limite di salvaguardia della vita
- **sap.spectrum_type_slc** spettro per la valutazione dello stato limite di collasso

comment una stringa contenente il commento allo spettro

N.B. Per i valori **q**, **n** e **s** un valore negativo indica che il dato non è valido.

q il fattore di comportamento q

n il fattore di smorzamento "η"

s il coeff. categoria sottosuolo e condizioni topografiche

kw il coefficiente di riduzione di q per strutture con pareti

autocomputed_q **true** se il fattore di struttura q è stato calcolato automaticamente

is_seismic_action_horz **true** se l'azione sismica è orizzontale

is_horz_regular **true** se la struttura è regolare in pianta

is_vert_regular **true** se la struttura è regolare in altezza

material il tipo di materiale, uno tra i seguenti valori enumerati:

sap.spectrum_mat_concrete struttura in calcestruzzo armato

sap.spectrum_mat_steel struttura in acciaio

sap.spectrum_mat_steel_and_concrete struttura mista in calcestruzzo e acciaio

sap.spectrum_mat_wood struttura in legno

sap.spectrum_mat_masonry struttura in muratura

struct_type la tipologia della struttura, uno tra i seguenti valori enumerati:

per le strutture in calcestruzzo armato:

sap.spectrum_cst_1 strutture a telaio di un piano

sap.spectrum_cst_2 strutture a telaio con più piani ed una sola campata

sap.spectrum_cst_3 strutture a telaio con più piani e più campate

sap.spectrum_cst_4 strutture con solo due pareti non accoppiate per direzione orizzontale

sap.spectrum_cst_5 altre strutture a pareti non accoppiate

sap.spectrum_cst_6 strutture a pareti accoppiate o miste equivalenti a pareti

sap.spectrum_cst_7 strutture deformabili torsionalmente

sap.spectrum_cst_8 strutture a pendolo inverso

sap.spectrum_cst_9 strutture a pendolo inverso intelaiate monopiano

per le strutture in acciaio e miste calcestruzzo / acciaio:

sap.spectrum_sst_1 strutture a telaio di un piano

sap.spectrum_sst_2 strutture a telaio con più piani ed una sola campata

sap.spectrum_sst_3 strutture a telaio con più piani e più campate

sap.spectrum_sst_4 strutture con controventi eccentrici a un piano

sap.spectrum_sst_5 strutture con controventi eccentrici a più piani

sap.spectrum_sst_6 strutture con controventi concentrici a diagonale tesa attiva

sap.spectrum_sst_7 strutture con controventi concentrici a V

sap.spectrum_sst_8 strutture a mensola o a pendolo inverso

sap.spectrum_sst_9 strutture a telaio con controventi concentrici a un piano

sap.spectrum_sst_10 strutture a telaio con controventi concentrici a più piani ed una sola campata

sap.spectrum_sst_11 strutture a telaio con controventi concentrici a più piani e più campate

sap.spectrum_sst_12 strutture a telaio con tamponatura in murature

per le strutture in legno:

sap.spectrum_wst_1 pannelli a telaio chiodati con diaframmi incollati

sap.spectrum_wst_2 strutture reticolari iperstatiche con giunti chiodati

sap.spectrum_wst_3 portali iperstatici con mezzi di unione a gambo cilindrico

sap.spectrum_wst_4 pannelli a telaio chiodati con diaframmi chiodati

sap.spectrum_wst_5 pannelli di tavole incollate a strati incrociati

sap.spectrum_wst_6 strutture reticolari

sap.spectrum_wst_7 strutture miste con telaio in legno e tamponature non portanti

sap.spectrum_wst_8 strutture isostatiche in genere

per le strutture in muratura:

sap.spectrum_mst_1 costruzioni in muratura ordinaria

sap.spectrum_mst_2 costruzioni in muratura armata

sap.spectrum_mst_3 costruzioni in muratura armata con progettazione in capacità

sap.spectrum_mst_4 costruzioni in muratura confinata

sap.spectrum_mst_5 costruzioni in muratura confinata con progettazione in capacità

shape la morfologia dello spettro, uno tra i seguenti valori enumerati:

sap.spectrum_shape_inelastic anelastico

sap.spectrum_shape_elastic elastico

soil_cat la categoria del suolo, uno tra i seguenti valori enumerati:

sap.spectrum_soil_A Formazioni litoidi o suoli omogenei molto rigidi

sap.spectrum_soil_B Depositi di sabbie o ghiaie molto addensate o argille consistenti

sap.spectrum_soil_C Depositi di sabbie e ghiaie mediamente addensate o di argille di media consistenza

sap.spectrum_soil_D Depositi di terreni granulari da sciolti a poco addensati oppure poco coesivi da poco a mediamente consistenti

sap.spectrum_soil_E Profili di terreno costituiti da strati superficiali alluvionali

ductility_class la classe di duttilità, uno tra i seguenti valori enumerati:

sap.spectrum_duct_high alta

sap.spectrum_duct_low bassa

sap.spectrum_duct_none struttura non dissipativa

topographic_cat la categoria topografica, uno tra i seguenti valori enumerati:

sap.spectrum_topo_1 superficie pianeggiante, pendii e rilievi isolati con inclinazione media i = 15°

sap.spectrum_topo_2 pendii con inclinazione media i > 15°

sap.spectrum_topo_3 rilievi con larghezza in cresta molto minore che alla base e inclinazione media 15° ≤ i ≤ 30°

sap.spectrum_topo_4 rilievi con larghezza in cresta molto minore che alla base e inclinazione media i > 30°

perc_height_on_slope la percentuale di altezza sul pendio

regulation la normativa utilizzata, uno tra i seguenti valori enumerati:

sap.spectrum_reg_opcm Ordinanza del Presidente del Consiglio dei Ministri n.3274

sap.spectrum_reg_dm08 Decreto Ministeriale 2008

sap.spectrum_reg_dm18 Decreto Ministeriale 2018

reduced_seismic_zone **true** se AgS < 0.075g

damping il valore di smorzamento utilizzato

f0 l'amplificazione spettrale massima

ag il valore del parametro Ag(/g)

tc il periodo di inizio del tratto a velocità costante dello spettro in accelerazione orizzontale

longitude la longitudine di riferimento

latitude la latitudine di riferimento

pv la probabilità di superamento

vr la vita di riferimento

struct_desc la descrizione sintetica correlata alla tipologia della struttura

soil_desc	la descrizione sintetica correlata alla categoria del suolo
points	una tabella contenente i valori dello spettro. Ogni elemento della tabella è un array di due elementi, rispettivamente x e y del punto della funzione. È possibile passare direttamente questa tabella alla funzione **out.append_plot()** per ottenere il grafico dello spettro.

sap.get_metamaterials_names(type)

Restituisce una tabella che elenca i nomi dei metamateriali definiti nel file corrente. Il parametro *type* serve a limitare l'elenco al tipo specificato. Passando una stringa vuota si ottiene l'elenco completo.

Ad esempio il codice che segue stampa l'elenco dei materiali di tipo "Cemento armato":

```
names = sap.get_metamaterials_names("Cemento armato")

if names then
 for k, n in pairs(names) do
  out.append("\t\"" .. tostring(n) .. "\"\n")
 end
end
```

sap.get_metamaterial(name)

Restituisce una tabella che elenca le proprietà che definiscono il metamateriale.

Ad esempio il codice che segue stampa le proprietà del materiale "test":

```
props = sap.get_metamaterial("test")

for k, v in pairs(props) do
 out.append("\t\"" .. k .. "\":\t" .. "\"" .. tostring(v) .. "\"\n")
end
```

I nomi dei metamateriali definiti sono reperibili tramite la funzione **sap.get_metamaterials_names()**.

sap.get_mass_enabling( )

Restituisce una tabella che indica quali componenti delle masse sono abilitate per l'analisi corrente. La tabella restituita contiene tre campi, **x**, **y** e **z**, che hanno valore **true** se la componente relativa è abilitata, **false** altrimenti.

sap.get_modal_damping( )

Restituisce una tabella che descrive la partecipazione modale di ogni periodo calcolata nell'analisi dinamica corrente.

Ogni elemento della tabella è a sua volta una tabella con i seguenti campi definiti:

mode	indice del periodo

damping valore dello smorzamento modale

partecipation_x, **partecipation_y**, **partecipation_z** le tre componenti della partecipazione modale

sap.get_relative_mass( )

Restituisce una tabella che descrive le masse modali relative calcolate dall'analisi dinamica corrente.

Ogni elemento della tabella è a sua volta una tabella con i seguenti campi definiti:

mode l'indice del periodo

x, **y**, **z** le tre componenti della massa modale relativa

s la massa modale relativa totale

L'ultimo elemento della tabella raccoglie i totali degli elementi precedenti ed ha i seguenti campi:

mode è sempre "total"

sx, **sy**, **sz** le somme delle componenti delle voci precedenti

ss la somma delle masse modali relative totali delle voci precedenti

Si rimanda al manuale teorico di Nòlian per la spiegazione di questi valori.

sap.get_periods( )

Restituisce il numero di modi di vibrare calcolati dall'analisi dinamica corrente ed una tabella contenente i relativi autovalori

sap.get_displacements( )

Restituisce una tabella che descrive tutti i set di risultati di analisi di tipo "spostamenti" salvati nel file corrente.

Ogni valore della tabella è a sua volta una tabella che ha i seguenti campi:

key chiave del set, da utilizzare ad esempio in **sap.use_displacements**

name nome assegnato dall'utente al set

phase indice della fase a cui i risultati si riferiscono

comment commento assegnato dall'utente per descrivere i risultati

analysis_type tipo di analisi, uno tra i valori enumerati descritti in **sap.get_analyses**

size numero di risultati nel set

visible flag di visibilità del set impostato dall'utente

sap.get_stresses( )

Restituisce una tabella che descrive tutti i set di risultati di analisi di tipo "sforzi" salvati nel file corrente.

Ogni valore della tabella è a sua volta una tabella che ha i seguenti campi:

key chiave del set, da utilizzare ad esempio in **sap.use_stresses**

name nome assegnato dall'utente al set

phase indice della fase a cui i risultati si riferiscono

comment commento assegnato dall'utente per descrivere i risultati

analysis_type tipo di analisi, uno tra i valori enumerati descritti in **sap.get_analyses**

size numero di risultati nel set

visible flag di visibilità del set impostato dall'utente

sap.get_reactions( )

Restituisce una tabella che descrive tutti i set di risultati di analisi di tipo "reazioni" salvati nel file corrente.

Ogni valore della tabella è a sua volta una tabella che ha i seguenti campi:

key chiave del set, da utilizzare ad esempio in **sap.use_reactions**

name nome assegnato dall'utente al set

phase indice della fase a cui i risultati si riferiscono

comment commento assegnato dall'utente per descrivere i risultati

analysis_type tipo di analisi, uno tra i valori enumerati descritti in **sap.get_analyses**

size numero di risultati nel set

visible flag di visibilità del set impostato dall'utente

sap.get_buckling_results( )

Restituisce una tabella che descrive tutti i set di risultati di analisi di tipo "buckling" salvati nel file corrente.

Ogni valore della tabella è a sua volta una tabella che ha i seguenti campi:

key chiave del set, da utilizzare ad esempio in **sap.use_buckling**

name nome assegnato dall'utente al set

phase indice della fase a cui i risultati si riferiscono

comment commento assegnato dall'utente per descrivere i risultati

analysis_type tipo di analisi, uno tra i valori enumerati descritti in **sap.get_analyses**

size numero di risultati nel set

visible flag di visibilità del set impostato dall'utente

sap.get_capacity_results( )

Restituisce una tabella che descrive tutti i set di risultati di analisi di tipo "capacity" salvati nel file corrente.

Ogni valore della tabella è a sua volta una tabella che ha i seguenti campi:

key chiave del set, da utilizzare ad esempio in **sap.use_capacity**

name nome assegnato dall'utente al set

phase indice della fase a cui i risultati si riferiscono

comment commento assegnato dall'utente per descrivere i risultati

analysis_type tipo di analisi, uno tra i valori enumerati descritti in **sap.get_analyses**

size numero di risultati nel set

visible flag di visibilità del set impostato dall'utente

sap.get_nolian_options( )

Restituisce una tabella che descrive le opzioni di progetto impostate da Nòlian per il file corrente.

La tabella definisce le seguenti chiavi:

length_units unità di lunghezza impostata (m, cm o mm)

force_units unità di forza impostata (kg, N o kN)

sap.get_pp_load_cases( )

Restituisce una tabella che elenca le condizioni di carico nei post-processori salvate nel file corrente, insieme con il tipo di carico associato.

Ogni valore della tabella è a sua volta una tabella che ha i seguenti campi:

name nome della condizione, equivalente al set di risultati di tipo "sforzi" prodotto da Nòlian

load_type nome del tipo di carico assegnato dall'utente

sap.get_pp_geo_data( )

Restituisce le caratteristiche del terreno impostate nei postprocessori e salvate nel file corrente.

I parametri sono inseriti in una tabella che definisce i seguenti campi:

friction_angle angolo di attrito

cohesion coesione

wet_specific_gravity peso specifico

ground_water_depth profondità della falda

granules_specific_gravity densità del terreno umido

umidity percentuale di umidità

laying_surface_depth profondità del piano di posa

Coefficienti parziali per i parametri geotecnici del terreno (gruppo 'M'):

gamma_m_a correlato all'angolo di attrito

gamma_m_c correlato alla coesione

gamma_m_d correlato alla densità

Coefficienti parziali per verifiche SLU (gruppo 'R'):

gamma_r_p capacità portante

gamma_r_s scorrimento

Coefficienti parziali per le azioni (gruppo 'A'):

gamma_g_1 permanenti

gamma_q_i variabili

sap.get_beam_options( )

Restituisce i parametri di progetto impostati in EasyBeam e salvati nel file corrente.

I parametri sono inseriti in una tabella che definisce i seguenti campi:

length_units unità di lunghezza impostata (m, cm o mm)

force_units unità di forza impostata (kg, N o kN)

pressure_units unità di pressione impostata (ad es. "kg/cm2")

concrete_design_strength resistenza caratteristica del calcestruzzo

tau_limit_0 limite per la tensione tangenziale senza armatura

tau_limit_1 parte del taglio affidato ad altri fenomeni

tau_limit_2 limite superiore per la tensione tangenziale

regulation normativa impiegata, uno tra i seguenti valori enumerati:

- **sap.eb_reg_none** nessuna
- **sap.eb_reg_default** default
- **sap.eb_reg_dm92_ta** Decreto Ministeriale 1992 - Tensioni ammissibili
- **sap.eb_reg_dm96_slu** Decreto Ministeriale 1996 - Stati limite
- **sap.eb_reg_opcm_a** Ordinanza del Presidente del Consiglio dei Ministri 3274 - Alta duttilità
- **sap.eb_reg_opcm_b** Ordinanza del Presidente del Consiglio dei Ministri 3274 - Bassa duttilità
- **sap.eb_reg_dm08_ns** Decreto Ministeriale 2008 - Zona non sismica
- **sap.eb_reg_dm08_a** Decreto Ministeriale 2008 - Alta duttilità
- **sap.eb_reg_dm08_b** Decreto Ministeriale 2008 - Bassa duttilità
- **sap.eb_reg_dm08_b_4** Decreto Ministeriale 2008 - Bassa duttilità, solo particolari costruttivi (zona sismica 4)
- **sap.eb_reg_dm18_nd** Decreto Ministeriale 2018 - Comportamento non dissipativo
- **sap.eb_reg_dm18_a** Decreto Ministeriale 2018 - Alta duttilità
- **sap.eb_reg_dm18_b** Decreto Ministeriale 2018 - Bassa duttilità
- **sap.eb_reg_dm18_c** Decreto Ministeriale 2018 - agS < 0.075g

method metodo di progetto e verifica utilizzato, uno tra i seguenti valori enumerati:

sap.eb_method_TA tensioni ammissibili

sap.eb_method_SL stati limite

In caso di progetto alle tensioni ammissibili, la tabella conterrà anche i seguenti campi:

concrete_allowable_stress tensione ammissibile nel calcestruzzo

steel_allowable_stress tensione ammissibile nell'acciaio

elasticity_moduli_ratio rapporto tra i moduli elastici

In caso di progetto agli stati limite, la tabella conterrà anche i seguenti campi:

concrete_partial_safety_factor coefficiente di sicurezza parziale del calcestruzzo

concrete_additional_safety_factor coefficiente di sicurezza addizionale del calcestruzzo

concrete_max_comp_strain accorciamento unitario massimo nel calcestruzzo

steel_design_strength resistenza caratteristica dell'acciaio

steel_partial_safety_factor coefficiente di sicurezza parziale dell'acciaio

steel_max_tens_strain allungamento unitario massimo nell'acciaio

axial_strength_reduction_factor fattore di riduzione della resistenza assiale

flexural_strength_reduction_factor fattore di riduzione della resistenza flessionale

shear_strength_reduction_factor fattore di riduzione della resistenza a taglio

srv_rare **true** se vengono effettuate combinazioni di servizio rare

srv_frequent **true** se vengono effettuate combinazioni di servizio frequenti

srv_almost_perm **true** se vengono effettuate combinazioni di servizio quasi permanenti

env_aggressiveness tipo di aggressività chimica dell'ambiente, uno tra i seguenti valori enumerati:

sap.eb_envagg_slightly poco aggressivo

sap.eb_envagg_fairly moderatamente aggressivo

sap.eb_envagg_highly molto aggressivo

sap.eb_envagg_user definito dall'utente

Si noti che i parametri **tau_limit_*** dipendono dal metodo di progetto utilizzato.

Se i parametri di combinazione per EasyBeam non sono definiti, viene restituito il valore **nil**.

sap.get_steel_options( )

Restituisce i parametri di progetto impostati in EasySteel e salvati nel file corrente.

I parametri sono inseriti in una tabella che definisce i seguenti campi:

length_units unità di lunghezza impostata (m, cm o mm)

force_units unità di forza impostata (kg, N o kN)

pressure_units unità di pressione impostata (ad es. "kg/cm2")

steel_class classe di resistenza dell'acciaio, uno tra i seguenti valori enumerati:

sap.es_steel_class_235

sap.es_steel_class_275

sap.es_steel_class_355

sap.es_steel_class_420

sap.es_steel_class_450

sap.es_steel_class_460

sap.es_steel_class_User

steel_strength	resistenza di snervamento dell'acciaio
steel_collapse	resistenza di rottura dell'acciaio
regulation	normativa impiegata, uno tra i seguenti valori enumerati:
sap.es_reg_dm08_ns	Decreto Ministeriale 2008 - Zona non sismica
sap.es_reg_dm08_b	Decreto Ministeriale 2008 - Bassa duttilità
sap.es_reg_dm08_a	Decreto Ministeriale 2008 - Alta duttilità
sap.es_reg_dm18_nd	Decreto Ministeriale 2018 - Comportamento non dissipativo
sap.es_reg_dm18_b	Decreto Ministeriale 2018 - Bassa duttilità
sap.es_reg_dm18_a	Decreto Ministeriale 2018 - Alta duttilità
partial_safety_factor_gm0	fattore di sicurezza parziale, resistenza delle sezioni di classe 1,2,3,4
partial_safety_factor_gm1	fattore di sicurezza parziale, resistenza all'instabilità delle membrature
partial_safety_factor_gm2	fattore di sicurezza parziale, resistenza nei riguardi della frattura delle sezioni tese indebolite da fori
over_strength_factor	fattore di sovraresistenza
global_ductility_factor	fattore di duttilità dell'intera struttura
steel_spec_weight	peso specifico dell'acciaio
steel_thermal_expansion_factor	coefficiente di dilatazione termica dell'acciaio

Se i parametri di combinazione per EasySteel non sono definiti, viene restituito il valore **nil**.

sap.get_wall_options( )

Restituisce i parametri di progetto impostati in EasyWall e salvati nel file corrente.

I parametri sono inseriti in una tabella che definisce i seguenti campi:

length_units unità di lunghezza impostata (m, cm o mm)

force_units unità di forza impostata (kg, N o kN)

pressure_units unità di pressione impostata (ad es. "kg/cm2")

regulation normativa impiegata, uno tra i seguenti valori enumerati:

- **sap.eb_reg_none** nessuna
- **sap.eb_reg_dm08_ns** Decreto Ministeriale 2008 - Zona non sismica
- **sap.eb_reg_dm08_a** Decreto Ministeriale 2008 - Alta duttilità
- **sap.eb_reg_dm08_b** Decreto Ministeriale 2008 - Bassa duttilità
- **sap.eb_reg_dm08_b_4** Decreto Ministeriale 2008 - Bassa duttilità, solo particolari costruttivi (zona sismica 4)
- **sap.eb_reg_dm18_nd** Decreto Ministeriale 2018 - Comportamento non dissipativo
- **sap.eb_reg_dm18_a** Decreto Ministeriale 2018 - Alta duttilità
- **sap.eb_reg_dm18_b** Decreto Ministeriale 2018 - Bassa duttilità
- **sap.eb_reg_dm18_c** Decreto Ministeriale 2018 - agS < 0.075g

method metodo di progetto e verifica utilizzato ("stati limite" o "tensioni ammissibili")

concrete_design_strength resistenza caratteristica del calcestruzzo

concrete_allowable_stress (solo per t.a.) tensione ammissibile calcestruzzo

steel_allowable_stress (solo per t.a.) tensione ammissibile acciaio

elasticity_moduli_ratio (solo per t.a.) rapporto moduli elastici

concrete_partial_safety_factor (solo per s.l.) coefficiente di sicurezza parziale calcestruzzo

concrete_additional_safety_factor	(solo per s.l.) coefficiente di sicurezza addizionale calcestruzzo
concrete_max_comp_strain	(solo per s.l.) accorciamento unitario massimo nel calcestruzzo
steel_design_strength	(solo per s.l.) resistenza caratteristica acciaio
steel_partial_safety_factor	(solo per s.l.) coefficiente di sicurezza parziale acciaio
steel_max_tens_strain	(solo per s.l.) allungamento unitario massimo nell'acciaio
axial_strength_reduction_factor	(solo per s.l.) fattore di riduzione resistenza assiale
flexural_strength_reduction_factor	(solo per s.l.) fattore di riduzione resistenza flessionale
shear_strength_reduction_factor	(solo per s.l.) fattore di riduzione resistenza a taglio
srv_rare	(solo per s.l.) **true** se vengono effettuate combinazioni di servizio rare
srv_frequent	(solo per s.l.) **true** se vengono effettuate combinazioni di servizio frequenti
srv_almost_perm	(solo per s.l.) **true** se vengono effettuate combinazioni di servizio quasi permanenti

Se i parametri di combinazione per EasyWall non sono definiti, viene restituito il valore **nil**.

sap.get_elwood_options( )

Restituisce i parametri di progetto impostati in Elwood e salvati nel file corrente.

I parametri sono inseriti in una tabella che definisce i seguenti campi:

length_units	unità di lunghezza impostata (m, cm o mm)
force_units	unità di forza impostata (kg, N o kN)
pressure_units	unità di pressione impostata (ad es. "kg/cm2")
steel_strength	resistenza a snervamento dell'acciaio
steel_collapse	resistenza a rottura dell'acciaio
steel_partial_safety_factor	coefficiente di sicurezza parziale dell'acciaio

wood_Kind tipo del materiale, uno tra i seguenti valori enumerati:

sap.ew_wood_solid

sap.ew_wood_laminated

wood_ServClass classe di servizio, uno tra i seguenti valori enumerati:

sap.ew_servclass_1

sap.ew_servclass_2

sap.ew_servclass_3

wood_fmk resistenza a flessione

wood_ft0k resistenza a trazione parallela alle fibre

wood_ft90k resistenza a trazione ortogonale alle fibre

wood_fc0k resistenza a compressione parallela alle fibre

wood_fc90k resistenza a compressione parallela alle fibre

wood_fvk resistenza a taglio

wood_gm cofficiente di sicurezza parziale

wood_gmj coefficiente sicurezza parziale per i giunti

wood_rho massa volumica caratteristica

wood_Emean modulo elastico medio

wood_E90m modulo elastico medio perpendicolare alle fibre

wood_E005 modulo elastico caratteristico

wood_Gmean modulo tangenziale caratteristico

wood_kdef coefficiente di deformabilità

wood_kmod_perm coefficiente di correzione per effetti della durata del carico e dell'umidità permanenti

wood_kmod_long coefficiente di correzione per effetti della durata del carico e dell'umidità lunghi

wood_kmod_medium coefficiente di correzione per effetti della durata del carico e dell'umidità medi

wood_kmod_short coefficiente di correzione per effetti della durata del carico e dell'umidità brevi

wood_kmod_instant coefficiente di correzione per effetti della durata del carico e dell'umidità istantanei

parametri per mezzi di unione di tipo **"chiodo cilindrico"**:

wood_ut_cnail_diameter diametro

wood_ut_cnail_fy resistenza di snervamento dell'acciaio

wood_ut_cnail_predrilled preforato

wood_ut_cnail_extr_perc percentuale di estrazione

wood_ut_cnail_depth profondità di inserimento

wood_ut_cnail_drill_tol tolleranza dei fori

wood_ut_cnail_wood_species essenza, uno tra i seguenti valori enumerati:

sap.ew_species_coniferous conifere

sap.ew_species_lvl LVL

sap.ew_species_deciduous latifoglie

wood_ut_cnail_extr_res resistenza caratteristica ad estrazione

parametri per mezzi di unione di tipo **"chiodo scanalato"**:

wood_ut_qnail_diameter diametro

wood_ut_qnail_fy resistenza di snervamento dell'acciaio

wood_ut_qnail_predrilled preforato

wood_ut_qnail_extr_perc percentuale di estrazione

wood_ut_qnail_depth profondità di inserimento

wood_ut_qnail_drill_tol tolleranza dei fori

wood_ut_qnail_wood_species essenza, uno tra i seguenti valori enumerati:

sap.ew_species_coniferous conifere

sap.ew_species_lvl LVL

sap.ew_species_deciduous latifoglie

wood_ut_qnail_extr_res resistenza caratteristica ad estrazione

parametri per mezzi di unione di tipo **"bullone"**:

wood_ut_bolt_diameter diametro

wood_ut_bolt_fy resistenza di snervamento dell'acciaio

wood_ut_bolt_depth profondità di inserimento

wood_ut_bolt_drill_tol tolleranza dei fori

wood_ut_bolt_wood_species essenza, uno tra i seguenti valori enumerati:

sap.ew_species_coniferous conifere

sap.ew_species_lvl LVL

sap.ew_species_deciduous latifoglie

wood_ut_bolt_extr_res resistenza caratteristica ad estrazione

parametri per mezzi di unione di tipo **"spinotto"**:

wood_ut_spine_diameter diametro

wood_ut_spine_fy resistenza di snervamento dell'acciaio

wood_ut_spine_depth profondità di inserimento

wood_ut_spine_drill_tol tolleranza dei fori

wood_ut_spine_extr_res resistenza caratteristica ad estrazione

parametri per mezzi di unione di tipo **"vite"**:

wood_ut_screw_diameter diametro

wood_ut_screw_fy resistenza di snervamento dell'acciaio

wood_ut_screw_depth profondità di inserimento

wood_ut_screw_drill_tol tolleranza dei fori

wood_ut_screw_extr_res resistenza caratteristica ad estrazione

sap.get_exsys_options( )

Restituisce i parametri di verifica impostati in ExSys e salvati nel file corrente.

I parametri sono inseriti in una tabella che definisce i seguenti campi:

analysis_type tipo di analisi impiegato, uno tra i seguenti valori enumerati:

sap.ex_design_spectrum analisi con spettro di progetto

sap.ex_elastic_spectrum analisi con spettro elastico

sap.ex_static_nl_analysis analisi statica non lineare

confidence_factor fattore di confidenza

q fattore di struttura

load_cond_fragile nome della condizione di carico per la verifica di fragilità

load_cond_ductile nome della condizione di carico per la verifica di duttilità

Se i parametri di combinazione per ExSys non sono definiti, viene restituito il valore **nil**.

sap.get_wallver_options( )

Restituisce i parametri di verifica impostati in WallVerine e salvati nel file corrente.

I parametri sono inseriti in una tabella che definisce i seguenti campi:

ignore_oop_moments 'true' se non considera i momenti al di fuori del piano

is_reinforced 'true' se verifica come muratura armata

is_existing 'true' se verifica con metodi per strutture esistenti

confidence_factor fattore di confidenza

masonry_fk muratura: resistenza caratteristica a compressione

masonry_fvk0 muratura: resistenza caratteristica a taglio

masonry_partial_safety_factor muratura: coefficiente parziale di sicurezza

friction_angle fondazioni: angolo di attrito interno

cohesion fondazioni: coesione del terreno

wet_specific_gravity fondazioni: densità del terreno

laying_surface_depth	fondazioni: profondità del piano di posa
steel_design_strength	armatura: tensione di snervamento di progetto dell'acciaio
steel_lb_area	armatura: area delle barre longitudinali
steel_lb_step	armatura: passo delle barre longitudinali
steel_tb_area	armatura: area delle barre trasversali
steel_tb_step	armatura: passo delle barre trasversali

Se i parametri di combinazione per WallVerine non sono definiti, viene restituito il valore **nil**.

sap.get_frp_options( )

Restituisce i parametri di verifica impostati in FibRePower e salvati nel file corrente.

I parametri sono inseriti in una tabella che definisce i seguenti campi:

using_frp	true se si usano FRP, false se usano cerchiature in acciaio
elems_elast_mod	elementi: modulo di elasticità
elems_ultimate_eps	elementi: deformazione di rottura
elems_yield_stress	elementi: tensione di rottura
elems_post_yield_elasticity	elementi: hardening acciaio
elems_gamma_f	elementi: coefficiente di sicurezza a trazione
elems_gamma_rd	elementi: coefficiente di sicurezza a delaminazione
elems_eta_a	elementi: fattore di conversione ambientale
elems_eta_l	elementi: fattore di conversione per modalità di carico
elems_rounding_radius	elementi: raggio arrotondamento spigoli
nodes_yield_stress	nodi: tensione di rottura

nodes_walls_shear_strenght nodi: resistenza a taglio tamponatura

nodes_walls_compr_strenght nodi: resistenza a compressione tamponatura

Se i parametri di combinazione per FibRePower non sono definiti, viene restituito il valore **nil**.

Interazione con Nòlian

Il motore di scripting mette a disposizione alcune funzioni di interrogazione diretta a Nòlian per dati e risultati che non possono essere letti direttamente da file attraverso le funzioni della libreria **sap** descritta nel capitolo precedente.

Le funzioni della libreria **nolian** qui descritta si basano su uno stato di computazione modificato attraverso le funzioni di tipo **use_*** ed interrogato attraverso le funzioni **get_***. Ved. il capitolo precedente "Funzioni di accesso ai file SAP" per maggiori informazioni.

nolian.get_app_info( )

Restituisce una tabella con informazioni sintetiche sull'applicazione. La tabella ha i seguenti campi:

manufacturer il nome del produttore, Softing s.r.l.

name il nome dell'applicazione

version la versione dell'applicazione

license il numero di licenza dell'utente

nolian.use_phase(k)

Imposta la fase corrente a quella specificata. Il parametro *k* può essere un nome o una chiave numerica tra quelli restituiti dalla funzione **sap.get_phases**.

nolian.use_node(i)

Imposta il nodo corrente per le funzioni successive a quello avente indice *i*.

nolian.use_element(i)

Imposta l'elemento corrente per le funzioni successive a quello avente indice *i*.

nolian.use_load_case(s)

Imposta la condizione di carico per le funzioni successive a quella avente il nome specificato.

nolian.get_nodes_count( )

Restituisce il numero di nodi attivi nello stato corrente.

nolian.get_elems_count( )

Restituisce il numero di elementi attivi nello stato corrente.

nolian.get_elem_types( )

Restituisce il numero di elementi attivi divisi per tipo di elemento e per geometria.

Vengono restituite due tabelle. La prima tabella ha i seguenti campi:

truss	numero di elementi asta
beam	numero di elementi trave generica
rectbeam	numero di elementi trave rettangolare
winklerbeam	numero di elementi trave di fondazione
polybeam	numero di elementi trave poligonale
boundary	numero di elementi boundary
rigel	numero di elementi di connessione rigida
shell	numero di elementi guscio
winklershell	numero di elementi guscio di fondazione
axial	numero di elementi assialsimmetrici
planestrain	numero di elementi a deformazione piana
brick	numero di elementi brick

La seconda tabella ha i seguenti campi:

mono numero di elementi monodimensionali

plane_3, **plane_4**, **plane_8** numero di piani a tre, quattro e otto nodi

brick numero di elementi solidi

Ad esempio per stampare il numero di elementi monodimensionali di una struttura, può essere utilizzato il codice:

```
a, b = nolian.get_elem_types()
out.append(b.mono .. " elementi monodimensionali")
```

nolian.get_load_types_names( )

Restituisce una tabella che elenca i tipi di carico definiti in Nolian.

nolian.get_load_type_data(n)

Restituisce le impostazioni relative al tipo di carico con il nome richiesto.

Viene restituita una tabella avente i seguenti campi:

type tipo assegnato dall'utente; uno tra i seguenti valori enumerati:

- **nolian.loadtype_permanent** permanente
- **nolian.loadtype_variable** variabile
- **nolian.loadtype_not_concurrent_variable** variabile non contemporaneo
- **nolian.loadtype_seismic** sismico
- **nolian.loadtype_seismic_related** simico correlato

gravitational **true** se deriva da peso proprio

gamma_unf valore del coefficiente gamma

gamma_seismic_unf valore del coefficiente gamma sismico

psi_0 valore del coefficiente psi zero

psi_1 valore del coefficiente psi uno

psi_2 valore del coefficiente psi due

psi_2_seismic valore del coefficiente psi due sismico

phi valore del coefficiente di correlazione

nolian.get_displacement( )

Restituisce gli spostamenti relativi al nodo (ved. **nolian.use_node**) ed al set di spostamenti (ved. **sap.use_displacements**) correnti.

Viene restituita una tabella avente i seguenti campi:

tx, **ty**, **tz** componenti della traslazione del nodo

rx, **ry**, **rz** componenti della rotazione del nodo

nolian.get_stress( )

Restituisce gli sforzi relativi all'elemento (ved. **nolian.use_element**) ed al set di sforzi (ved. **sap.use_stresses**) correnti.

Viene restituita un array di sforzi avente tanti elementi quanti i nodi dell'elemento corrente. Ogni elemento dell'array è una tabella che definisce valori dipendenti dal tipo di elemento.

Per gli elementi monodimensionali la tabella restituita ha i seguenti campi:

N, **Vy**, **Vz** sforzo assiale e sforzi di taglio

Mx, **My**, **Mz** momento torcente e momenti flettenti

Per gli elementi piani la tabella restituita ha i seguenti campi:

Nx, **Ny**, **Nxy** sforzi membranali: tensioni piane e tensione tangenziale

Nz (solo assialsimmetrici) tensione circonferenziale

Mx, **My**, **Mxy** (tranne assialsimmetrici) momenti flettenti e momento torcente

Per gli elementi solidi la tabella restituita ha i seguenti campi:

Sx, **Sy**, **Sz**, **Sxy**, **Sxz**, **Syz** componenti del tensore tridimensionale di sforzo.

nolian.get_reaction( )

Restituisce le reazioni relative al nodo (ved. **nolian.use_node**) ed al set di reazioni (ved. **sap.use_reactions**) correnti.

Viene restituita una tabella avente i seguenti campi:

tx, **ty**, **tz** componenti della reazione alla traslazione del nodo

rx, **ry**, **rz** componenti della reazione alla rotazione del nodo

nolian.get_eigenvector( )

Restituisce l'autovettore relativo al nodo (ved. **nolian.use_node**) ed al modo di vibrare (ved. **sap.use_mode**) correnti.

Viene restituita una tabella avente i seguenti campi:

tx, **ty**, **tz** componenti della traslazione del nodo

rx, **ry**, **rz** componenti della rotazione del nodo

nolian.get_buckling( )

Restituisce il risultato dell'analisi buckling corrente (ved. **sap.use_buckling**).

Viene restituita una tabella avente i seguenti campi:

critical_factor moltiplicatore critico

nolian.get_capacity( )

Restituisce il risultato dell'analisi capacity corrente (ved. **sap.use_capacity**).

Viene restituita una tabella avente i seguenti campi:

G coeff. gamma

m	massa modale
fy	forza di snervamento del sistema SDOF
dy	spostamento di snervamento del sistema SDOF
t	periodo proprio del sistema SDOF
dcu	spostamento di controllo della struttura
tc	periodo TC dello spettro di risposta
set	ordinata dello spettro elastico al periodo T*
dmax	spostamento massimo del sistema MDOF

Dati della sezione in cui si verifica la prima plasticizzazione (se presente):

y_elem_index	indice dell'elemento in cui si verifica
y_displacement	spostamento
y_abscissa	ascissa
y_curvature	curvatura
y_rotation	rotazione
y_pl_hinge_length	lunghezza cerniera plastica
y_pga	peak ground acceleration

Dati della sezione in cui si verifica il primo danno grave (se presente):

d_elem_index	indice dell'elemento in cui si verifica
d_displacement	spostamento
d_abscissa	ascissa

d_curvature	curvatura
d_rotation	rotazione
d_pl_hinge_length	lunghezza cerniera plastica
d_pga	peak ground acceleration

Dati della sezione in cui si verifica il collasso (se presente):

c_elem_index	indice dell'elemento in cui si verifica
c_displacement	spostamento
c_abscissa	ascissa
c_curvature	curvatura
c_rotation	rotazione
c_pl_hinge_length	lunghezza cerniera plastica
c_pga	peak ground acceleration

nolian.get_ForFun(c)

Restituisce una tabella a due colonne con i valori dell'accelerogramma (c=1) o dello spettro (c=0) come definiti nel dialogo Forzanti.

get_num_of_critical_points(n)

Restituisce il numero degli stati critici e degli stati limite di capacità di spostamento per l'analisi di nome **n**.

Ad esempio:

```
a, b = nolian.get_num_of_critical_points("Analisi 1")
out.append("Stati critici: " .. a .. "\n")
out.append("Stati limite: " .. b .. "\n")
```

nolian.get_critical_state(n , i)

Restituisce i dati relativi allo stato critico di indice **i** per l'analisi di nome **n**. Viene restituito il valore **nil** se non esiste uno stato critico identificabile da tali parametri.

Viene restituita una tabella avente i seguenti campi:

critical_state tipo di stato critico, uno tra i seguenti valori:

- **nolian.crst_undefined** indefinito
- **nolian.crst_first_core_crunch** rottura a compressione nel calcestruzzo
- **nolian.crst_first_steel_yield** snervamento dell'acciaio a trazione
- **nolian.crst_shear_failure** superamento della resistenza a taglio della sezione
- **nolian.crst_rotation_failure** superamento della capacità di rotazione della sezione
- **nolian.crst_drift_failure** superamento del valore limite del drift di piano
- **nolian.crst_node_failure** superamento della resistenza dei nodi di telaio
- **nolian.crst_steel_breaking** superamento deformazione 1% in trazione acciaio
- **nolian.crst_contact_failure** raggiungimento del cinematismo del contatto

message descrizione dello stato critico

displacement spostamento

pga_vd PGA demand, per SLV

pga_vc PGA capacity, per SLV

tdr_vd tempo di ritorno demand, per SLV

tdr_vc tempo di ritorno capacity, per SLV

pga_dd PGA demand, per SLD

pga_dc PGA capacity, per SLD

tdr_dd tempo di ritorno demand, per SLD

tdr_dc tempo di ritorno capacity, per SLD

pga_od PGA demand, per SLO

pga_oc PGA capacity, per SLO

tdr_od tempo di ritorno demand, per SLO

tdr_oc tempo di ritorno capacity, per SLO

pga_cd PGA demand, per SLC

pga_cc PGA capacity, per SLC

tdr_cd tempo di ritorno demand, per SLC

tdr_cc tempo di ritorno capacity, per SLC

spectrum_type tipo di spettro, uno tra i seguenti valori:

- **sap.spectrum_type_generic** spettro non riconducibile ad un impiego specifico
- **sap.spectrum_type_slo** spettro per la valutazione dello stato limite di operatività
- **sap.spectrum_type_sld** spettro per la valutazione dello stato limite di danno
- **sap.spectrum_type_slv** spettro per la valutazione dello stato limite di salvaguardia della vita
- **sap.spectrum_type_slc** spettro per la valutazione dello stato limite di collasso

spectrum_name nome dello spettro

action_name nome dell'azione di riferimento

elem_index indice dell'elemento in cui si è prodotto lo stato critico

abscissa ascissa della sezione lungo l'elemento in cui si è prodotto lo stato critico

nolian.get_critical_capacity_state(n , i)

Restituisce i dati relativi allo stato limite di capacità di spostamento di indice **i** per l'analisi di nome **n**. Viene restituito il valor
nil se non esiste uno stato limite identificabile da tali parametri.

Viene restituita una tabella avente i seguenti campi:

capacity_state tipo di stato limite, uno tra i seguenti valori:

- **sap.capdsp_type_none** indefinito
- **sap.capdsp_type_sle** snervamento
- **sap.capdsp_type_slv** danno severo
- **sap.capdsp_type_slc** collasso

message descrizione dello stato critico

action_name nome dell'azione di riferimento

displacement spostamento

pga_vd PGA demand, per SLV

pga_vc PGA capacity, per SLV

tdr_vd tempo di ritorno demand, per SLV

tdr_vc tempo di ritorno capacity, per SLV

pga_dd PGA demand, per SLD

pga_dc PGA capacity, per SLD

tdr_dd tempo di ritorno demand, per SLD

tdr_dc tempo di ritorno capacity, per SLD

pga_od PGA demand, per SLO

pga_oc PGA capacity, per SLO

tdr_od tempo di ritorno demand, per SLO

tdr_oc tempo di ritorno capacity, per SLO

pga_cd PGA demand, per SLC

pga_cc PGA capacity, per SLC

tdr_cd tempo di ritorno demand, per SLC

tdr_cc tempo di ritorno capacity, per SLC

nolian.get_node_force( )

Restituisce i valori della forza applicata al nodo corrente (ved. **nolian.use_node**) per la condizione di carico corrente (ved. **nolian.use_load_case**).

Viene restituita una tabella avente le componenti della forza rispettivamente nei campi **fx, fy, fz, mx, my** ed **mz**. Viene restituito il valore **nil se il nodo non ha nessuna forza applicata.**

nolian.get_node_mass( )

Restituisce i valori della massa applicata al nodo corrente (ved. **nolian.use_node**) per la condizione di carico corrente (ved. **nolian.use_load_case**).

Viene restituita una tabella avente le componenti della massa rispettivamente nei campi **mx, my, mz, jx, jy** ed **jz**. Viene restituito il valore **nil se il nodo non ha nessuna forza applicata.**

nolian.get_node_data( )

Restituisce alcuni dati del nodo corrente (ved. **nolian.use_node**).

Viene restituita una tabella con i seguenti campi:

x, **y**, **z** coordinate del nodo

restraint_tx, **restraint_ty**, **restraint_tz**	vincoli sulle traslazioni
restraint_rx, **restraint_ry**, **restraint_rz**	vincoli sulle rotazioni
master_index	indice del nodo master

nolian.get_element_load( )

Restituisce i valori del carico applicato all'elemento corrente (ved. **nolian.use_element**) per la condizione di carico corrente (ved. **nolian.use_load_case**).

Viene restituito il valore **nil** se l'elemento non ha nessun carico applicato. In caso contrario viene restituita una tabella avent un campo per ogni componente di carico:

gen_tx_1, **gen_ty_1**, **gen_tz_1**	(monodimensionali) componente assiale, di taglio y e di taglio z del carico generico sul primo nodo
gen_rx_1, **gen_ry_1**, **gen_rz_1**	(monodimensionali) componente del momento torcente, y e z del carico generico sul primo nodo
gen_tx_2, **gen_ty_2**, **gen_tz_2**	(monodimensionali) componente assiale, di taglio y e di taglio z del carico generico sul secondo nodo
gen_rx_2, **gen_ry_2**, **gen_rz_2**	(monodimensionali) componente del momento torcente, y e z del carico generico sul secondo nodo
unif_x, **unif_y**, **unif_z**	componenti x, y e z del carico uniforme
unif_ref	riferimento per il carico uniforme: globale (0), locale (1) o globale proiettato (2)
unif_brick_face	(brick) indice da 1 a 6 della faccia del brick a cui è applicato il carico uniforme
edge_it, **edge_in**, **edge_ip**	(bidimensionali) componenti del carico di bordo per il primo spigolo dell'elemento
edge_jt, **edge_jn**, **edge_jp**	(bidimensionali) componenti del carico di bordo per il secondo spigolo dell'elemento
edge_kt, **edge_kn**, **edge_kp**	(bidimensionali) componenti del carico di bordo per il terzo spigolo dell'elemento
edge_lt, **edge_ln**, **edge_lp**	(bidimensionali) componenti del carico di bordo per il quarto spigolo dell'elemento
edge_ref	(bidimensionali) riferimento per il carico di bordo: globale (0) o locale (1)

trp_x_1, **trp_y_1**, **trp_z_1**	(monodimensionali) componenti del carico trapezio applicato al primo nodo
trp_x_2, **trp_y_2**, **trp_z_2**	(monodimensionali) componenti del carico trapezio applicato al secondo nodo
trp_ref	(monodimensionali) riferimento per il carico trapezio: globale (0) o locale (1)
thm_exp_coeff	coefficiente di dilatazione per il carico termico
thm_dt_unif, **thm_dt_linear**	delta t costante e lineare per il carico termico
dead_spec_w	peso specifico per il carico da peso proprio
dead_x, **dead_y**, **dead_z**	componenti dell'accelerazione di gravità per il carico da peso proprio
bnd_nx, **bnd_ny**, **bnd_nz**	(boundary) componenti della contrazione assiale dello spostamento imposto
bnd_rx, **bnd_ry**, **bnd_rz**	(boundary) componenti della rotazione sinistrorsa dello spostamento imposto

nolian.get_element_mass( )

Restituisce i valori della massa applicata all'elemento corrente (ved. **nolian.use_element**) per la condizione di carico corrente (ved. **nolian.use_load_case**).

Viene restituita una tabella avente i seguenti campi:

density	densità di massa
distributed	massa distribuita
damping	smorzamento
locked	flag pari a **true** se l'utente ha impostato che la massa non deve essere cambiata dal calcolo automatico della conversione dei carichi in massa

Viene restituito il valore **nil** se l'elemento non ha nessuna forza applicata.

nolian.get_element_data( )

Restituisce alcuni dati dell'elemento corrente (ved. **nolian.use_element**).

Viene restituita una tabella con i seguenti campi:

node_count numero di nodi che formano l'elemento

da **node_1** a **node_8** indici dei nodi che formano l'elemento; un indice nullo indica nessun nodo

knode indice del nodo k per l'elemento

layer nome del layer di cui l'elemento fa parte

nolian.get_element_type( )

Restituisce i dati assegnati all'elemento corrente (ved. **nolian.use_element**).

Viene restituita una tabella il cui primo campo, **type**, indica il tipo dell'elemento; gli altri campi dipendono dal tipo stesso. Il tipo è uno tra i seguenti valori enumerati:

elemtype_truss elemento asta

gli altri campi sono:

elast_mod modulo di elasticità

area area della sezione

pretension pretensione

dati del profilo, se assegnato (ved. sotto);

elemtype_beam elemento trave generica

gli altri campi sono:

elast_mod modulo di elasticità

shear_mod modulo di elasticità tangenziale

area area della sezione

shear_area_y e **shear_area_z** componenti y e z dell'area a taglio

inertia_x, **inertia_y**, **inertia_z** componenti dell'inerzia torsionale, Y e Z

i dati dei vincoli interni (ved. sotto);

i dati del profilo, se assegnato (ved. sotto);

elemtype_rectBeam elemento trave rettangolare

gli altri campi sono:

elast_mod modulo di elasticità

shear_mod modulo di elasticità tangenziale

web_depth e **web_thickness** altezza e larghezza dell'anima

up_flange_w e **up_flange_tk** larghezza e spessore dell'ala superiore

lo_flange_w e **lo_flange_tk** larghezza e spessore dell'ala inferiore

i dati dei vincoli interni (ved. sotto);

elemtype_winklerBeam elemento trave di fondazione

gli altri campi sono:

elast_mod modulo di elasticità

shear_mod modulo di elasticità tangenziale

subgrade_mod coefficiente di sottofondo

width, depth base e altezza della trave

flange_tk, web_tk spessore della suola e dell'anima

width_on_soil base sul suolo

elemtype_polyBeam	elemento trave poligonale

gli altri campi sono:

elast_mod	modulo di elasticità
shear_mod	modulo di elasticità tangenziale
vtx_count	numero di vertici definiti
coppie **zN**, **yN**, con N da 1 a **vtx_count**	coordinate dei vertici del profilo poligonale

i dati dei vincoli interni (ved. sotto);

elemtype_boundary	elemento boundary

gli altri campi sono:

nx, ny, nz	rigidezze assiali rispetto agli assi del riferimento locale
rx, ry, rz	rigidezze torsionali rispetto agli assi del riferimento locale

elemtype_rigel	elemento di connessione rigida; nessun altro campo definito

elemtype_shell	elemento guscio

gli altri campi sono:

elast_mod	modulo di elasticità
poisson	coefficiente di Poisson
thickness	spessore
flex_elast_coef	moltiplicatore di elasticità flessionale
thickness	spessore

flex_ortho_mult moltiplicatore ortotropia Y flessionale

memb_ortho_mult moltiplicatore ortotropia Y membranale

cyl_hinge_on_side_1 flag booleano inserimento cerniera cilindrica su lato 1

cyl_hinge_on_side_2 flag booleano inserimento cerniera cilindrica su lato 2

cyl_hinge_on_side_3 flag booleano inserimento cerniera cilindrica su lato 3

cyl_hinge_on_side_4 flag booleano inserimento cerniera cilindrica su lato 4

elemtype_winklerShell elemento guscio di fondazione

gli altri campi sono:

elast_mod modulo di elasticità

poisson coefficiente di Poisson

thickness spessore

subgrade_mod coefficiente di sottofondo

elemtype_axial elemento assialsimmetrico

gli altri campi sono:

elast_mod modulo di elasticità

poisson coefficiente di Poisson

elemtype_planeStrain elemento a deformazione piana

gli altri campi sono:

elast_mod modulo di elasticità

poisson	coefficiente di Poisson
elemtype_brick	elemento solido brick

gli altri campi sono:

elast_mod	modulo di elasticità
poisson	coefficiente di Poisson

I dati dei vincoli interni di una trave, quando presenti, sono rappresentati dai seguenti campi:

release_i_tx, **release_i_ty**, **release_i_tz**	componenti di spostamento dei vincoli interni del primo nodo
release_i_rx, **release_i_ry**, **release_i_rz**	componenti di rotazione dei vincoli interni del primo nodo
release_j_tx, **release_j_ty**, **release_j_tz**	componenti di spostamento dei vincoli interni del secondo nodo
release_j_rx, **release_j_ry**,**release_j_rz**	componenti di rotazione dei vincoli interni del secondo nodo

I dati del profilo di una trave, quando assegnati, sono rappresentati dai seguenti campi:

profile_name	nome del profilo
profile_area	area del profilo
profile_jx, **profile_jy**, **profile_jz**	componenti dell'inerzia torsionale, Y e Z
profile_wey, **profile_wez**	moduli di resistenza elastici relativi a Y e Z
profile_wpy, **profile_wpz**	moduli di resistenza plastici relativi a Y e Z
profile_code	codice di forma del profilo
profile_height, **profile_width**	altezza e larghezza della sezione
profile_web_tk, **profile_flange_tk**	spessore dell'anima e delle ali
profile_radius	raggio di raccordo

profile_rot_cod1, **profile_rot_cod2** codici di rotazione per profili accoppiati

profile_dist_y, **profile_dist_z** distanza in Y e Z tra i profili nel caso di profili accoppiati

Si rimanda al manuale di Nòlian per una descrizione più dettagliata di tutti i valori qui brevemente descritti.

nolian.get_user_views_names( )

Restituisce una tabella contenente i nomi delle viste utente definite per Nòlian nel file corrente.

nolian.append_user_view(s)

Inserisce nel documento di uscita un'immagine generata da Nòlian per la vista utente specificata nel parametro *s*.

Restituisce il valore massimo calcolato nella rappresentazione (generalmente mostrato in calce all'immagine) o zero se per la rappresentazione scelta non è previsto alcun valore massimo.

nolian.add_param_view_layer(s, type)

Imposta la visibilità del layer *s* per le seguenti generazioni di viste parametriche da Nòlian con **nolian.append_param_view**.

Il parametro *type* indica il tipo di visibilità del layer specificato e può assumere uno tra i seguenti valori enumerati:

nolian.layer_visible il layer sarà visibile

nolian.layer_invisible il layer non sarà visibile

nolian.layer_grayed gli elementi del layer saranno mostrati in grigio

Le opzioni di visibilità dei layer possono essere resettate con la funzione **nolian.clear_param_view_layers**. Quando nessuna opzione è specificata, tutti i layer sono considerati visibili.

nolian.clear_param_view_layers( )

Azzera le specifiche dei layer precedentemente impostate con **nolian.add_param_view_layer** per la generazione programmatica delle viste in Nòlian.

nolian.append_param_view(w, h)

Inserisce nel documento di uscita un'immagine generata da Nòlian in base a parametri predefiniti. I parametri *w* e *h* sono rispettivamente la larghezza e l'altezza dell'immagine in centimetri.

Restituisce il valore massimo calcolato nella rappresentazione (generalmente mostrato in calce all'immagine) o zero se per la rappresentazione scelta non è previsto alcun valore massimo.

Un terzo parametro opzionale può essere una tabella contenente le impostazioni per la generazione dell'immagine. I campi che formano la tabella possono avere indice numerico o letterale indifferentemente per un più rapido accesso ai singoli parametri. I campi riconosciuti sono (tra parentesi l'indice da poter usare in alternativa al nome):

alpha (1) angolo di rotazione orizzontale della vista

beta (2) angolo di rotazione verticale della vista

section_view (3) se **true** viene impostata una vista in sezione

workplane (4) se **section_view** è **true**, specifica il tipo di piano su cui impostare la sezione, uno tra i valori enumerati:

- **nolian.plane_xy**
- **nolian.plane_yz**
- **nolian.plane_xz**
- **nolian.plane_generic**

workplane_elevation (5) se **section_view** è **true**, specifica l'elevazione del piano di sezione in base al tipo indicato da **workplane**;

drawing_mode (6) codice della rappresentazione voluta, uno tra i seguenti valori enumerati:

nolian.render_standard	rappresentazione standard in wireframe
nolian.render_solid_model	rappresentazione del modello solido
nolian.render_hidden_lines_removal	rappresentazione con rimozione linee nascoste
nolian.render_shrink	rappresentazione shrink
nolian.render_solid_shrink	rappresentazione shrink solida
nolian.render_loads	diagramma dei carichi
nolian.render_forces	diagramma delle forze

nolian.render_moments	diagramma dei momenti
nolian.render_all_loads	diagramma globale dei carichi
nolian.render_distr_masses	diagramma delle masse distribuite
nolian.render_local_system	diagramma dei riferimenti locali
nolian.render_free_edges	diagramma dei lati liberi
nolian.render_constraints	diagramma dei vincoli
nolian.render_deformed_shape	rappresentazione deformata
nolian.render_modal_shape	rappresentazione delle forme modali (utilizza il modo di vibrare corrente impostato con **sap.use_mode**)
nolian.render_isodisplacements	rappresentazione degli isospostamenti
nolian.render_flexural_stresses_diagram	diagramma dei momenti
nolian.render_shear_stresses_diagram	diagramma degli sforzi di taglio
nolian.render_axial_stresses_diagram	diagramma degli sforzi assiali
nolian.render_flexural_stress_directions	rappresentazione delle direzioni principali flessionali
nolian.render_membrane_stress_directions	rappresentazione delle direzioni principali membranali
nolian.render_equiv_flexural_isostresses	rappresentazione degli isosforzi equivalenti flessionali
nolian.render_equiv_membrane_isostresses	rappresentazione degli isosforzi equivalenti membranali
nolian.render_equiv_shell_isostresses	rappresentazione degli isosforzi equivalenti guscio
nolian.render_principal_traction_isostresses	rappresentazione degli isosforzi principali a trazione
nolian.render_principal_compression_isostresses	rappresentazione degli isosforzi principali a compressione
nolian.render_principal_shear_isostresses	rappresentazione degli isosforzi principali a taglio membrana

nolian.render_mesh_quality		diagramma di valutazione della mesh

Si rimanda al manuale di Nòlian per la descrizione delle rappresentazioni disponibili.

amplification	(7)	fattore di scala per la deformazione; se 0, la scala viene scelta in automatico;
add_deformed	(8)	se **true** in alcune rappresentazioni (isosforzi, isospostamenti...) viene aggiunto l'effetto degli spostamenti dei nodi;
show_iso_scale	(9)	specifica (**true** / **false**) se mostrare la legenda di riferimento nelle rappresentazioni a mappa di colori (isosforzi, isospostamenti...);
show_node_indices	(10)	indica (**true** / **false**) se mostrare gli indici dei nodi;
show_elem_indices	(11)	indica (**true** / **false**) se mostrare gli indici degli elementi;
show_load_in_color	(12)	indica se nella rappresentazione **nolian.render_loads** si devono mostrare i carichi con una scala di colore (**true**) oppure con delle frecce proporzionali all'entità del carico (**false**).

nolian.regularity_check()

Esegue un'analisi di regolarità con i criteri descritti nel manuale di Nòlian. Viene eseguita solo sui piani rigidi modellati con il metodo master-slave.

Restituisce una tabella che elenca i seguenti valori:

max_ecc	Massima eccentricità in rapporto al lato maggiore
max_omega	Massimo rapporto frequenze disaccoppiate translazionale/torsionale
max_stiff_gir_ratio	Massimo rapporto giratori di rigidezza
max_vert_ecc_var	Massima variazione verticale di eccentricità
max_vert_mass_var	Massima variazione verticale di massa
is_regular	Ipotesi di regolarità (true = regolare) secondo il criterio euristico espresso nel manuale di Nòlian. I valori restituiti possono essere impiegati per formulare un diverso criterio.

nolian.open_inmod_doc(path)

Apre e rende corrente un documento inMod corrispondente alla posizione specificata con *path*.

nolian.close_inmod_doc( )

Chiude il documento inMod corrente.

nolian.get_inmod_load_packets_names(filter)

Restituisce una tabella che elenca i nomi dei pacchetti di carico utilizzati, per le tipologie di elementi piani specificate con *filter*, nel file inMod corrente.

Il parametro *filter* può essere costituito da una delle seguenti costanti oppure dalla somma di due o più di esse, purché non ripetute.

nolian.inmod_pe_any	prende in considerazione tutte le tipologie di elementi piani
nolian.inmod_pe_plate	prende in considerazione la tipologia "guscio"
nolian.inmod_pe_floor	prende in considerazione la tipologia "solaio"
nolian.inmod_pe_winkler	prende in considerazione la tipologia "piastra Winkler"
nolian.inmod_pe_wall	prende in considerazione la tipologia "tamponatura"

nolian.get_inmod_load_packet_applied_types(filter, packet)

Restituisce una tabella che elenca i nomi dei tipi di elementi piani cui è applicato il pacchetto di carico identificato dal nome *packet*, per le tipologie di elementi piani specificate con *filter*, nel file inMod corrente.

Le tipologie di elementi piani vengono indicate solo se, soggette allo stesso pacchetto di carico, abbiano però valori different di peso proprio (derivati da un diverso spessore e materiale).

La tabella sarà invece vuota se il peso proprio è lo stesso per tutti gli elementi coinvolti, o nel caso in cui il peso proprio non compaia nel pacchetto di carico.

Il parametro *filter* può essere costituito da una delle seguenti costanti oppure dalla somma di due o più di esse, purché non ripetute.

nolian.inmod_pe_any	prende in considerazione tutte le tipologie di elementi piani
nolian.inmod_pe_plate	prende in considerazione la tipologia "guscio"

nolian.inmod_pe_floor prende in considerazione la tipologia "solaio"

nolian.inmod_pe_winkler prende in considerazione la tipologia "piastra Winkler"

nolian.inmod_pe_wall prende in considerazione la tipologia "tamponatura"

I nomi dei pacchetti di carico utilizzati sono reperibili tramite la funzione **nolian.get_inmod_load_packets_names()**.

nolian.get_inmod_load_packet_records_count(packet)

Restituisce il numero dei records definiti nel pacchetto di carico identificato dal nome *packet*, definito nel file inMod corrente.

I nomi dei pacchetti di carico utilizzati sono reperibili tramite la funzione **nolian.get_inmod_load_packets_names()**.

nolian.get_inmod_load_packet_record(filter, packet, type, index)

Restituisce una tabella con le proprietà del record di indice *index* appartenente al pacchetto di carico identificato dal nome *packet*, applicato ad elementi piani di tipo *type* appartenenti alle tipologie di elementi piani specificate con *filter*, definito ne file inMod corrente.

Il parametro *filter* può essere costituito da una delle seguenti costanti oppure dalla somma di due o più di esse, purché non ripetute. Esso viene utilizzato solo nel caso in cui *type* sia uguale a "", ciò che accade quando **nolian.get_inmod_load_packet_applied_types()** abbia restituito una tabella vuota.

nolian.inmod_pe_any prende in considerazione tutte le tipologie di elementi piani

nolian.inmod_pe_plate prende in considerazione la tipologia "guscio"

nolian.inmod_pe_floor prende in considerazione la tipologia "solaio"

nolian.inmod_pe_winkler prende in considerazione la tipologia "piastra Winkler"

nolian.inmod_pe_wall prende in considerazione la tipologia "tamponatura"

I nomi dei pacchetti di carico utilizzati sono reperibili tramite la funzione **nolian.get_inmod_load_packets_names()**.

I nomi dei tipi di elementi piani definiti sono reperibili tramite la funzione **nolian.get_inmod_load_packet_applied_types()**.

Il numero di records definiti in un pacchetto di carico è reperibile tramite la funzione **nolian.get_inmod_load_packet_records_count()**.

Le proprietà di un record sono le seguenti:

name nome del record

lc_name nome della combinazione di carico di riferimento

dir_x componente X della direzione di applicazione

dir_y componente Y della direzione di applicazione

dir_z componente Z della direzione di applicazione

ref_type tipo di riferimento (0 = locale, 1 = globale, 2 = globale proiettato)

mod_linear modulo per l'applicazione come carico lineare

mod_surface modulo per l'applicazione come carico di superficie

Nell'esempio seguente viene mostrato come stampare l'elenco integrale dei carichi definiti in un file inMod.

```
-- Accesso all'analisi dei carichi in InMod

path = dlg.ask_file("", {dlg.filter_mod})

if(path and nolian.open_inmod_doc(path)) then

 out.set_font("AvantGarde", 10)
 out.set_font_style(out.style_bold)
 out.append("\n\n-- Accesso all'analisi dei carichi in InMod --\n\n")
 out.set_font_style()
 out.set_font("Courier New", 9)

 type_filter  = nolian.inmod_pe_any;
 packets_names = nolian.get_inmod_load_packets_names(type_filter)

 if packets_names then
  for p, packet_name in pairs(packets_names) do
   out.append("Pacchetto \"" .. tostring(packet_name) .. "\":\n")

   records_count = nolian.get_inmod_load_packet_records_count(packet_name)
   applied_types = nolian.get_inmod_load_packet_applied_types(type_filter, pa

   if applied_types then

    -- se l'elenco dei tipi è vuoto
    if # applied_types == 0 then
     applied_types[0] = ""
    end

    for t, type_name in pairs(applied_types) do
     out.append("Tipo \"" .. tostring(type_name) .. "\":\n")

     for r = 0, records_count-1 do
      out.append("\tRecord " .. r .. ":\n")

      record_data = nolian.get_inmod_load_packet_record(type_filter, packet_n

      if record_data then
```

```
        for field, field_value in pairs(record_data) do
         out.append("\t\t" .. field .. ": " .. tostring(field_value) .. ";\n")
        end
       end
      end
     end
    end
   end
  end

 nolian.close_inmod_doc()
end
```

Interazione con EasyBeam

Il motore di scripting mette a disposizione alcune funzioni di interrogazione diretta ad EasyBeam per dati e risultati che non possono essere letti direttamente da file attraverso le funzioni della libreria **sap** descritta nel capitolo precedente.

Le funzioni della libreria **easybeam** qui descritta si basano su uno stato di computazione modificato attraverso le funzioni di tipo **use_*** ed interrogato attraverso le funzioni **get_***. Ved. il capitolo precedente "Funzioni di accesso ai file SAP" per maggiori informazioni.

easybeam.get_app_info( )

Restituisce una tabella con informazioni sintetiche sull'applicazione. La tabella ha i seguenti campi:

manufacturer il nome del produttore, Softing s.r.l.

name il nome dell'applicazione

version la versione dell'applicazione

license il numero di licenza dell'utente

easybeam.get_combinations(t)

Restituisce una tabella che descrive le combinazioni di carico impostate in EasyBeam: ogni elemento della tabella è una espressione letterale della combinazione, ad es. "*1.00 * permanente + 0.30 * accidentale*".

Il parametro *t* può assumere uno tra i seguenti valori enumerati:

easybeam.combinations_design per ottenere le combinazioni di progetto

easybeam.combinations_service per ottenere le combinazioni di esercizio

easybeam.combinations_damage per ottenere le combinazioni di danno

easybeam.combinations_operativity per ottenere le combinazioni di operatività

easybeam.combinations_geotechnical per ottenere le combinazioni geotecniche

easybeam.combinations_exceptional per ottenere le combinazioni eccezionali

easybeam.get_load_types_names( )

Restituisce una tabella che elenca i tipi di carico definiti in EasyBeam.

easybeam.get_load_type_data(n)

Restituisce le impostazioni relative al tipo di carico con il nome richiesto.

Viene restituita una tabella avente i seguenti campi:

type tipo assegnato dall'utente; uno tra i seguenti valori enumerati:

- **easybeam.loadtype_permanent** permanente
- **easybeam.loadtype_variable** variabile
- **easybeam.loadtype_not_concurrent_variable** variabile non contemporaneo
- **easybeam.loadtype_seismic** sismico
- **easybeam.loadtype_seismic_related** sismico correlato

gravitational **true** se deriva da peso proprio

gamma_unf valore del coefficiente gamma

gamma_seismic_unf valore del coefficiente gamma sismico

psi_0 valore del coefficiente psi zero

psi_1 valore del coefficiente psi uno

psi_2 valore del coefficiente psi due

psi_2_seismic valore del coefficiente psi due sismico

phi valore del coefficiente di correlazione

easybeam.get_user_views_names( )

Restituisce una tabella contenente i nomi delle viste utente definite per EasyBeam nel file corrente.

easybeam.append_user_view(s)

Inserisce nel documento di uscita un'immagine generata da EasyBeam per la vista utente specificata nel parametro *s*.

Restituisce il valore massimo calcolato nella rappresentazione (generalmente mostrato in calce all'immagine) o zero se per la rappresentazione scelta non è previsto alcun valore massimo.

easybeam.add_param_view_layer(s, type)

Imposta la visibilità del layer s per le seguenti generazioni di viste parametriche da EasyBeam con **easybeam.append_param_view**.

Il parametro *type* indica il tipo di visibilità del layer specificato e può assumere uno tra i seguenti valori enumerati:

easybeam.layer_visible il layer sarà visibile

easybeam.layer_invisible il layer non sarà visibile

easybeam.layer_grayed gli elementi del layer saranno mostrati in grigio

Le opzioni di visibilità dei layer possono essere resettate con la funzione **easybeam.clear_param_view_layers**. Quando nessuna opzione è specificata, tutti i layer sono considerati visibili.

easybeam.clear_param_view_layers( )

Azzera le specifiche dei layer precedentemente impostate con **easybeam.add_param_view_layer** per la generazione programmatica delle viste in EasyBeam.

easybeam.add_param_view_elements(idx1, idx2)

Aggiunge gli elementi con indice compreso nell'intervallo tra *idx1* ed *idx2* alla selezione corrente di EasyBeam per la generazione programmatica della vista attraverso la funzione **easybeam.append_param_view.**

La selezione degli elementi può essere resettata con la funzione **easybeam.clear_param_view_elements**. Quando nessuna selezione è specificata, tutti gli elementi sono considerati selezionati.

easybeam.clear_param_view_elements( )

Azzera la selezione degli elementi precedentemente impostata con **easybeam.add_param_view_elements** per la generazione programmatica delle viste in EasyBeam.

easybeam.append_param_view(w, h)

Inserisce nel documento di uscita un'immagine generata da EasyBeam in base a parametri predefiniti. I parametri *w* e *h* sonc rispettivamente la larghezza e l'altezza dell'immagine in centimetri.

Restituisce il valore massimo calcolato nella rappresentazione (generalmente mostrato in calce all'immagine) o zero se per la rappresentazione scelta non è previsto alcun valore massimo.

Un terzo parametro opzionale può essere una tabella contenente le impostazioni per la generazione dell'immagine. I campi che formano la tabella possono avere indice numerico o letterale indifferentemente per un più rapido accesso ai singoli parametri. I campi riconosciuti sono (tra parentesi l'indice da poter usare in alternativa al nome):

alpha	(1)	angolo di rotazione orizzontale della vista
beta	(2)	angolo di rotazione verticale della vista
section_view	(3)	se **true** viene impostata una vista in sezione
workplane	(4)	se section_view è true, specifica il tipo di piano su cui impostare la sezione, uno tra i valori enumerati:
easybeam.plane_xy		
easybeam.plane_yz		
easybeam.plane_xz		
easybeam.plane_generic		
workplane_elevation	(5)	se **section_view** è **true**, specifica l'elevazione del piano di sezione in base al tipo indicato da **workplane**;
show_code	(6)	codice della rappresentazione voluta, uno tra i seguenti valori enumerati:
easybeam.render_none		rappresentazione standard in wireframe
easybeam.render_reinforcement		rappresentazione delle armature

easybeam.render_concrete_envelope	rappresentazione dell'inviluppo delle sollecitazioni nel calcestruzzo
easybeam.render_steel_envelope	rappresentazione dell'inviluppo delle sollecitazioni nell'acciaio
easybeam.render_sls_cracking	rappresentazione delle verifiche allo stato limite di servizio (fessurazione)
easybeam.render_sls_stress	rappresentazione delle verifiche allo stato limite di servizio (tensione)
easybeam.render_shear_global	rappresentazione della verifica a taglio globale, progetto e sismico
easybeam.render_shear_design	rappresentazione della verifica a taglio, solo progetto
easybeam.render_shear_seismic	rappresentazione della verifica a taglio, solo sismico
easybeam.render_safety_factor	rappresentazione del fattore di sicurezza
easybeam.render_stress_diagram_moment	rappresentazione dei diagrammi di sollecitazione del momento flettente
easybeam.render_stress_diagram_shear	rappresentazione dei diagrammi di sollecitazione dello sforzo di taglio
easybeam.render_stress_diagram_axial	rappresentazione dei diagrammi di sollecitazione dello sforzo assiale
easybeam.render_ductility	rappresentazione della verifica a duttilità
easybeam.render_print	rappresentazione delle sezioni di riferimento (impostate con **easybeam.add_param_view_elements**)
easybeam.render_joints_hierarchy	rappresentazione della verifica dei giunti - gerarchia delle resistenze
easybeam.render_joints_shear	rappresentazione della verifica dei giunti - resistenza a taglio
easybeam.render_sld	rappresentazione delle verifiche allo stato limite di danno
easybeam.render_slv	rappresentazione delle verifiche allo stato limite di salvaguardia della vita
easybeam.render_slo	rappresentazione delle verifiche allo stato limite di operatività
easybeam.render_global_check	rappresentazione della verifica generale

easybeam.render_geo_deflection	rappresentazione delle verifiche geotecniche - mappa della deflessione
easybeam.render_geo_pressure	rappresentazione delle verifiche geotecniche - mappa della pressione
easybeam.render_geo_lift_exploitation	rappresentazione delle verifiche geotecniche - mappa dello sfruttamento della portanza
easybeam.render_quarmon_flex_check	(Quarmon) rappresentazione del fattore di sicurezza flessionale degli elementi sottoposti a incendio
easybeam.render_quarmon_shear_check	(Quarmon) rappresentazione del fattore di sicurezza a taglio degli elementi sottoposti a incendio
easybeam.render_exsys_classification	(ExSys) classificazione duttile/fragile
easybeam.render_exsys_joint_check	(ExSys) verifica nodi
easybeam.render_exsys_element_check	(ExSys) verifica elementi
easybeam.render_exsys_rot_capacity_slv	(ExSys) verifica della capacità rotazionale - stato limite di salvaguardia della vita
easybeam.render_exsys_rot_capacity_slc	(ExSys) verifica della capacità rotazionale - stato limite di collasso
easybeam.render_exsys_rot_capacity_sld	(ExSys) verifica della capacità rotazionale - stato limite di danno
easybeam.render_exsys_pga_multiplier	(ExSys) moltiplicatore PGA
easybeam.render_frp_classification	(FibRePower) classificazione duttile/fragile
easybeam.render_frp_joint_check	(FibRePower) verifica nodi
easybeam.render_frp_element_check	(FibRePower) verifica elementi
easybeam.render_frp_rot_capacity_slv	(FibRePower) verifica della capacità rotazionale - stato limite di salvaguardia della vita
easybeam.render_frp_rot_capacity_slc	(FibRePower) verifica della capacità rotazionale - stato limite di collasso
easybeam.render_frp_rot_capacity_sld	(FibRePower) verifica della capacità rotazionale - stato limite di danno

easybeam.render_frp_pga_multiplier	(FibRePower) moltiplicatore PGA
easybeam.render_frp_joint_hierarchy	(FibRePower) verifica nodi - gerarchia delle resistenze
easybeam.render_frp_coverings	(FibRePower) rappresentazione solida rinforzi in FRP
easybeam.render_wallver_numbering	(WallVerine) numerazione
easybeam.render_wallver_frp_in_plane	(WallVerine) disposizione rinforzi FRP in piano
easybeam.render_wallver_frp_out_of_plane	(WallVerine) disposizione rinforzi FRP fuori piano
easybeam.render_wallver_chk_without_frp	(WallVerine) verifica resistenza muratura senza rinforzo FRP
easybeam.render_wallver_pga_without_frp	(WallVerine) moltiplicatore PGA senza rinforzo FRP
easybeam.render_wallver_chk_with_frp	(WallVerine) verifica resistenza muratura con rinforzo FRP
easybeam.render_wallver_pga_with_frp	(WallVerine) moltiplicatore PGA con rinforzo FRP
easybeam.render_wallver_oop_bending	(WallVerine) meccanismo flessione fuori piano
easybeam.render_enjoist_check	(EnJoist) verifica

È inteso che le rappresentazioni relative a Quarmon, ExSys, WallVerine e FibRePower sono utilizzabili esclusivamente dall'interno dell'applicazione All In One e solo nel caso in cui lo specifico ambiente sia effettivamente disponibile. Si rimanda ai relativi manuali per la descrizione delle rappresentazioni disponibili.

diagrams_scale	(7)	fattore di scala per i diagrammi
show_unselected_axis	(8)	se **true** in alcune rappresentazioni viene mostrato l'asse degli elementi non selezionati
show_selected_axis	(9)	se **true** in alcune rappresentazioni viene mostrato l'asse degli elementi selezionati
show_solid	(10)	se **true** in alcune rappresentazioni viene mostrato il solido degli elementi selezionati
show_bars	(11)	se **true** in alcune rappresentazioni viene mostrata l'armatura degli elementi selezionati

show_bars_exploded	(12)	se **true** mostra le armature esplose
show_names	(13)	indica la modalità di denominazione degli elementi (0: nascosta; 1: singoli elementi; 2: travate)
show_bar_edit_mode	(14)	se **true** attiva la rappresentazione dell'editing piano delle barre
show_max_shear	(15)	se **true** mostra il valore massimo nella verifica a taglio
show_max_ductility	(16)	se **true** mostra il valore massimo nella verifica a duttilità
show_max_interaction	(17)	se **true** mostra il valore massimo nella verifica di interazione
shown_index	(18)	indica di quale elemento mostrare l'indice (0: tutti; -1: nessuno)
sls_max_cracks_width_rare	(19)	massima ampiezza delle fessure per la verifica a fessurazione in combinazione rara
sls_max_cracks_width_frequent	(20)	massima ampiezza delle fessure per la verifica a fessurazione in combinazione frequente
sls_max_cracks_width_almost_permanent	(21)	massima ampiezza delle fessure per la verifica a fessurazione in combinazione quasi permanente
sls_max_tension_rare_concrete	(22)	coefficiente per il calcolo della massima tensione nella verifica allo stato limite di servizio per la combinazione rara nel calcestruzzo
sls_max_tension_rare_steel	(23)	coefficiente per il calcolo della massima tensione nella verifica allo stato limite di servizio per la combinazione rara nell'acciaio
sls_max_tension_almost_permanent_concrete	(24)	coefficiente per il calcolo della massima tensione nella verifica allo stato limite di servizio per la combinazione quasi permanente nel calcestruzzo
sls_max_tension_almost_permanent_steel	(25)	coefficiente per il calcolo della massima tensione nella verifica allo stato limite di servizio per la combinazione quasi permanente nell'acciaio
sls_comb_rare	(26)	se **true** attiva la combinazione rara nella verifica allo stato limite di servizio
sls_comb_frequent	(27)	se **true** attiva la combinazione frequente nella verifica allo stato limite di servizio
sls_comb_almost_permanent	(28)	se **true** attiva la combinazione quasi permanente nella verifica allo stato limite di servizio

geo_drained	(29)	se **true** attiva la rappresentazione delle verifiche geotecniche in condizione drenata, altrimenti non drenata.

easybeam.get_max_rel_displacement(code, amplif)

Restituisce il massimo spostamento relativo (spostamento / lunghezza) tra nodi collegati da pilastri nella struttura correntemente aperta in EasyBeam (ved. **sap.use_file**).

Il parametro *code* può essere:

easybeam.render_sld	stato limite di danno
easybeam.render_slv	stato limite di salvaguardia della vita
easybeam.render_slo	stato limite di operatività

Il parametro *amplif* è il moltiplicatore applicato agli spostamenti.

Il calcolo viene effettuato in base alle impostazioni presenti sul file per le combinazioni relative. Se le impostazioni non sono presenti nel file, verrà richiesto di specificarle.

La funzione restituisce tre valori, rispettivamente:

- il valore del massimo spostamento relativo
- la quota massima del pilastro
- la lunghezza del pilastro

Il calcolo è quello svolto dalla verifica allo stato limite di danno in EasyBeam. Si rimanda quindi alla sua documentazione per maggiori dettagli.

easybeam.get_stress_strain_params( )

Restituisce una tabella che elenca i seguenti valori caratteristici dei legami costitutivi dei materiali:

eps_s	Deformazione massima dell'acciaio
eps_c	Deformazione massima del calcestruzzo
eps_0	Ascissa in deformazione dell'asse della parabola del calcestruzzo
gamma_s	Fattore di sicurezza parziale per l'acciaio
gamma_c	Fattore di sicurezza parziale per il calcestruzzo

easybeam.get_exsys_summary( )

Restituisce una tabella con una sintesi dei risultati delle verifiche condotte nell'ambiente ExSys:

analysis_type	tipo di analisi impiegato, uno tra i seguenti valori enumerati:
sap.ex_design_spectrum	analisi con spettro di progetto
sap.ex_elastic_spectrum	analisi con spettro elastico
sap.ex_static_nl_analysis	analisi statica non lineare
num_of_elements	numero totale di elementi
num_of_non_struct_elements	numero di elementi non strutturali
num_of_ductile_elements	numero di elementi duttili
num_of_fragile_elements	numero di elementi fragili
num_of_verified_elements	numero di elementi verificati
acc_collapse_mult_design	moltiplicatore di collasso dell'accelerazione di progetto

Se **analysis_type** corrisponde a **sap.ex_design_spectrum**, la tabella conterrà anche i seguenti valori:

min_safety_factor_fragile	minimo fattore di sicurezza elementi fragili
acc_collapse_mult_shear	moltiplicatore di collasso dell'accelerazione - taglio (valido solo quando **analysis_type** corrisponde a **sap.ex_elastic_spectrum**)
acc_collapse_mult_flexure	moltiplicatore di collasso dell'accelerazione - flessione (valido solo quando **analysis_type** corrisponde a **sap.ex_elastic_spectrum**)
return_period_shear	periodo di ritorno - taglio (valido solo quando **analysis_type** corrisponde a **sap.ex_elastic_spectrum**)
return_period_flexure	periodo di ritorno - flessione (valido solo quando **analysis_type** corrisponde a **sap.ex_elastic_spectrum**)

num_of_nodes	numero totale di nodi
num_of_non_confined_nodes	numero di nodi non confinati
min_safety_factor_nodes	minimo fattore di sicurezza nodi
acc_collapse_mult_nodes	PGA o moltiplicatore di collasso dell'accelerazione nodi, se accelerazione di progetto non è reperibile
return_period_nodes	periodo di ritorno - nodi

Se **analysis_type** corrisponde a **sap.ex_elastic_spectrum**, la tabella conterrà anche i seguenti valori:

method_is_admitted	*true* se il metodo è ammesso (valido solo quando **analysis_type** corrisponde a **sap.ex_elastic_spectrum**)
num_of_admitted_elements	numero di elementi ammessi (valido solo quando **analysis_type** corrisponde a **sap.ex_elastic_spectrum**)
stiffness_distr_ratio	rapporto distribuzione rigidezze (valido solo quando **analysis_type** corrisponde a **sap.ex_elastic_spectrum**)
min_safety_factor_ductile	minimo fattore di sicurezza elementi duttili (valido solo quando **analysis_type** corrisponde a **sap.ex_elastic_spectrum**)
min_safety_factor_fragile	minimo fattore di sicurezza elementi fragili

easybeam.get_enjoist_materials( )

Restituisce una tabella con i parametri dei materiali usati da EnJoist:

concrete_design_strength	resistenza caratteristica del calcestruzzo
concrete_partial_safety_factor	coefficiente di sicurezza parziale del calcestruzzo
steel_design_strength	resistenza caratteristica dell'acciaio
steel_partial_safety_factor	coefficiente di sicurezza parziale dell'acciaio

easybeam.get_enjoist_summary( )

Restituisce una tabella con i minimi fattori di sicurezza usati da EnJoist:

min_safety_factor_flexural	coefficiente di sicurezza a flessione
min_safety_factor_shear	coefficiente di sicurezza a taglio
min_safety_factor_cracking	coefficiente di sicurezza a fessurazione
min_safety_factor_deflection	coefficiente di sicurezza di deformabilità

Interazione con EasySteel

Il motore di scripting mette a disposizione alcune funzioni di interrogazione diretta ad EasySteel per dati e risultati che non possono essere letti direttamente da file attraverso le funzioni della libreria **sap** descritta nel capitolo precedente.

Le funzioni della libreria **easysteel** qui descritta si basano su uno stato di computazione modificato attraverso le funzioni di tipo **use_*** ed interrogato attraverso le funzioni **get_***. Ved. il capitolo precedente "Funzioni di accesso ai file SAP" per maggiori informazioni.

easysteel.get_app_info( )

Restituisce una tabella con informazioni sintetiche sull'applicazione. La tabella ha i seguenti campi:

manufacturer	il nome del produttore, Softing s.r.l.
name	il nome dell'applicazione
version	la versione dell'applicazione
license	il numero di licenza dell'utente

easysteel.get_combinations(t)

Restituisce una tabella che descrive le combinazioni di carico impostate in EasySteel: ogni elemento della tabella è una espressione letterale della combinazione, ad es. "*1.00 * permanente + 0.30 * accidentale*".

Il parametro *t* può assumere uno tra i seguenti valori enumerati:

easysteel.combinations_design	per ottenere le combinazioni di progetto

easysteel.combinations_damage per ottenere le combinazioni di danno

easysteel.combinations_operativity per ottenere le combinazioni di operatività

easysteel.get_load_types_names( )

Restituisce una tabella che elenca i tipi di carico definiti in EasySteel.

easysteel.get_load_type_data(n)

Restituisce le impostazioni relative al tipo di carico con il nome richiesto.

Viene restituita una tabella avente i seguenti campi:

type tipo assegnato dall'utente; uno tra i seguenti valori enumerati:

- **easysteel.loadtype_permanent** permanente
- **easysteel.loadtype_variable** variabile
- **easysteel.loadtype_not_concurrent_variable** variabile non contemporaneo
- **easysteel.loadtype_seismic** sismico
- **easysteel.loadtype_seismic_related** simico correlato

gravitational **true** se deriva da peso proprio

gamma_unf valore del coefficiente gamma

gamma_seismic_unf valore del coefficiente gamma sismico

psi_0 valore del coefficiente psi zero

psi_1 valore del coefficiente psi uno

psi_2 valore del coefficiente psi due

psi_2_seismic valore del coefficiente psi due sismico

phi valore del coefficiente di correlazione

easysteel.get_user_views_names( )

Restituisce una tabella contenente i nomi delle viste utente definite per EasySteel nel file corrente.

easysteel.append_user_view(s)

Inserisce nel documento di uscita un'immagine generata da EasySteel per la vista utente specificata nel parametro *s*.

easysteel.add_param_view_layer(s, type)

Imposta la visibilità del layer s per le seguenti generazioni di viste parametriche da EasySteel con **easysteel.append_param_view**.

Il parametro *type* indica il tipo di visibilità del layer specificato e può assumere uno tra i seguenti valori enumerati:

easysteel.layer_visible il layer sarà visibile

easysteel.layer_invisible il layer non sarà visibile

easysteel.layer_grayed gli elementi del layer saranno mostrati in grigio

Le opzioni di visibilità dei layer possono essere resettate con la funzione **easysteel.clear_param_view_layers**. Quando nessuna opzione è specificata, tutti i layer sono considerati visibili.

easysteel.clear_param_view_layers( )

Azzera le specifiche dei layer precedentemente impostate con **easysteel.add_param_view_layer** per la generazione programmatica delle viste in EasySteel.

easysteel.add_param_view_elements(idx1, idx2)

Aggiunge gli elementi con indice compreso nell'intervallo tra *idx1* ed *idx2* alla selezione corrente di EasySteel per la generazione programmatica della vista attraverso la funzione **easysteel.append_param_view.**

La selezione degli elementi può essere resettata con la funzione **easysteel.clear_param_view_elements**. Quando nessuna selezione è specificata, tutti gli elementi sono considerati selezionati.

easysteel.clear_param_view_elements( )

Azzera la selezione degli elementi precedentemente impostata con **easysteel.add_param_view_elements** per la generazion programmatica delle viste in EasySteel.

easysteel.append_param_view(w, h)

Inserisce nel documento di uscita un'immagine generata da EasySteel in base a parametri predefiniti. I parametri *w* e *h sono rispettivamente la larghezza e l'altezza dell'immagine in centimetri.*

Un terzo parametro opzionale può essere una tabella contenente le impostazioni per la generazione dell'immagine. I campi che formano la tabella possono avere indice numerico o letterale indifferentemente per un più rapido accesso ai singoli parametri. I campi riconosciuti sono (tra parentesi l'indice da poter usare in alternativa al nome):

alpha (1) angolo di rotazione orizzontale della vista

beta (2) angolo di rotazione verticale della vista

section_view (3) se **true** viene impostata una vista in sezione

workplane (4) se section_view è true, specifica il tipo di piano su cui impostare la sezione, uno tra i valori enumerati:

- **easysteel.plane_xy**
- **easysteel.plane_yz**
- **easysteel.plane_xz**
- **easysteel.plane_generic**

workplane_elevation (5) se **section_view** è **true**, specifica l'elevazione del piano di sezione in base al tipo indicato da **workplane**

show_mode (6) codice della rappresentazione voluta, uno tra i seguenti valori enumerati:

- **easysteel.render_none** rappresentazione standard in wireframe
- **easysteel.render_element_check** rappresentazione della verifica degli elementi per la flessione
- **easysteel.render_element_shea_check** rappresentazione della verifica degli elementi per il taglio
- **easysteel.render_joints_check** rappresentazione della verifica dei giunti
- **easysteel.render_stress_diagram** rappresentazione dei diagrammi di sollecitazione

easysteel.render_solid rappresentazione con rendering solido

Si rimanda al manuale di EasySteel per la descrizione delle rappresentazioni disponibili.

check_flags (7) flags per la modalità combinata di rappresentazione delle verifiche, uno tra i seguenti valori enumerati:

easysteel.check_none nessuno

easysteel.check_shear_y_strength resistenza a taglio Y

easysteel.check_shear_z_strength resistenza a taglio Z

easysteel.check_shear_instability instabilità a taglio

easysteel.check_axial_flexural_instability instabilità presso-flessionale

easysteel.check_flexural_torsional_instability instabilità flesso-torsionale

easysteel.check_axial_flexural_strength resistenza assiale-flessionale

easysteel.check_axial_strength resistenza assiale

easysteel.check_seismic_shear taglio sismico

easysteel.check_all tutti

diagrams_scale (8) fattore di scala per i diagrammi

diagrams_mode (9) tipo di diagramma da mostrare con **easysteel.render_stress_diagram**, uno a scelta tra i valori enumerati:

easysteel.diagram_moment (momento flettente)

easysteel.diagram_shear (sforzo di taglio)

easysteel.diagram_axial (sforzo assiale)

show_node_names (10) se **true** mostra la denominazione dei nodi

show_elem_names	(11)	se **true** mostra la denominazione degli elementi
show_prof_names	(12)	se **true** mostra la denominazione dei profili
show_fillets	(13)	se **true** nel rendering solido mostra il raccordo circolare dei profili
show_joints	(14)	se **true** nel rendering solido mostra i giunti
show_shrinking	(15)	se **true** nel rendering solido mostra la calettatura
show_bolts	(16)	se **true** nel rendering solido mostra i bulloni
show_weldings	(17)	se **true** nel rendering solido mostra le saldature
show_hooks	(18)	se **true** nel rendering solido mostra i tirafondi

easysteel.get_max_rel_displacement(code, amplif)

Restituisce il massimo spostamento relativo (spostamento / lunghezza) tra nodi collegati da pilastri nella struttura correntemente aperta in EasySteel (ved. **sap.use_file**).

Il parametro *code* può essere:

easysteel.reldisp_sld	stato limite di danno
easysteel.reldisp_slv	stato limite di salvaguardia della vita
easysteel.reldisp_slo	stato limite di operatività

Il parametro *amplif* è il moltiplicatore applicato agli spostamenti.

Il calcolo viene effettuato in base alle impostazioni presenti sul file per le combinazioni relative. Se le impostazioni non sono presenti nel file, verrà richiesto di specificarle.

La funzione restituisce tre valori, rispettivamente:

- il valore del massimo spostamento relativo
- la quota massima del pilastro
- la lunghezza del pilastro

Il calcolo è quello svolto dalla verifica allo stato limite di danno in EasySteel. Si rimanda quindi alla sua documentazione per maggiori dettagli.

Interazione con EasyWall

Il motore di scripting mette a disposizione alcune funzioni di interrogazione diretta ad EasyWall per dati e risultati che non possono essere letti direttamente da file attraverso le funzioni della libreria **sap** descritta nel capitolo precedente.

Le funzioni della libreria **easywall** qui descritta si basano su uno stato di computazione modificato attraverso le funzioni di tipo **use_*** ed interrogato attraverso le funzioni **get_***. Ved. il capitolo precedente "Funzioni di accesso ai file SAP" per maggiori informazioni.

easywall.get_app_info( )

Restituisce una tabella con informazioni sintetiche sull'applicazione. La tabella ha i seguenti campi:

manufacturer	il nome del produttore, Softing s.r.l.
name	il nome dell'applicazione
version	la versione dell'applicazione
license	il numero di licenza dell'utente

easywall.get_combinations(t)

Restituisce una tabella che descrive le combinazioni di carico impostate in EasyWall: ogni elemento della tabella è una espressione letterale della combinazione, ad es. "*1.00 * permanente + 0.30 * accidentale*".

Il parametro *t* può assumere uno tra i seguenti valori enumerati:

easywall.combinations_design	per ottenere le combinazioni di progetto
easywall.combinations_service	per ottenere le combinazioni di esercizio
easywall.combinations_geotechnical	per ottenere le combinazioni geotecniche

easywall.get_load_types_names( )

Restituisce una tabella che elenca i tipi di carico definiti in EasyWall.

easywall.get_load_type_data(n)

Restituisce le impostazioni relative al tipo di carico con il nome richiesto.

Viene restituita una tabella avente i seguenti campi:

type	tipo assegnato dall'utente; uno tra i seguenti valori enumerati:
easywall.loadtype_permanent	permanente
easywall.loadtype_variable	variabile
easywall.loadtype_not_concurrent_variable	variabile non contemporaneo
easywall.loadtype_seismic	sismico
easywall.loadtype_seismic_related	simico correlato
gravitational	**true** se deriva da peso proprio
gamma_unf	valore del coefficiente gamma
gamma_seismic_unf	valore del coefficiente gamma sismico
psi_0	valore del coefficiente psi zero
psi_1	valore del coefficiente psi uno
psi_2	valore del coefficiente psi due
psi_2_seismic	valore del coefficiente psi due sismico
phi	valore del coefficiente di correlazione

easywall.get_user_views_names( )

Restituisce una tabella contenente i nomi delle viste utente definite per EasyWall nel file corrente.

easywall.append_user_view(s)

Inserisce nel documento di uscita un'immagine generata da EasyWall per la vista utente specificata nel parametro *s*.

easywall.add_param_view_layer(s, type)

Imposta la visibilità del layer s per le seguenti generazioni di viste parametriche da EasyWall con **easywall.append_param_view**.

Il parametro *type* indica il tipo di visibilità del layer specificato e può assumere uno tra i seguenti valori enumerati:

easywall.layer_visible	il layer sarà visibile
easywall.layer_invisible	il layer non sarà visibile
easywall.layer_grayed	gli elementi del layer saranno mostrati in grigio

Le opzioni di visibilità dei layer possono essere resettate con la funzione **easywall.clear_param_view_layers**. Quando nessuna opzione è specificata, tutti i layer sono considerati visibili.

easywall.clear_param_view_layers( )

Azzera le specifiche dei layer precedentemente impostate con **easywall.add_param_view_layer** per la generazione programmatica delle viste in EasyWall.

easywall.add_param_view_elements(idx1, idx2)

Aggiunge gli elementi con indice compreso nell'intervallo tra *idx1* ed *idx2* alla selezione corrente di EasyWall per la generazione programmatica della vista attraverso la funzione **easywall.append_param_view.**

La selezione degli elementi può essere resettata con la funzione **easywall.clear_param_view_elements**. Quando nessuna selezione è specificata, tutti gli elementi sono considerati selezionati.

easywall.clear_param_view_elements( )

Azzera la selezione degli elementi precedentemente impostata con **easywall.add_param_view_elements** per la generazione programmatica delle viste in EasyWall.

easywall.append_param_view(w, h)

Inserisce nel documento di uscita un'immagine generata da EasyWall in base a parametri predefiniti. I parametri *w* e *h* sono rispettivamente la larghezza e l'altezza dell'immagine in centimetri.

Un terzo parametro opzionale può essere una tabella contenente le impostazioni per la generazione dell'immagine. I campi che formano la tabella possono avere indice numerico o letterale indifferentemente per un più rapido accesso ai singoli parametri. I campi riconosciuti sono (tra parentesi l'indice da poter usare in alternativa al nome):

alpha	(1)	angolo di rotazione orizzontale della vista

beta (2) angolo di rotazione verticale della vista

section_view (3) se **true** viene impostata una vista in sezione

workplane (4) se **section_view** è true, specifica il tipo di piano su cui impostare la sezione, uno tra i valori enumerati:

- **easywall.plane_xy**
- **easywall.plane_yz**
- **easywall.plane_xz**
- **easywall.plane_generic**

workplane_elevation (5) se **section_view** è **true**, specifica l'elevazione del piano di sezione in base al tipo indicato da **workplane**

show_mode (6) codice della rappresentazione voluta, uno tra i seguenti valori enumerati:

easywall.render_undefined	rappresentazione standard in wireframe
easywall.render_hidden_lines_removal	rappresentazione a linee nascoste
easywall.render_solid	rappresentazione con rendering solido
easywall.render_walls	rappresentazione delle pareti
easywall.render_membrane_isostresses	rappresentazione degli isosforzi membranali
easywall.render_shear_isostresses	rappresentazione degli isosforzi di taglio
easywall.render_flexural_isostresses	rappresentazione degli isosforzi flessionali
easywall.render_membrane_stress_directions	rappresentazione delle direzioni principali membranali
easywall.render_flexural_stress_directions	rappresentazione delle direzioni principali flessionali
easywall.render_sls_cracking	rappresentazione della fessurazione allo stato limite di servizio

easywall.render_sls_stress rappresentazione delle tensioni allo stato limite di servizio

easywall.render_elements_check rappresentazione della verifica degli elementi

Si rimanda al manuale di EasyWall per la descrizione delle rappresentazioni disponibili.

diagrams_scale (7) fattore di scala per i diagrammi di sollecitazione

show_scale (8) fattore di scala per la rappresentazione specifica

show_envelopes (9) se **true** mostra i diagrammi di inviluppo

show_solid (10) se **true** mostra gli elementi selezionati come solidi

shown_bars (11) specificare 0 per non mostrare barre di armatura, altrimenti specificare la somma tra uno o più dei seguenti valori enumerati per mostrare le barre corrispondenti:

easywall.bars_visible_x armatura visibile lungo x

easywall.bars_visible_y armatura visibile lungo y

easywall.bars_hidden_x armatura nascosta lungo x

easywall.bars_hidden_y armatura nascosta lungo y

easywall.bars_main armatura diffusa

easywall.bars_secondary armatura aggiuntiva

show_names (12) se **true** mostra la numerazione degli elementi

shown_index (13) indica di quale elemento mostrare l'indice (0: tutti; -1: nessuno)

sls_max_crack_width_rare (14) massima ampiezza delle fessure per la verifica a fessurazione nelle combinazioni rare

sls_max_crack_width_frequent (15) massima ampiezza delle fessure per la verifica a fessurazione nelle combinazioni frequenti

sls_max_crack_width_almost_permanent (16) massima ampiezza delle fessure per la verifica a fessurazione nelle

		combinazioni quasi permanenti
sls_cracks_visible_face_shown	(17)	se **true** mostra la faccia visibile nella verifica a fessurazione, altrimenti mostra la faccia nascosta
sls_comb_rare	(18)	se **true** attiva la combinazione rara allo stato limite di servizio nella verifica a fessurazione
sls_comb_frequent	(19)	se **true** attiva la combinazione frequente allo stato limite di servizio nella verifica a fessurazione
sls_comb_almost_permanent	(20)	se **true** attiva la combinazione frequente allo stato limite di servizio nella verifica a fessurazione
geo_drained	(21)	se **true** attiva la rappresentazione delle verifiche geotecniche in condizione drenata, altrimenti non drenata.

Interazione con Elwood

Il motore di scripting mette a disposizione alcune funzioni di interrogazione diretta ad Elwood per dati e risultati che non possono essere letti direttamente da file attraverso le funzioni della libreria **sap** descritta nel capitolo precedente.

Le funzioni della libreria **elwood** qui descritta si basano su uno stato di computazione modificato attraverso le funzioni di tip **use_*** ed interrogato attraverso le funzioni **get_***. Ved. il capitolo precedente "Funzioni di accesso ai file SAP" per maggiori informazioni.

elwood.get_app_info( )

Restituisce una tabella con informazioni sintetiche sull'applicazione. La tabella ha i seguenti campi:

manufacturer	il nome del produttore, Softing s.r.l.
name	il nome dell'applicazione
version	la versione dell'applicazione
license	il numero di licenza dell'utente

elwood.get_combinations(t)

Restituisce una tabella che descrive le combinazioni di carico impostate in Elwood: ogni elemento della tabella è una espressione letterale della combinazione, ad es. "*1.00 * permanente + 0.30 * accidentale*".

Il parametro *t* può assumere uno tra i seguenti valori enumerati:

elwood.combinations_design	per ottenere le combinazioni di progetto
elwood.combinations_service	per ottenere le combinazioni di esercizio
elwood.combinations_damage	per ottenere le combinazioni di danno
elwood.combinations_operativity	per ottenere le combinazioni di operatività
elwood.combinations_exceptional	per ottenere le combinazioni eccezionali

elwood.get_load_types_names( )

Restituisce una tabella che elenca i tipi di carico definiti in Elwood.

elwood.get_load_type_data(n)

Restituisce le impostazioni relative al tipo di carico con il nome richiesto.

Viene restituita una tabella avente i seguenti campi:

type	tipo assegnato dall'utente; uno tra i seguenti valori enumerati:	
	elwood.loadtype_permanent	permanente
	elwood.loadtype_variable	variabile
	elwood.loadtype_not_concurrent_variable	variabile non contemporaneo
	elwood.loadtype_seismic	sismico
	elwood.loadtype_seismic_related	simico correlato
gravitational	**true** se deriva da peso proprio	
gamma_unf	valore del coefficiente gamma	
gamma_seismic_unf	valore del coefficiente gamma sismico	
psi_0	valore del coefficiente psi zero	

psi_1 valore del coefficiente psi uno

psi_2 valore del coefficiente psi due

psi_2_seismic valore del coefficiente psi due sismico

phi valore del coefficiente di correlazione

elwood.get_user_views_names( )

Restituisce una tabella contenente i nomi delle viste utente definite per Elwood nel file corrente.

elwood.append_user_view(s)

Inserisce nel documento di uscita un'immagine generata da Elwood per la vista utente specificata nel parametro *s*.

elwood.add_param_view_layer(s, type)

Imposta la visibilità del layer s per le seguenti generazioni di viste parametriche da Elwood con **easywall.append_param_view**.

Il parametro *type* indica il tipo di visibilità del layer specificato e può assumere uno tra i seguenti valori enumerati:

elwood.layer_visible il layer sarà visibile

elwood.layer_invisible il layer non sarà visibile

elwood.layer_grayed gli elementi del layer saranno mostrati in grigio

Le opzioni di visibilità dei layer possono essere resettate con la funzione **elwood.clear_param_view_layers**. Quando nessuna opzione è specificata, tutti i layer sono considerati visibili.

elwood.clear_param_view_layers( )

Azzera le specifiche dei layer precedentemente impostate con **elwood.add_param_view_layer** per la generazione programmatica delle viste in Elwood.

elwood.add_param_view_elements(idx1, idx2)

Aggiunge gli elementi con indice compreso nell'intervallo tra *idx1* ed *idx2* alla selezione corrente di Elwood per la generazione programmatica della vista attraverso la funzione **elwood.append_param_view.**

La selezione degli elementi può essere resettata con la funzione **elwood.clear_param_view_elements**. Quando nessuna

selezione è specificata, tutti gli elementi sono considerati selezionati.

elwood.clear_param_view_elements( )

Azzera la selezione degli elementi precedentemente impostata con **elwood.add_param_view_elements** per la generazione programmatica delle viste in Elwood.

elwood.append_param_view(w, h)

Inserisce nel documento di uscita un'immagine generata da Elwood in base a parametri predefiniti. I parametri *w* e *h* sono rispettivamente la larghezza e l'altezza dell'immagine in centimetri.

Un terzo parametro opzionale può essere una tabella contenente le impostazioni per la generazione dell'immagine. I campi che formano la tabella possono avere indice numerico o letterale indifferentemente per un più rapido accesso ai singoli parametri. I campi riconosciuti sono (tra parentesi l'indice da poter usare in alternativa al nome):

alpha (1) angolo di rotazione orizzontale della vista

beta (2) angolo di rotazione verticale della vista

section_view (3) se **true** viene impostata una vista in sezione

workplane (4) se **section_view** è true, specifica il tipo di piano su cui impostare la sezione, uno tra i valori enumerati:

- **elwood.plane_xy**
- **elwood.plane_yz**
- **elwood.plane_xz**
- **elwood.plane_generic**

workplane_elevation (5) se **section_view** è **true**, specifica l'elevazione del piano di sezione in base al tipo indicato da **workplane**

show_code (6) codice della rappresentazione voluta, uno tra i seguenti valori enumerati:

- **elwood.render_none** rappresentazione standard in wireframe
- **elwood.render_stress_envelope** rappresentazione delle tensioni flessionali sull'involucro dell'elemento
- **elwood.render_tau_envelope** rappresentazione delle tensioni di taglio sull'involucro dell'elemento

elwood.render_torsional_envelope	rappresentazione delle tensioni da torsione sull'involucro dell'elemento
elwood.render_instability_envelope	rappresentazione delle tensioni da instabilità sull'involucro dell'elemento
elwood.render_deflection_envelope	rappresentazione del fattore (costante su tutto l'elemento) di sfruttamento deformabilità
elwood.render_summary_envelope	rappresentazione del fattore (costante su tutto l'elemento) di sfruttamento massimo
elwood.render_sld	rappresentazione del fattore di sfruttamento allo stato limite di danno

Si rimanda al manuale di Elwood per la descrizione delle rappresentazioni disponibili.

diagrams_scale (7) fattore di scala per i diagrammi di sollecitazione