Request Dispatching

La classe **controller.php** del framework **itaEngine** svolge la funzione di dispatcher: ad ogni evento della pagina corrisponde una chiamata ajax che invia al server diverse informazioni, alcune indispensabili per identificare la classe ed il metodo con cui quest'ultimo dovrà rispondere all'evento:

model	Nome model form (Es. cwbBtaGrunaz)
event	Evento scatenante (Es. onClick)
id	Componente scatenante (Es. cwbBtaGrunaz_Nuovo)
nameform	Nome instanza (solitamente corrisponde con il nome del model

La richiesta verrà inoltrata ad un model form, la cui classe php è situata obbligatoriamente all'interno della cartella **apps**; per quanto riguarda la sottocartella, invece, occorre vedere le prime tre lettere di model, e vedere il valore corrispondente presente nella classe **lib/AppDefinitions/AppDefinitions.php**. Nell'esempio, il prefisso **cwb** corrisponde a **CityBase**, pertanto il file php che risponde alla richiesta, per modelform=cwbBtaGrunaz, è **apps/CityBase/cwbBtaGrunaz.php**

Casi d'uso

Le tipologie di finestre che possono essere gestite con il framework possono essere raggruppate secondo questa classificazione:

Tipologia	Descrizione
Tabella di gestione CRUD	Consente di effettuare le classiche operazioni di ricerca, inserimento, aggiornamento, cancellazione su una determinata tabella mediante una rappresentazione tabellare. E' composta da tre elementi: ricerca - paginazione - dettaglio.
Albero	Come sopra, solo che la rappresentazione dei dati, anzichè essere tabellare, è fatta utilizzando una “TreeTable”. Utilizzare quando la tabella presenta una struttura gerarchica.
Mono-record	Consente di inserire/modificare l'unico record presente in tabella. Sulla base di questo, la struttura, rispetto ad una tabella di gestione tradizionale, è semplificata, in quanto non è presente la paginazione con una griglia di risultati, ma si accede direttamente al dettaglio (se il record non esiste, si effettuerà un inserimento, altrimenti si andrà in aggiornamento. In questo tipo di gestione non è prevista la cancellazione.
Finesta “fuori standard”	Rientrano in questa casistica tutte le altre casistiche (console, monitor, programmi di stampa, programmi di elaborazione). Queste finestre, essendo tutte diverse tra di loro, avranno una gestione propria, che va vista caso per caso.

Tabella classica di gestione (CRUD) semplice

Esempio: **cwbBtaGrunaz**

Metodi da specificare (obbligatori)

Metodo	Descrizione
initVars	Valorizzare le variabili: **GRID_NAME**, **AUTOR_NAME**, **AUTOR_NUMERO**, **searchOpenElenco** e **libDB**
postSqlElenca	Valorizzare la variabile **SQL** utilizzata per caricare l'elenco di paginazione (Utilizzare un metodo in una lib, è buona norma non scrivere l'istruzione SQL all'interno della classe di gestione della form
sqlDettaglio	Valorizzare la variabile **SQL** utilizzata per caricare il singolo record (Utilizzare un metodo in una lib, è buona norma non scrivere l'istruzione SQL all'interno della classe di gestione della form

Metodi da specificare (facoltativi)

Metodo	Descrizione
postApriForm	Inizializzazione combo di gestione; impostazione del focus sul campo desiderato.
postAltraRicerca	Impostazione del focus sul campo desiderato.
postNuovo	In questo caso, sblocca il campo chiave perchè deve essere specificato manualmente; impostazione del focus sul campo desiderato.
postAggiungi	Impostazione del focus sul campo desiderato.
setGridFilters	Impostazione dei filtri provenienti dalle colonne della grid.
elaboraRecords	Formatta l'output della grid.

Tabella classica di gestione (CRUD) complessa

TODO: completare

Albero

Esempio: **cwbBorOrgan**

Metodi da specificare (obbligatori)

Metodo	Descrizione
initVars	Valorizzare le variabili: **GRID_NAME**, **AUTOR_NAME**, **AUTOR_NUMERO**, **searchOpenElenco** e **libDB**
caricaNodiPrimoLivello	Effettua il caricamento dei soli nodi al primo livello. Demandare l'operazione di caricamento ad una apposita lib.
caricaFigli	Effettua il caricamento dei figli, dato in ingresso l'identificativo del nodo padre. Demandare l'operazione di caricamento ad una apposita lib.
getLivello	Restituisce il livello del nodo corrente dell'albero.
caricaAlbero	Effettua il caricamento dell'albero, applicando i filtri di ricerca
caricaGerarchiaNodo	Effettua il caricamento di tutti i nodi padre relativi al nodo passato come parametro in ingresso
cercaPadre	Restituisce il record padre del record selezionato
postElaboraNodiCaricati	Effettua un ordinamento dell'array

Metodo	Descrizione
livelloDaNodo	Restituisce il livello del nodo specificato in ingresso
parentDaAlbero	Restituisce il codice padre relativo al nodo specificato in ingresso
getFoglia	Restituisce TRUE se il nodo passato come parametro è una foglia, altrimenti FALSE
expandedDaCaricamento	Controlla se presenti figli nell'albero caricato per il nodo corrente (se si, restituisce TRUE, altrimenti FALSE)

Metodi da specificare (facoltativi)

Metodo	Descrizione
postApriForm	Inizializzazione combo di gestione; impostazione del focus sul campo desiderato.
postAltraRicerca	Impostazione del focus sul campo desiderato.
postNuovo	In questo caso, sblocca il campo chiave perchè deve essere specificato manualmente; impostazione del focus sul campo desiderato.
postAggiungi	Impostazione del focus sul campo desiderato.
setGridFilters	Impostazione dei filtri provenienti dalle colonne della grid.
elaboraRecords	Formatta l'output della grid.

Finestra mono-record

Esempio: **cwbBgdParott**

Metodi da specificare (obbligatori)

Metodo	Descrizione
initVars	Valorizzare le variabili: **GRID_NAME**, **AUTOR_NAME**, **AUTOR_NUMERO**, **searchOpenElenco** e **libDB**
sqlDettaglio	Valorizzare la variabile **SQL** utilizzata per caricare il singolo record (Utilizzare un metodo in una lib, è buona norma non scrivere l'istruzione SQL all'interno della classe di gestione della form

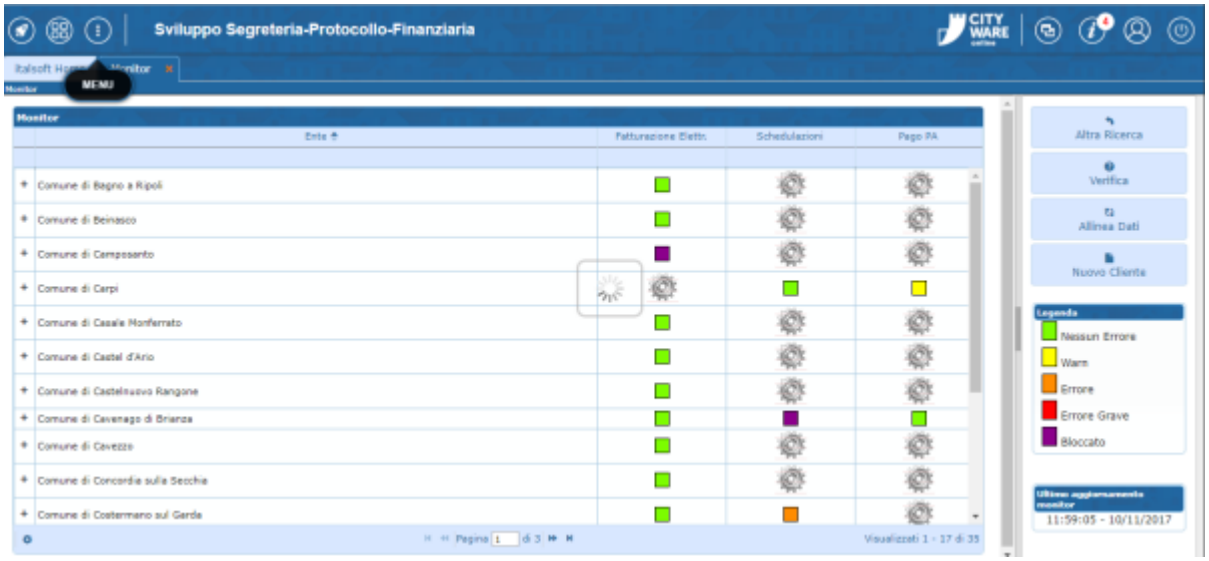
Metodi da specificare (facoltativi)

Metodo	Descrizione
postApriForm	Inizializzazione combo di gestione; impostazione del focus sul campo desiderato.
postAltraRicerca	Impostazione del focus sul campo desiderato.
postNuovo	In questo caso, sblocca il campo chiave perchè deve essere specificato manualmente; impostazione del focus sul campo desiderato.
postAggiungi	Impostazione del focus sul campo desiderato.
setGridFilters	Impostazione dei filtri provenienti dalle colonne della grid.

Finestre fuori standard

Monitor Eventi

Esempio: **cwbBgeMonitor**



Questo caso è abbastanza simile ad una finestra CRUD, infatti la classe estende la superclasse **cwbBpaGenTab**. Una particolarità importante è che il metodo **elenca** è abbastanza differente da quello standard, per questo motivo è stato completamente sovrascritto.

Console Diagnostica

Esempio: **cwbDiagnostica**

In questo caso, la classe estende semplicemente **itaFrontController** in quanto non deve lavorare con il database, ma lanciare dei test per una diagnostica di secondo livello. Anche a livello di generator la struttura della pagina sarà molto diversa da una pagina CRUD tradizionale. L'unico punto in comune con le CRUD può essere quindi solamente il **parseEvent**.

Finestra mono-record con tabella senza la chiave primaria

TODO: descrivere il caso specifico

Superclassi

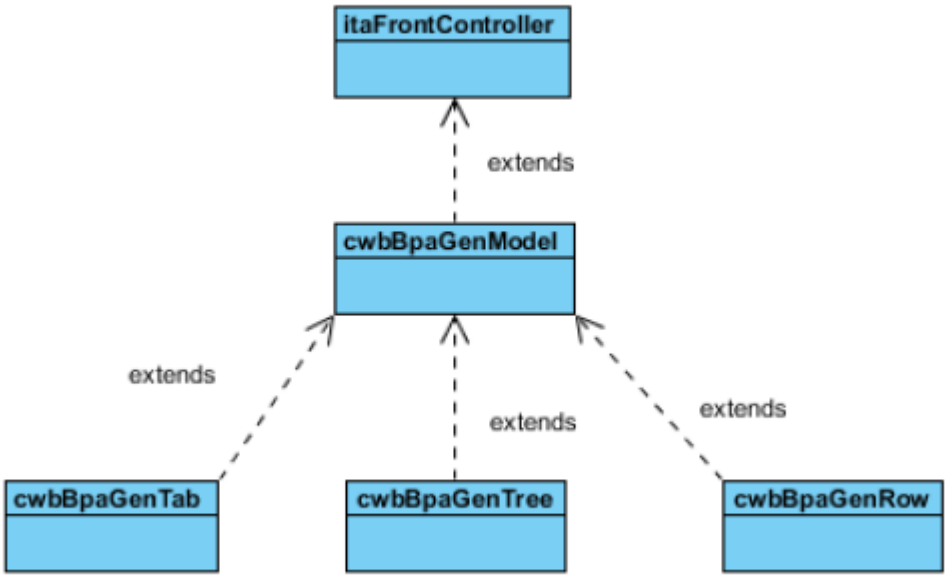
Per soddisfare la richiesta, una classe model form deve estendere obbligatoriamente una delle due classi sotto descritta:

itaModel	Superclasse originale presente nel framework. Utilizzata in diversi applicativi.
itaFrontController	Superclasse introdotta con la modifica del multi-database e dei service. Utilizzata negli applicativi City*

In questa sezione verrà descritto il filone di itaFrontController.

Gerarchia

Diagramma UML:



itaFrontController

Variabili private principali

nameForm	nome grafico della form, può essere sostituito con un alias nel caso di finestre che richiamano se stesso (es btaVieOriginali)
nameFormOrig	Nome form originale, da usare per accedere al modello (es btaVie)

wizardParameters	Serve per passare dei parametri tra una form e l'altra nel caso venga usata all'interno di un wizard (2)
TABLE_NAME	Nome tabella
TABLE_VIEW	Nome vista usata per la ricerca
modelService	Oggetto modelService per operazioni di inserimento-modifica-cancellazione su database
helper	Oggetto helper (serve per accentrare operazioni comuni tra gli stelli livelli della gerarchia)
CITYWARE_DB	Connessione al database Cityware (1)
GRID_NAME	Nome jqGrid (2)
AUTOR_MODULO	Check autorizzazioni: modulo da controllare (3)
AUTOR_NUMERO	Check autorizzazioni: numero autorizzazione da controllare (3)
formData	Dati passati alla form, filtrati e bonificati dall'array $_POST
returnModel	Modello di rientro dai lookup
returnNameForm	nameform di rientro dai lookup (in caso di alias è diverso da model)
returnEvent	event di rientro dai lookup
returnId	id che ha causato il rientro di rientro dai lookup

NOTE:
(1) = TODO: Fare refactor del nome della variabile.
(2) = Questa variabile non dovrebbe essere posizionata in questo livello gerarchico, ma in un sottolivello. Per ora è stato messo qui per comodità, in modo da avere la variabile sempre a disposizione
(3) = TODO: Spostare queste variabili, in modo da rendere la classe agnostica rispetto a Cityware.

Variabili utilizzate in sessione

Ci sono alcune variabili che vengono scritte in sessione al distruttore della classe, e rilette dalla sessione alla request successiva. Sono le seguenti:

- perms
- formData
- modelConfig
- returnModel
- returnNameForm
- returnEvent
- returnId
- procObj
- eqAudit

Utilizzo del modelService

Metodo	Descrizione	Esempio
initModelService	Qui è dove viene istanziato il modelService	$this→modelService = itaModelServiceFactory::newModelService($this→nameFormOrig);

Metodo	Descrizione	Esempio
insertAudit	Inserimento evento su tabella di audit di sistema	$this→modelService→insertAudit($DB, $tableName, $data, $opCode);
openRecord	Apertura record	$this→modelService→openRecord($DB, $tableName, $data);
insertRecord	Inserimento record	$this→modelService→insertRecord($DB, $tableName, $data, $record_Info);
updateRecord	Aggiornamento record	$this→modelService→updateRecord($DB, $tableName, $data, $record_Info);
deleteRecord	Cancellazione record	$this→modelService→deleteRecord($DB, $tableName, $dataPk, $record_Info);
getLastInsertId	Recupera ultimo id inserito	$this→modelService→getLastInsertId();
setLastInsertId	Imposta ultimo id inserito (in caso di gestione progressivo calcolato manualmente)	$this→modelService→setLastInsertId($lastInsertId);

Utilizzo del modelService all'interno di una lib

Il modelService può essere utilizzato in qualsiasi punto dell'applicazione, comprese le lib. Per prima cosa, occorre istanziarlo, quetso è possibile farlo attraverso la factory **cwbModelServiceFactory**:

```
$modelService = cwbModelServiceFactory::newModelService($modelName);
```

se non si conosce il nome del model, ma solo della table, è possibile risalire all'informazione utilizzando il metodo **modelNameByTableName** di **cwbModelHelper**:

```
$modelService =
cwbModelServiceFactory::newModelService(cwbModelHelper::modelNameByTableName
($tableName));
```

Eventi gestiti nel metodo parseEvent

- openform
- openportlet
- startProcess
- endProcess
- broadcastMsg
- refreshProcess

cwbBpaGenModel

Variabili private principali

CURRENT_RECORD	Array che contiene il record corrente
RECORD_INFO	Array che contiene le informazioni sul record per audit
PK	Array chiavi primaria
ALIAS	Alias finestra (per istanze multiple)
LOCK	Array che identifica un lock logico
libDB	Libreria Database
gridFilters	Array che contiene i filtri sulla jqGrid
detailView	Libreria Database
filtriFissi	Array di filtri fissi
hasSequence	true se la tabella ha la sequence
modelData	Dati da passare al ModelService per validazione/salvataggio
noCrud	indica se la pagina deve gestire o meno le operazioni di crud
SQL	
masterRecord	Record principale (per master-detail)
flagSearch	Indica se la finestra è stata chiamata in ricerca
externalParams	Array che contiene i parametri esterni
searchOpenElenco	Indica se in fase di search la finestra deve presentare subito l'elenco
authenticator	Oggetto Authenticator
printStrategy	Print Strategy (Omnis Report/Jasper)
operationsData	Operazioni (U/D, la insert non va specificata, se è senza pk inserisco diretto) effettuate sulle tabelle relazionate (array('nameForm' ⇒ array('relationTableName' ⇒ array('operation' ⇒ 'I/U/D','guuid' ⇒ 'xxx', 'pks'⇒ array('key'⇒'value'))))
operationMapping	contiene i riferimenti con il data su $operationsData
skipAuth	se true salta il controllo sulle autorizzazioni
omnisReportName	Nome report Omnis
openDetailFlag	Flag che indica se deve segnalare o meno l'apertura di una pagina di dettaglio.
actionAfterNew	Prende una costante GOTO_*, indica cosa fare dopo aver creato un nuovo elemento.
actionAfterModify	Prende una costante GOTO_*, indica cosa fare dopo aver modificato un elemento.
actionAfterDelete	Prende una costante GOTO_*, indica cosa fare dopo aver eliminato un elemento.

Variabili utilizzate in sessione

- flagSearch
- detailView
- externalParams
- autorLevel
- stopEvent
- operationsData

Eventi gestiti nel metodo parseEvent

Evento	Descrizione	Comportamento
openForm	Apertura della form	Se si tratta di una gestione master-detail, e alla form corrente viene passato il **masterRecord**, richiama il metodo **elenca**. Nel caso di una gestione tradizionale, viene richiamato il metodo **apriForm** (se presenti dei filtri esterni - **externalParams** valorizzato - prima di chiamare **apriForm** lancia il metodo **initExternalFilter**.
dbClickRow	Doppio click sulla griglia di paginazione dei risultati	Se la finestra è stata richiamata da binocolo, lancia il metodo **ricercaEsterna** in modo da poter effettuare una lookup. Se invece la finestra è stata richiamata in modo tradizionale, ricade nella stessa casistica di **editGridRow**.
editGridRow	Click su pulsante modifica della griglia di paginazione	Lancia il metodo **dettaglio**, che imposta la finestra nel pannello di dettaglio del record selezionato.
addGridRow	Click su pulsante aggiungi della griglia di paginazione	Lancia il metodo **nuovo**, che imposta la finestra nel pannello di dettaglio del record selezionato, predisponendosi per l'inserimento di un nuovo record.
delGridRow	Click su pulsante elimina della griglia di paginazione	Lancia il metodo **cancella**, che mostra a video una dialog di conferma per la cancellazione del record selezionato.
onClick	Click su un qualsiasi componente della pagina “cliccabile”	Nella superclasse vengono gestiti i pulsanti della buttonbar che sono presenti in tutte le finestre.
onClickTablePager	Click su pulsante refresh della griglia	Lancia il metodo **elenca** per caricare i record, in funzione dei filtri impostati. N.B.: Questo evento scatta automaticamente dopo il primo ridimensionamento grafico della pagina, in modo da calcolare esattamente il numero di record che possono essere effettivamente visualizzati nella griglia, in funzione delle dimensioni dello schermo.
printTableToHTML	Click su pulsante stampa della griglia	Lancia il metodo **stampaElenco**. La stampa viene effettuata in funzione di **printStrategy** (Omnis/Jasper).
exportTableToExcel	Click su pulsante esporta in Excel della griglia	Lancia il metodo **exportXLS**.

Click su ButtonBar

Nella superclasse sono sttai gestiti i pulsanti presenti in tutte le form:

Nome Pulsante	Descrizione	Metodo corrispondente
Nuovo	Preparazione per creazione nuovo record	**nuovo**
Aggiungi	Inserimento nuovo record	**aggiungi**
Aggiorna	Aggiornamento nuovo record	**aggiorna**
Cancella	Richiede conferma per cancellazione record selezionato	**cancella**

Nome Pulsante	Descrizione	Metodo corrispondente
ConfermaCancella	Cancella record	**confermaCancella**
Elenca	Effettua ricerca e mostra i risultati nella griglia	**elenca**
AltraRicerca	Imposta la pagina sui filtri, per cosentire di effettuare una nuova ricerca	**altraRicerca**
Torna	Ritorna a elenco risultati	**tornaAElenco**
CaricaParametriRicerca	Effettua il caricamento dei parametri di ricerca da database	**caricaParametriRicerca**
SalvaParametriRicerca	Effettua il salvataggio dei parametri di ricerca su database	**salvaParametriRicerca**
ResetParametriRicerca	Pulisce i parametri di ricerca a video	**resetParametriRicerca**
ConfermaWarning	Conferma i messaggi di warning	A seconda dell'operazione effettuata in precedenza, lancia il metodo giusto: nel caso di inserimento, lancia **aggiungi**; se aggiornamento, lancia **aggiorna**; se cancellazione, lancia **confermaCancella**

Azioni e punti di personalizzazione

Di seguito, un elenco di tutti i metodi che possono essere sovrascritti al bisogno nelle sottoclassi, raggruppati per funzione.

Nuovo

Metodo	Descrizione	Quando utilizzarlo
preNuovo	Viene lanciato all'inizio del metodo, prima di effettuare qualsiasi operazione.	Pulizia dei campi a video, inizializzazione di combo box
postNuovo	Viene lanciato alla fine del metodo.	Aggiustamenti grafici dei campi a video (Es. campi readonly, applicazioni di classi/stili css), impostazione del focus sul campo desiderato

Aggiungi

Metodo	Descrizione	Quando utilizzarlo
preAggiungi	Viene lanciato all'inizio del metodo, prima di effettuare qualsiasi operazione	Normalizzazione valori $_POST, controlli aggiuntivi, inizio transazione manuale (casistiche particolari, vedi cwbBtaSogg)

Aggiorna

Metodo	Descrizione	Quando utilizzarlo
preAggiorna	Viene lanciato all'inizio del metodo, prima di effettuare qualsiasi operazione	Vedi preAggiungi

Conferma Cancella

Metodo	Descrizione	Quando utilizzarlo
preConfermaCancella	Viene lanciato all'inizio del metodo, prima di effettuare qualsiasi operazione	Inizio transazione manuale (casistiche particolari, vedi cwbBtaSogg)

Elenca

Il metodo deve essere implementato nelle sottoclassi.

Altra Ricerca

Metodo	Descrizione	Quando utilizzarlo
preAltraRicerca	Viene lanciato dopo lo sblocco della chiave	Impostazione autocomplete campi, per casistiche particolari (vedi cwbBorOrgan)
postAltraRicerca	Viene lanciato alla fine del metodo	Impostazione del focus sul campo di ricerca desiserato

Torna A Elenco

Metodo	Descrizione	Quando utilizzarlo
preTornaElenco	Viene lanciato dopo lo sblocco della chiave	Previsto per casi particolari, mai utilizzato
postTornaElenco	Viene lanciato alla fine del metodo	Visualizzazione pulsanti custom nella button bar

Carica parametri di ricerca

Metodo	Descrizione	Quando utilizzarlo
postCaricaParametriRicerca	Viene lanciato alla fine del metodo	Previsto per casi particolari, mai utilizzato

Salva parametri di ricerca

Metodo	Descrizione	Quando utilizzarlo
postSalvaParametriRicerca	Viene lanciato alla fine del metodo	Previsto per casi particolari, mai utilizzato

Reset parametri di ricerca

Metodo	Descrizione	Quando utilizzarlo
postResetParametriRicerca	Viene lanciato alla fine del metodo	Previsto per casi particolari, mai utilizzato

cwbBpaGenTab

La superclasse è specializzata per caricare i risultati su una datatable, accentrando le logiche comuni. I metodi specializzati sono:

Metodo	Descrizione
elenca	Si occupa del caricamento dei dati.
initializeTable	Si occupa dell'inizializzazione del componente jqGrid
getDataPage	Restituisce risultati dopo un'elaborazione della griglia
elaboraGrid	Formatta risultati provenienti dal caricamento da presentare a video
setSortParameter	Imposta parametri per ordinamento

Azioni e punti di personalizzazione

Di seguito, un elenco di tutti i metodi che possono essere sovrascritti al bisogno nelle sottoclassi, raggruppati per funzione.

elenca

Metodo	Descrizione	Quando utilizzarlo
preElenca	Viene lanciato dopo l'impostazione dei filtri sulla griglia (metodo: setGridFilters)	Effettua delle operazioni grafiche particolari (vedere ad esempio cwfFtaPagam)
postElenca	Viene lanciato alla fine del metodo.	Visualizzazione dei pulsanti particolari nella button bar

elaboraGrid

Metodo	Descrizione	Quando utilizzarlo
elaboraRecords	Viene lanciato all'interno di elaboraGrid. Al metodo viene passato l'array dei risultati, che è possibile manipolare in modo da poter ottenere l'output desiderato.	Inserimento di icone, applicazioni css a determinate colonne, concatenazione di più campi da mostrare in una singola colonna.

cwbBpaGenRow

La superclasse è specializzata per la gestione di tabelle con un solo record. Graficamente queste si presentano direttamente in dettaglio, e gestiscono l'inserimento e l'aggiornamento in modo automatico, a seconda della presenza o meno del record.

Metodo	Descrizione
elenca	Se il record è già presente sul database, lancia il metodo **dettaglio**, altrimenti lancia il metodo **nuovo** (per effettuare il controllo, si basa sulla variabile **CURRENT_RECORD**

Azioni e punti di personalizzazione

Vedi **cwbBpaGenTable** per quanto riguarda il metodo **elenca**.

cwbBpaGenTree

TODO: completare

Helper

TODO: completare