

# 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:

<b>model</b>	Nome model form (Es. cwbBtaGrunaz)
<b>event</b>	Evento scatenante (Es. onClick)
<b>id</b>	Componente scatenante (Es. cwbBtaGrunaz_Nuovo)
<b>nameform</b>	Nome istanza (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
<b>Tabella di gestione CRUD</b>	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.
<b>Albero</b>	Come sopra, solo che la rappresentazione dei dati, anzichè essere tabellare, è fatta utilizzando una "TreeTable". Utilizzare quando la tabella presenta una struttura gerarchica.
<b>Mono-record</b>	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.
<b>Finestra "fuori standard"</b>	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
<b>initVars</b>	Valorizzare le variabili: <b>GRID_NAME</b> , <b>AUTOR_NAME</b> , <b>AUTOR_NUMERO</b> , <b>searchOpenElenco</b> e <b>libDB</b>
<b>postSqlElenco</b>	Valorizzare la variabile <b>SQL</b> 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)
<b>sqlDettaglio</b>	Valorizzare la variabile <b>SQL</b> 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
<b>postApriForm</b>	Inizializzazione combo di gestione; impostazione del focus sul campo desiderato.
<b>postAltraRicerca</b>	Impostazione del focus sul campo desiderato.
<b>postNuovo</b>	In questo caso, sblocca il campo chiave perchè deve essere specificato manualmente; impostazione del focus sul campo desiderato.
<b>postAggiungi</b>	Impostazione del focus sul campo desiderato.
<b>setGridFilters</b>	Impostazione dei filtri provenienti dalle colonne della grid.
<b>elaboraRecords</b>	Formatta l'output della grid.

## Tabella classica di gestione (CRUD) complessa

TODO: completare

## Albero

TODO: completare

## Finestra mono-record

Esempio: **cwbBgdParott**

## Metodi da specificare (obbligatori)

Metodo	Descrizione
<b>initVars</b>	Valorizzare le variabili: <b>GRID_NAME</b> , <b>AUTOR_NAME</b> , <b>AUTOR_NUMERO</b> , <b>searchOpenElenco</b> e <b>libDB</b>
<b>sqlDettaglio</b>	Valorizzare la variabile <b>SQL</b> 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
<b>postApriForm</b>	Inizializzazione combo di gestione; impostazione del focus sul campo desiderato.
<b>postAltraRicerca</b>	Impostazione del focus sul campo desiderato.
<b>postNuovo</b>	In questo caso, sblocca il campo chiave perchè deve essere specificato manualmente; impostazione del focus sul campo desiderato.
<b>postAggiungi</b>	Impostazione del focus sul campo desiderato.
<b>setGridFilters</b>	Impostazione dei filtri provenienti dalle colonne della grid.

## Finestre fuori standard

TODO: gestire le varie casistiche ....

## Superclassi

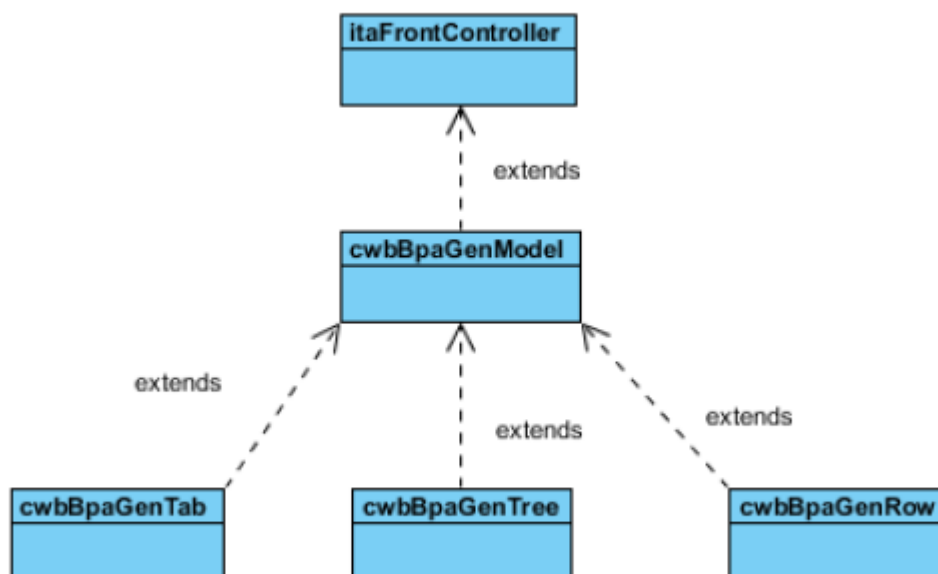
Per soddisfare la richiesta, una classe model form deve estendere obbligatoriamente una delle due classi sotto descritta:

<b>itaModel</b>	Superclasse originale presente nel framework. Utilizzata in diversi applicativi.
<b>itaFrontController</b>	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

<b>nameForm</b>	nome grafico della form, può essere sostituito con un alias nel caso di finestre che richiamano se stesso (es btaVieOriginali)
<b>nameFormOrig</b>	Nome form originale, da usare per accedere al modello (es btaVie)
<b>wizardParameters</b>	Serve per passare dei parametri tra una form e l'altra nel caso venga usata all'interno di un wizard (2)
<b>TABLE_NAME</b>	Nome tabella
<b>TABLE_VIEW</b>	Nome vista usata per la ricerca
<b>modelService</b>	Oggetto modelService per operazioni di inserimento-modifica-cancellazione su database
<b>helper</b>	Oggetto helper (serve per accentrare operazioni comuni tra gli stelli livelli della gerarchia)
<b>CITYWARE_DB</b>	Connessione al database Cityware (1)
<b>GRID_NAME</b>	Nome jqGrid (2)
<b>AUTOR_MODULO</b>	Check autorizzazioni: modulo da controllare (3)
<b>AUTOR_NUMERO</b>	Check autorizzazioni: numero autorizzazione da controllare (3)
<b>formData</b>	Dati passati alla form, filtrati e bonificati dall'array \$_POST
<b>returnModel</b>	Modello di rientro dai lookup
<b>returnNameForm</b>	nameform di rientro dai lookup (in caso di alias è diverso da model)
<b>returnEvent</b>	event di rientro dai lookup
<b>returnId</b>	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
<b>initModelService</b>	Qui è dove viene istanziato il modelService	<code>\$this-&gt;modelService = itaModelServiceFactory::newModelService(\$this-&gt;nameFormOrig);</code>
<b>insertAudit</b>	Inserimento evento su tabella di audit di sistema	<code>\$this-&gt;modelService-&gt;insertAudit(\$DB, \$tableName, \$data, \$opCode);</code>
<b>openRecord</b>	Apertura record	<code>\$this-&gt;modelService-&gt;openRecord(\$DB, \$tableName, \$data);</code>
<b>insertRecord</b>	Inserimento record	<code>\$this-&gt;modelService-&gt;insertRecord(\$DB, \$tableName, \$data, \$record_Info);</code>
<b>updateRecord</b>	Aggiornamento record	<code>\$this-&gt;modelService-&gt;updateRecord(\$DB, \$tableName, \$data, \$record_Info);</code>
<b>deleteRecord</b>	Cancellazione record	<code>\$this-&gt;modelService-&gt;deleteRecord(\$DB, \$tableName, \$dataPk, \$record_Info);</code>
<b>getLastInsertId</b>	Recupera ultimo id inserito	<code>\$this-&gt;modelService-&gt;getLastInsertId();</code>
<b>setLastInsertId</b>	Imposta ultimo id inserito (in caso di gestione progressivo calcolato manualmente)	<code>\$this-&gt;modelService-&gt;setLastInsertId(\$lastInsertId);</code>

## Eventi gestiti nel metodo parseEvent

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

## cwbBpaGenModel

### Variabili private principali

<b>CURRENT_RECORD</b>	Array che contiene il record corrente
<b>RECORD_INFO</b>	Array che contiene le informazioni sul record per audit
<b>PK</b>	Array chiavi primaria
<b>ALIAS</b>	Alias finestra (per istanze multiple)
<b>LOCK</b>	Array che identifica un lock logico
<b>libDB</b>	Libreria Database
<b>gridFilters</b>	Array che contiene i filtri sulla jqGrid
<b>detailView</b>	Libreria Database
<b>filtriFissi</b>	Array di filtri fissi
<b>hasSequence</b>	true se la tabella ha la sequence
<b>modelData</b>	Dati da passare al ModelService per validazione/salvataggio
<b>noCrud</b>	indica se la pagina deve gestire o meno le operazioni di crud
<b>SQL</b>	
<b>masterRecord</b>	Record principale (per master-detail)
<b>flagSearch</b>	Indica se la finestra è stata chiamata in ricerca
<b>externalParams</b>	Array che contiene i parametri esterni
<b>searchOpenElenco</b>	Indica se in fase di search la finestra deve presentare subito l'elenco
<b>authenticator</b>	Oggetto Authenticator
<b>printStrategy</b>	Print Strategy (Omnis Report/Jasper)
<b>operationsData</b>	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'))))
<b>operationMapping</b>	contiene i riferimenti con il data su \$operationsData
<b>skipAuth</b>	se true salta il controllo sulle autorizzazioni
<b>omnisReportName</b>	Nome report Omnis
<b>openDetailFlag</b>	Flag che indica se deve segnalare o meno l'apertura di una pagina di dettaglio.
<b>actionAfterNew</b>	Prende una costante GOTO_*, indica cosa fare dopo aver creato un nuovo elemento.
<b>actionAfterModify</b>	Prende una costante GOTO_*, indica cosa fare dopo aver modificato un elemento.
<b>actionAfterDelete</b>	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
<b>openForm</b>	Apertura della form	Se si tratta di una gestione master-detail, e alla form corrente viene passato il <b>masterRecord</b> , richiama il metodo <b>elenca</b> . Nel caso di una gestione tradizionale, viene richiamato il metodo <b>apriForm</b> (se presenti dei filtri esterni - <b>externalParams</b> valorizzato - prima di chiamare <b>apriForm</b> lancia il metodo <b>initExternalFilter</b> .
<b>dbClickRow</b>	Doppio click sulla griglia di paginazione dei risultati	Se la finestra è stata richiamata da binocolo, lancia il metodo <b>ricercaEsterna</b> in modo da poter effettuare una lookup. Se invece la finestra è stata richiamata in modo tradizionale, ricade nella stessa casistica di <b>editGridRow</b> .
<b>editGridRow</b>	Click su pulsante modifica della griglia di paginazione	Lancia il metodo <b>dettaglio</b> , che imposta la finestra nel pannello di dettaglio del record selezionato.
<b>addGridRow</b>	Click su pulsante aggiungi della griglia di paginazione	Lancia il metodo <b>nuovo</b> , che imposta la finestra nel pannello di dettaglio del record selezionato, predisponendosi per l'inserimento di un nuovo record.
<b>delGridRow</b>	Click su pulsante elimina della griglia di paginazione	Lancia il metodo <b>cancella</b> , che mostra a video una dialog di conferma per la cancellazione del record selezionato.
<b>onClick</b>	Click su un qualsiasi componente della pagina "cliccabile"	Nella superclasse vengono gestiti i pulsanti della buttonbar che sono presenti in tutte le finestre.
<b>onClickTablePager</b>	Click su pulsante refresh della griglia	Lancia il metodo <b>elenca</b> 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.
<b>printTableToHTML</b>	Click su pulsante stampa della griglia	Lancia il metodo <b>stampaElenco</b> . La stampa viene effettuata in funzione di <b>printStrategy</b> (Omnis/Jasper).
<b>exportTableToExcel</b>	Click su pulsante esporta in Excel della griglia	Lancia il metodo <b>exportXLS</b> .

## Click su ButtonBar

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

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

## 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
<b>preNuovo</b>	Viene lanciato all'inizio del metodo, prima di effettuare qualsiasi operazione.	Pulizia dei campi a video, inizializzazione di combo box
<b>postNuovo</b>	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
<b>preAggiungi</b>	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
<b>preAggiorna</b>	Viene lanciato all'inizio del metodo, prima di effettuare qualsiasi operazione	Vedi preAggiungi

### Conferma Cancella

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

### Torna A Elenco

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

### Carica parametri di ricerca

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

### Salva parametri di ricerca

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

## Reset parametri di ricerca

Metodo	Descrizione	Quando utilizzarlo
<b>postResetParametriRicerca</b>	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
<b>elenca</b>	Si occupa del caricamento dei dati.
<b>initializeTable</b>	Si occupa dell'inizializzazione del componente jqGrid
<b>getDataPage</b>	Restituisce risultati dopo un'elaborazione della griglia
<b>elaboraGrid</b>	Formatta risultati provenienti dal caricamento da presentare a video
<b>setSortParameter</b>	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
<b>preElenca</b>	Viene lanciato dopo l'impostazione dei filtri sulla griglia (metodo: setGridFilters)	Effettua delle operazioni grafiche particolari (vedere ad esempio cwfFtaPagam)
<b>postElenca</b>	Viene lanciato alla fine del metodo.	Visualizzazione dei pulsanti particolari nella button bar

### elaboraGrid

Metodo	Descrizione	Quando utilizzarlo
<b>elaboraRecords</b>	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
<b>elenca</b>	Se il record è già presente sul database, lancia il metodo <b>dettaglio</b> , altrimenti lancia il metodo <b>nuovo</b> (per effettuare il controllo, si basa sulla variabile <b>CURRENT_RECORD</b> )

## Azioni e punti di personalizzazione

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

## cwbBpaGenTree

TODO: completare

## Helper

TODO: completare

From:  
<https://wiki.nuvolaitalsoft.it/> - **wiki**

Permanent link:  
[https://wiki.nuvolaitalsoft.it/doku.php?id=sviluppo:cityware\\_superclassi&rev=1510308300](https://wiki.nuvolaitalsoft.it/doku.php?id=sviluppo:cityware_superclassi&rev=1510308300)

Last update: **2018/03/19 10:45**

