**Object Design Document**

Sommario

[**1.** **Introduction** 2](#_Toc27488462)

[**1.1.** **Object design trade-offs** 2](#_Toc27488463)

[**1.2.** **Interface documentation guidelines** 2](#_Toc27488464)

[**1.3.** **Definitions, acronyms, and abbrevations** 2](#_Toc27488465)

[**1.4.** **References** 3](#_Toc27488466)

[**2.** **Packages** 3](#_Toc27488467)

[**3.** **Class interfaces** 3](#_Toc27488468)

# **Introduction**

## **Object design trade-offs**

• **Memory space vs. Response time:**

poiché il nostro sistema EmmeShop di progetto visto che si tratta di un e-commerce dovrà gestire una grande mole di dati per cui abbiamo bisogno di un DBMS che memorizzi tutti i dati, dal carrello dell’utente, ai prodotti e categorie dei relativi negozi. Quindi questo implica scritture e letture in un DB però avendo un tempo di risposta contenuto nei limiti dettati dai requisiti non funzionali nel RAD.

• **Functionality vs. Usability**

Il nostro software incorpora funzionalità che soddisfano le specifiche dei requisiti. Con MySQL come componente del database, è un [Relational database management system](https://it.wikipedia.org/wiki/Relational_database_management_system) (RDBMS) composto da un [client](https://it.wikipedia.org/wiki/Client) a [riga di comando](https://it.wikipedia.org/wiki/Riga_di_comando) e un [server](https://it.wikipedia.org/wiki/Server). Invece per la logica di controllo e business utilizziamo le servlet basato sul linguaggio Java e JSP, Css e JavaScript come front-end. L'usabilità del sistema è pienamente raggiunta.

• **Portability vs. Readability**

Il nostro sistema sacrifica la leggibilità del codice, per far fronte ad un requisito non funzionale la portabilità del sistema attraverso vari dispositivi, che siano schermi dei pc, tablet, o smartphone. Usando opportuni linguaggio di formattazione delle pagine Web “Css”, tramite le Media Queries permette di strutturare il contenuto di una pagina Html e JSP adattandola alla dimensione dello schermo. Inoltre usando Java Script siamo riusciti a realizzare un sito multilingua.

• **Rapid Development vs. Robustness**

Il Sistema sacrifica lo sviluppo di software robusto, per garantire un sistema funzionante nel più breve tempo richiesto, per poi garantire in futuro aggiornamenti per rendere il software robusto e per implementare altre funzionalità, o rendere più dettagliate quelle preesistenti.

• **Delivery Time vs Functionality**

Si preferisce dare priorità al delivery time rispetto al numero di funzionalità offerte dal sistema. Il sistema EmmeShop deve mettere a disposizione dell’utente le funzionalità definite. Tuttavia, il sistema deve essere consegnato entro la data prefissata. Questo porta inevitabilmente a un trade off.

Per tanto si preferisce il delivery time a discapito delle funzionalità, utilizzando un approccio in cui vengano realizzate prima le funzionalità ad alta priorità, lasciando le funzionalità con priorità media o bassa a future evoluzioni del sistema nel caso in cui il tempo a disposizione non fosse sufficiente a fornire tutte le imlementazioni.

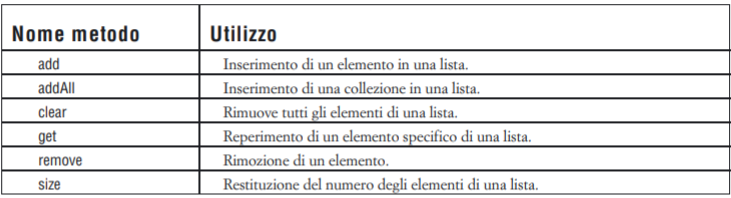
## **Interface documentation guidelines**

• Le classi sono nominate con nomi singolari, relativamente brevi e significativi. Non usare verbi, aggiungere un suffisso relativo allo strato di appartenenza, usare suffisso Exception per le classi di eccezioni.

Per lo strato di integrazione con il database usare nome classe più DAO. Per lo strato control e view la scelta ricade su nomi significativi per lo scopo che sono state ralizzate.

• I metodi sono denominati con frasi verbali, campi e parametri con frasi di nomi, brevi ed intuitivi. Non ripetere nomi della classe nel nome del metodo.

•usare una convenzione chiara nella selezione dei nomi dei metodi (add, addAll, get, set, size, clear, remove)



• selezionare nomi significativi per attributi e variabili e parametri, brevi.

• considerare l’utilizzo delle lettere i, j, k per le variabili di ciclo

• Lo stato dell'errore viene restituito tramite un'eccezione, non un valore di ritorno.

• utilizzare il livello di accesso più ristretto possibile



• massimizzare la coesione tra le classi e minimizzare accoppiamento.

• eseguire controllo del valore dei parametri passati ad un metodo appena possibile.

• L’indentazione deve essere effettuata con un TAB e qualunque sia il linguaggio usato per la produzione di codice, ogni istruzione deve essere opportunamente indentata.

Es.

for(int i=0; i<x; i++){

if(x>y){

x = 5;

return;

}

}

Deve essere sostituita da:

for(int i=0; i<x;i++){

if(x>y){

x = 5;

return;

}

}

• Mettere le dichiarazioni all’inizio dei blocchi. Non aspettare di dichiarare le variabile al loro primo uso: può confondere il programmatore inesperto e impedire la portabilità del codice dentro lo scope.

Le variabili di istanza verranno dichiarate nelle ultime righe di codice del file. L’unica eccezione a questa regola sono gli indici dei cicli for che in Java possono essere dichiarati nell’ istruzione stessa.

• Inizializzare le variabili locali nel punto in cui sono state dichiarate a meno che il suo valore iniziale non dipenda da un calcolo che occorre eseguire prima.

# **Definizioni, acronimi e abbreviazioni**

**DBMS**: Data Base Management System.

**Off-The-Shelf:** Servizi esterni di cui viene fatto utilizzo da terzi.

**HTML**: Linguaggio di mark-up per pagine web.

**CSS**: Linguaggio usato per definire la formattazione di pagine web.

**JavaScript**: Linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivitramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso

## **References**

* Problem Statement
* Requirements Analysis Document
* System Design Document
* [http://java.sun.com](http://java.sun.com/)

**Organizzazione dei file**

Ogni file deve appartenere ad uno dei seguenti package, in base al suo ruolo nella soluzione del problema:

-**Model:** contiene tutti qui file che si occupano della gestione della persistenza dei dati;

-**Eccezione:** contiene i file per nuovi tipi di eccezione creati per l’applicazione;

**-managerAccaouting:** contiene le classi per la gestione dei vari utenti del sistema, con le relative funzionalità di registrazione e login;

-**managerNegozio:** contiene le classi per la gestione del negozio, categorie e prodotti, con funzionalità di creazione, modifica e cancellazione;

-**managerOrdine:** contiene le classi per la gestione degli ordine;

-**Testing:** contiene testing blackbox e whiteblox delle varie componenti;

-**SystemTesting:** contiene i casi di test relativi al testing di sistema.

- Organizzare in una cartella “images” le varie risorse necessarie per la memorizzazione delle immagini del logo del negozio, delle categorie e dei prodotti. Queste saranno salvate secondo una specifica struttura di directory. Con convenzione di creare una cartella con nome negozio ed inserire tutte le relative immagini, con la specifica di usare nome prodotto uguale al proprio id, invece per la categoria il nome della categoria, per facilitare il salvataggio e il caricamento delle immagini, senza appesantire il server.

-Tutte le view che si interfacciano con l’utente sono divise tra diverse cartelle per permettere agli sviluppatori di facilitare la individuazione. La scelta è ricaduta su nomi specifici collegati a ciò che si andava a realizzare: seller (per le jsp di creazione del venditore e del negozio), venditore (per le varie jsp di visualizzazione e modifica dei vari elementi).

-Sono stati creati 4 Css uno generale, e uno per ogni colore che il venditore può scegliere per il proprio e-commerce.

-Tutta la documentazione è organizzata in una cartella “doc”.

## **Scelta del Design Pattern**



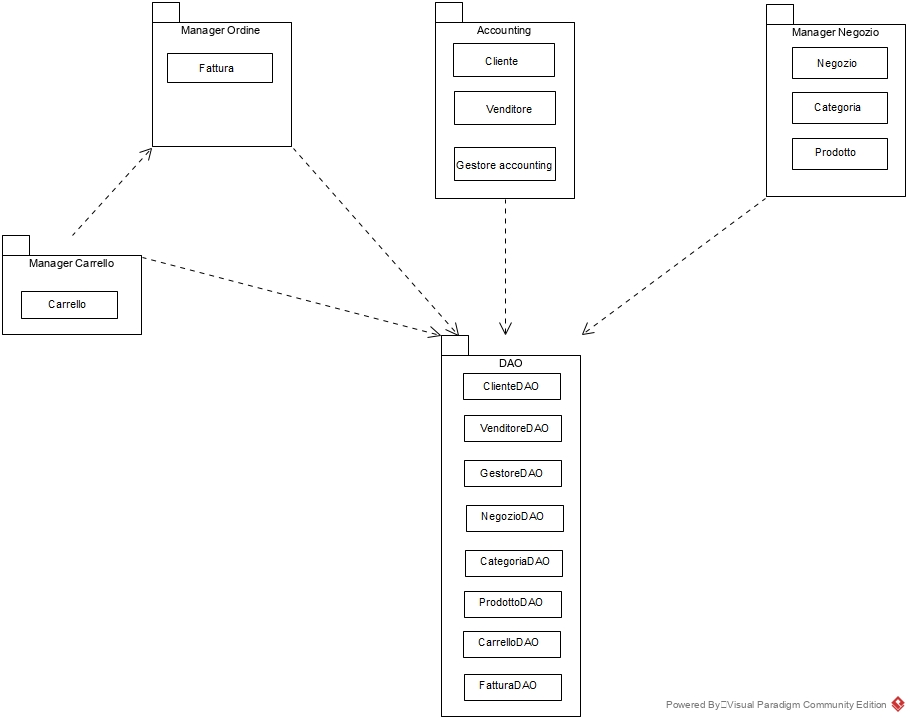
In successive versioni del sistema si potrebbe utilizzare il Bridge Pattern per la gestione della connessione del database. In questo modo si renderebbe indipendente l’implementazione per l’interfaccia al database a chi la utilizza.

Questo design pattern non è stato implementato, perché tra il trade off manutenibilità & tempo di consegna.

All’inizio del documento di System Design si era preso in considerazione di usare il design pattern Facade, che dava priorità alla manuntenibilità a discapito dell’efficienza. Facade permetteva di utilizzare ogni sottosistema in maniera semplice e indipendente dalla loro implementazione. Però durante la implementazione del sistema ci siamo accorti che portava all’aggiunta di altre classi, con un peggioramento della velocità del sistema.

Ciò ci ha portato ad utilizzare dipendenze dirette tra le varie classi permettendoci oltre ad una migliore facilità di implementazione, anche un ottima scelta per non avere un alto accoppiamento tramite la stessa interfaccia tra le componenti del View (jsp e servlet) e del livello intermedio.

# **Packages**



# JavaDocEmmeShop

|  |  |
| --- | --- |
| **Packages** | |
| **Package** | **Description** |
| [**control**](file:///D:\cetra\Users\control\package-summary.html) |  |
| [**eccezione**](file:///D:\cetra\Users\eccezione\package-summary.html) |  |
| [**manageraccouting**](file:///D:\cetra\Users\manageraccouting\package-summary.html) |  |
| [**managernegozio**](file:///D:\cetra\Users\managernegozio\package-summary.html) |  |
| [**managerordine**](file:///D:\cetra\Users\managerordine\package-summary.html) |  |
| [**model**](file:///D:\cetra\Users\model\package-summary.html) |  |
| [**systemtesting**](file:///D:\cetra\Users\systemtesting\package-summary.html) |  |
| [**testing**](file:///D:\cetra\Users\testing\package-summary.html) |  |

# Package control

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**Controlli**](file:///D:\cetra\Users\control\Controlli.html) |  |
| [**InserisciCategoria**](file:///D:\cetra\Users\control\InserisciCategoria.html) | Servlet usata per reindirezzare utente all' aggiunta della categoria |
| [**InserisciProdotto**](file:///D:\cetra\Users\control\InserisciProdotto.html) | Servlet che reindirizza utente alla creazione di un nuovo prodotto |
| [**InsertCategoria**](file:///D:\cetra\Users\control\InsertCategoria.html) | Servlet per inserire una nuova categoria |
| [**InsertProduct**](file:///D:\cetra\Users\control\InsertProduct.html) | Servlet che inserisce un nuovo prodotto |
| [**LoginCliente**](file:///D:\cetra\Users\control\LoginCliente.html) | Servlet effettua il login del cliente |
| [**LoginGestoreAccouting**](file:///D:\cetra\Users\control\LoginGestoreAccouting.html) | Servlet effettua il login del gestore accouting |
| [**LoginVenditore**](file:///D:\cetra\Users\control\LoginVenditore.html) | Servlet effettua il login al venditore |
| [**Logout**](file:///D:\cetra\Users\control\Logout.html) | Servlet che fa il logout dell'utente |
| [**ModificaCategoria**](file:///D:\cetra\Users\control\ModificaCategoria.html) | Servlet che effetua la modifica della categoria |
| [**ModificaProdotto**](file:///D:\cetra\Users\control\ModificaProdotto.html) | Servlet effettua la modifica del prodotto |
| [**RegisterNegozio**](file:///D:\cetra\Users\control\RegisterNegozio.html) | Servlet Registrazione del negozio |
| [**RegisterSeller**](file:///D:\cetra\Users\control\RegisterSeller.html) | Servlet registrazione del venditore |
| [**RemoveCat**](file:///D:\cetra\Users\control\RemoveCat.html) | Servlet cancella categoria |
| [**RemoveProd**](file:///D:\cetra\Users\control\RemoveProd.html) | Servlet rimuove prodotto |
| [**UpdateCategoria**](file:///D:\cetra\Users\control\UpdateCategoria.html) | Servlet update Categoria |
| [**UpdateProdotto**](file:///D:\cetra\Users\control\UpdateProdotto.html) | Servlet modifica prodotto |
| [**Upload**](file:///D:\cetra\Users\control\Upload.html) | Servlet fa upload del logo del negozio |
| [**UploadCategoria**](file:///D:\cetra\Users\control\UploadCategoria.html) | Servlet effettua upload del logo della categoria |
| [**UploadProdotto**](file:///D:\cetra\Users\control\UploadProdotto.html) | Servlet effettua upload del logo del prodotto |
| [**ViewCategoria**](file:///D:\cetra\Users\control\ViewCategoria.html) | Servlet ricarica la jsp delle categorie di un negozio |
| [**ViewProdotto**](file:///D:\cetra\Users\control\ViewProdotto.html) | Servlet ricarica la jsp dei prodotti di una relativa categoria di un negozio |
| [**VisualizzaOrdineVenditore**](file:///D:\cetra\Users\control\VisualizzaOrdineVenditore.html) | Servlet che reindirizza il venditore alla visione degli ordini |
| [**VisualizzaProdotti**](file:///D:\cetra\Users\control\VisualizzaProdotti.html) | Servlet che fa carica i prodotti di una determinata categoria di un negozio |

# Package eccezione

|  |  |
| --- | --- |
| **Exception Summary** | |
| **Exception** | **Description** |
| [**CategoriaNonEsisteException**](file:///D:\cetra\Users\eccezione\CategoriaNonEsisteException.html) | Eccezzione categoria non esiste |
| [**NegozioNonEsistenteException**](file:///D:\cetra\Users\eccezione\NegozioNonEsistenteException.html) | Eccezione Negozio non esiste |
| [**ParametroNonCorrettoException**](file:///D:\cetra\Users\eccezione\ParametroNonCorrettoException.html) | Eccezione parametro non corretto |
| [**UtenteNonTrovatoException**](file:///D:\cetra\Users\eccezione\UtenteNonTrovatoException.html) | Eccezione Utente non trovato |

# Package manageraccouting

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**Utente**](file:///D:\cetra\Users\manageraccouting\Utente.html) | Classe Astratta Utente |
| [**Venditore**](file:///D:\cetra\Users\manageraccouting\Venditore.html) | La classe rappresenta un Venditore, è una specializzazione di Utente Permette di trovare se un venditore è presente nel database, |

# Package managernegozio

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**Categoria**](file:///D:\cetra\Users\managernegozio\Categoria.html) | la classe Categoria gestisce le operazioni di modifica,cancellazione e rimozione di una categoria costruzine del path per inserimento delle imagini |
| [**Negozio**](file:///D:\cetra\Users\managernegozio\Negozio.html) | La classe negozio permette di aggiungere un negozio, creare la cartella del negozio sul server restituire oggetto negozio di un determinato venditore |
| [**Prodotto**](file:///D:\cetra\Users\managernegozio\Prodotto.html) | la classe Prodotto gestisce le operazioni di modifica, cancellazione di un prodotto |
|  |  |

# Package managerordine

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**ArrayFattura**](file:///D:\cetra\Users\managerordine\ArrayFattura.html) | Permette di gestire una lista di fatture con operazioni aggiunta |
| [**ArrayRiferimento**](file:///D:\cetra\Users\managerordine\ArrayRiferimento.html) | Ha le funzionalità per gestire un array di oggetti riferimento |
| [**Fattura**](file:///D:\cetra\Users\managerordine\Fattura.html) |  |
| [**Riferimento**](file:///D:\cetra\Users\managerordine\Riferimento.html) | Contiene i dati permenenti di una fattura, ha i metodi getter e setter per modificare l'oggetto riferimento inoltre permette di gestire arrayriferimento. |

# Package model

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**CategoriaDAO**](file:///D:\cetra\Users\model\CategoriaDAO.html) | Posside le operazioni di gestione la table Categoria all'interno del database |
| [**DriverManagerConnectionPool**](file:///D:\cetra\Users\model\DriverManagerConnectionPool.html) | Connessione con il db tramite ConnectionPool |
| [**NegozioDAO**](file:///D:\cetra\Users\model\NegozioDAO.html) | Realizza le operazioni da effettuare con li databese riferite alla table Negozio |
| [**ProdottoDAO**](file:///D:\cetra\Users\model\ProdottoDAO.html) | Permette di gestire la table Prodotto nel database con le operazioni piů comuni |
| [**RiferimentoDAO**](file:///D:\cetra\Users\model\RiferimentoDAO.html) | Permette di operare sulla table riferimento per operazioni CRUD |
| [**VenditoreDAO**](file:///D:\cetra\Users\model\VenditoreDAO.html) | Permette di effettuare le operazioni CRUD per la table Venditore |

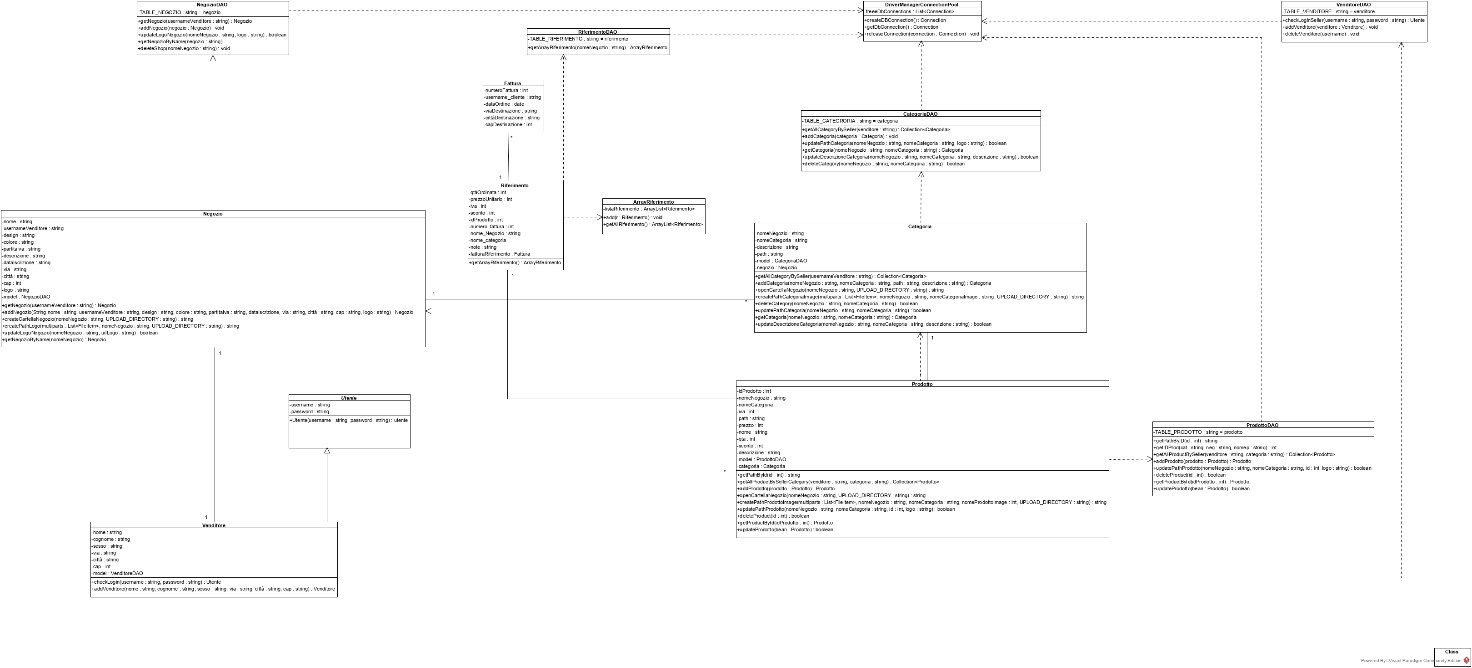
# Package systemtesting

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**LoginVenditoreTest**](file:///D:\cetra\Users\systemtesting\LoginVenditoreTest.html) | Test di sistema Login Venditore |
| [**RegistrazioneVenditoreTest**](file:///D:\cetra\Users\systemtesting\RegistrazioneVenditoreTest.html) | Test di sistema Registrazione Venditore |

# Package Testing

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| **CategoriaDAOTest**  **CategoriaTest**  **NegozioDAOTest**  **NegozioTest**  **ProdottoDAOTest**  **ProdottoTest**  **Tc1\_0\_Login\_Venditore**  **Tc2\_0\_Registrazione\_Venditore**  **Tc3\_0\_Registrazione\_Negozio**  **Tc4\_0\_Inserisci\_Categoria**  **Tc5\_0\_Inserisci\_Prodotto**  **Tc6\_0\_Modifica\_Categoria**  **Tc7\_0 Cancellazione Categoria**  **Tc8\_0\_Modifica\_Prodotto**  **Tc9\_0\_Cancellazione\_Prodotto**  **VenditoreDAOTest**  **VenditoreTest** | Test di unità CategoriaDAO  Test di unità Categoria  Test di unità NegozioDAO  Test di unità Negozio  Test di unità ProdottoDAO  Test di unità Prodotto  Test di sistema  Test di sistema  Test di sistema  Test di sistema  Test di sistema  Test di sistema  Test di sistema  Test di sistema  Test di sistema  Test di unità VenditoreDAO  Test di unità Venditore |

# **Class interfaces**



# JavaDocEmmeShop

|  |  |
| --- | --- |
| **Packages** | |
| **Package** | **Description** |
| [**control**](file:///D:\cetra\Users\doc\control\package-summary.html) |  |
| [**eccezione**](file:///D:\cetra\Users\doc\eccezione\package-summary.html) |  |
| [**manageraccouting**](file:///D:\cetra\Users\doc\manageraccouting\package-summary.html) |  |
| [**managernegozio**](file:///D:\cetra\Users\doc\managernegozio\package-summary.html) |  |
| [**managerordine**](file:///D:\cetra\Users\doc\managerordine\package-summary.html) |  |
| [**model**](file:///D:\cetra\Users\doc\model\package-summary.html) |  |
| [**systemtesting**](file:///D:\cetra\Users\doc\systemtesting\package-summary.html) |  |
| [**testing**](file:///D:\cetra\Users\doc\testing\package-summary.html) |  |

# Package control

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**Controlli**](file:///D:\cetra\Users\doc\control\Controlli.html) |  |
| [**InserisciCategoria**](file:///D:\cetra\Users\doc\control\InserisciCategoria.html) | Servlet usata per reindirezzare utente all' aggiunta della categoria |
| [**InserisciProdotto**](file:///D:\cetra\Users\doc\control\InserisciProdotto.html) | Servlet che reindirizza utente alla creazione di un nuovo prodotto |
| [**InsertCategoria**](file:///D:\cetra\Users\doc\control\InsertCategoria.html) | Servlet per inserire una nuova categoria |
| [**InsertProduct**](file:///D:\cetra\Users\doc\control\InsertProduct.html) | Servlet che inserisce un nuovo prodotto |
| [**LoginCliente**](file:///D:\cetra\Users\doc\control\LoginCliente.html) | Servlet effettua il login del cliente |
| [**LoginGestoreAccouting**](file:///D:\cetra\Users\doc\control\LoginGestoreAccouting.html) | Servlet effettua il login del gestore accouting |
| [**LoginVenditore**](file:///D:\cetra\Users\doc\control\LoginVenditore.html) | Servlet effettua il login al venditore |
| [**Logout**](file:///D:\cetra\Users\doc\control\Logout.html) | Servlet che fa il logout dell'utente |
| [**ModificaCategoria**](file:///D:\cetra\Users\doc\control\ModificaCategoria.html) | Servlet che effetua la modifica della categoria |
| [**ModificaProdotto**](file:///D:\cetra\Users\doc\control\ModificaProdotto.html) | Servlet effettua la modifica del prodotto |
| [**RegisterNegozio**](file:///D:\cetra\Users\doc\control\RegisterNegozio.html) | Servlet Registrazione del negozio |
| [**RegisterSeller**](file:///D:\cetra\Users\doc\control\RegisterSeller.html) | Servlet registrazione del venditore |
| [**RemoveCat**](file:///D:\cetra\Users\doc\control\RemoveCat.html) | Servlet cancella categoria |
| [**RemoveProd**](file:///D:\cetra\Users\doc\control\RemoveProd.html) | Servlet rimuove prodotto |
| [**UpdateCategoria**](file:///D:\cetra\Users\doc\control\UpdateCategoria.html) | Servlet update Categoria |
| [**UpdateProdotto**](file:///D:\cetra\Users\doc\control\UpdateProdotto.html) | Servlet modifica prodotto |
| [**Upload**](file:///D:\cetra\Users\doc\control\Upload.html) | Servlet fa upload del logo del negozio |
| [**UploadCategoria**](file:///D:\cetra\Users\doc\control\UploadCategoria.html) | Servlet effettua upload del logo della categoria |
| [**UploadProdotto**](file:///D:\cetra\Users\doc\control\UploadProdotto.html) | Servlet effettua upload del logo del prodotto |
| [**ViewCategoria**](file:///D:\cetra\Users\doc\control\ViewCategoria.html) | Servlet ricarica la jsp delle categorie di un negozio |
| [**ViewProdotto**](file:///D:\cetra\Users\doc\control\ViewProdotto.html) | Servlet ricarica la jsp dei prodotti di una relativa categoria di un negozio |
| [**VisualizzaOrdineVenditore**](file:///D:\cetra\Users\doc\control\VisualizzaOrdineVenditore.html) | Servlet che reindirizza il venditore alla visione degli ordini |
| [**VisualizzaProdotti**](file:///D:\cetra\Users\doc\control\VisualizzaProdotti.html) | Servlet che fa carica i prodotti di una determinata categoria di un negozio |
|  |  |

# Package eccezione

|  |  |
| --- | --- |
| **Exception Summary** | |
| **Exception** | **Description** |
| [**CategoriaNonEsisteException**](file:///D:\cetra\Users\doc\eccezione\CategoriaNonEsisteException.html) | Eccezzione categoria non esiste |
| [**NegozioNonEsistenteException**](file:///D:\cetra\Users\doc\eccezione\NegozioNonEsistenteException.html) | Eccezione Negozio non esiste |
| [**ParametroNonCorrettoException**](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html) | Eccezione parametro non corretto |
| [**UtenteNonTrovatoException**](file:///D:\cetra\Users\doc\eccezione\UtenteNonTrovatoException.html) | Eccezione Utente non trovato |

# Package manageraccouting

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**Utente**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html) | Classe Astratta Utente |
| [**Venditore**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html) | La classe rappresenta un Venditore, è una specializzazione di Utente Permette di trovare se un venditore è presente nel database, |

## Class Utente

* java.lang.Object
  + manageraccouting.Utente
* **All Implemented Interfaces:**

java.io.Serializable

**Direct Known Subclasses:**

[Venditore](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html)

public abstract class **Utente**

extends java.lang.Object

implements java.io.Serializable

Classe Astratta Utente

**Author:**

cetra

**See Also:**

Serializable, [Serialized Form](file:///D:\cetra\Users\doc\serialized-form.html#manageraccouting.Utente)

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Utente**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#Utente--)()  Costruttore vuoto |
| [**Utente**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#Utente-java.lang.String-java.lang.String-)(java.lang.String username, java.lang.String password)  Costruttore con i parametri |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| java.lang.String | [**getPassword**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#getPassword--)() |
| java.lang.String | [**getUsername**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#getUsername--)() |
| void | [**setPassword**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#setPassword-java.lang.String-)(java.lang.String password) |
| void | [**setUsername**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#setUsername-java.lang.String-)(java.lang.String username) |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### Utente

public Utente()

Costruttore vuoto

#### Utente

* + - public Utente(java.lang.String username,

java.lang.String password)

Costruttore con i parametri

**Parameters:**

username -

password -

### *Method Detail*

#### getUsername

public java.lang.String getUsername()

**Returns:**

the username

#### setUsername

public void setUsername(java.lang.String username)

**Parameters:**

username - the username to set

#### getPassword

public java.lang.String getPassword()

**Returns:**

the password

#### setPassword

public void setPassword(java.lang.String password)

**Parameters:**

password - the password to set

## Class Venditore

* java.lang.Object
  + [manageraccouting.Utente](file:///D:\cetra\Users\doc\manageraccouting\Utente.html)
    - manageraccouting.Venditore
* **All Implemented Interfaces:**

java.io.Serializable

public class **Venditore**

extends [Utente](file:///D:\cetra\Users\doc\manageraccouting\Utente.html)

La classe rappresenta un Venditore, č una specializzazione di Utente Permette di trovare se un venditore č presente nel database,

**Author:**

cetra

**See Also:**

[Utente](file:///D:\cetra\Users\doc\manageraccouting\Utente.html), [Serialized Form](file:///D:\cetra\Users\doc\serialized-form.html#manageraccouting.Venditore)

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Venditore**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#Venditore--)()  Costruttore vuoto |
| [**Venditore**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#Venditore-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String username, java.lang.String password, java.lang.String nome, java.lang.String cognome, java.lang.String email, java.lang.String sesso, java.lang.String telefono, java.lang.String via, java.lang.String citta, java.lang.String cap)  Costruttore che crea un istanza di venditore |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| [**Venditore**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html) | [**addVenditore**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#addVenditore-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String username, java.lang.String password, java.lang.String nome, java.lang.String cognome, java.lang.String email, java.lang.String sesso, java.lang.String telefono, java.lang.String via, java.lang.String citta, java.lang.String cap)  Crea un nuovo venditore, i parametri sono controllati da javascript e espressioni regolari direttamente all'inserimento. |
| [**Utente**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html) | [**checkLogin**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#checkLogin-java.lang.String-java.lang.String-)(java.lang.String username, java.lang.String password)  Controlla che nel database sia presente un istanza di venditore con username e password identiche come quelle passate come parametro |
| java.lang.String | [**getCap**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#getCap--)() |
| java.lang.String | [**getCitta**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#getCitta--)() |
| java.lang.String | [**getCognome**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#getCognome--)() |
| java.lang.String | [**getEmail**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#getEmail--)() |
| java.lang.String | [**getNome**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#getNome--)() |
| java.lang.String | [**getSesso**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#getSesso--)() |
| java.lang.String | [**getTelefono**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#getTelefono--)() |
| java.lang.String | [**getVia**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#getVia--)() |
| void | [**setCap**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#setCap-java.lang.String-)(java.lang.String cap) |
| void | [**setCitta**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#setCitta-java.lang.String-)(java.lang.String citta) |
| void | [**setCognome**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#setCognome-java.lang.String-)(java.lang.String cognome) |
| void | [**setEmail**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#setEmail-java.lang.String-)(java.lang.String email) |
| void | [**setNome**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#setNome-java.lang.String-)(java.lang.String nome) |
| void | [**setSesso**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#setSesso-java.lang.String-)(java.lang.String sesso) |
| void | [**setTelefono**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#setTelefono-java.lang.String-)(java.lang.String telefono) |
| void | [**setVia**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html#setVia-java.lang.String-)(java.lang.String via) |

### Methods inherited from class manageraccouting.[Utente](file:///D:\cetra\Users\doc\manageraccouting\Utente.html)

[getPassword](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#getPassword--), [getUsername](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#getUsername--), [setPassword](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#setPassword-java.lang.String-), [setUsername](file:///D:\cetra\Users\doc\manageraccouting\Utente.html#setUsername-java.lang.String-)

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### Venditore

public Venditore()

Costruttore vuoto

#### Venditore

* + - public Venditore(java.lang.String username,
    - java.lang.String password,
    - java.lang.String nome,
    - java.lang.String cognome,
    - java.lang.String email,
    - java.lang.String sesso,
    - java.lang.String telefono,
    - java.lang.String via,
    - java.lang.String citta,

java.lang.String cap)

Costruttore che crea un istanza di venditore

**Parameters:**

username -

password -

nome -

cognome -

email -

sesso -

telefono -

via -

citta -

cap -

### *Method Detail*

#### checkLogin

* + - public [Utente](file:///D:\cetra\Users\doc\manageraccouting\Utente.html) checkLogin(java.lang.String username,
    - java.lang.String password)
    - throws java.sql.SQLException,
    - [UtenteNonTrovatoException](file:///D:\cetra\Users\doc\eccezione\UtenteNonTrovatoException.html),

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

Controlla che nel database sia presente un istanza di venditore con username e password identiche come quelle passate come parametro

**Parameters:**

username, - username del venditore da cercare

password, - password del venditore da cercare

**Returns:**

ritorna l'istanza del utente cercato

**Throws:**

java.sql.SQLException

[UtenteNonTrovatoException](file:///D:\cetra\Users\doc\eccezione\UtenteNonTrovatoException.html)

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

#### addVenditore

* + - public [Venditore](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html) addVenditore(java.lang.String username,
    - java.lang.String password,
    - java.lang.String nome,
    - java.lang.String cognome,
    - java.lang.String email,
    - java.lang.String sesso,
    - java.lang.String telefono,
    - java.lang.String via,
    - java.lang.String citta,
    - java.lang.String cap)

throws java.sql.SQLException

Crea un nuovo venditore, i parametri sono controllati da javascript e espressioni regolari direttamente all'inserimento.

## Precondizione

username formato: lettere e numeri, min 3, max 16

password formato: almeno lettera minuscola e maiuscola, un numero, carattere speciale   
 min 6, max 25

nome formato: solo lettere minimo 1 max 25

cognome formato: solo lettere minimo 1 max 25

email formato: una sola @, un punto con max 3 lettere alla fine.

sesso formato: uno dei tre specificati M,F,Other

telefono formato: 10 cifre.Senza spazi

via formato: lettere e cifre

citta formato: lettere e cifre

cap formato: 5 cifre

**Parameters:**

username, - del venditore da inserire

password, - del venditore da inserire

nome - , del venditore da inserire

cognome, - del venditore da inserire

email, - del venditore da inserire

sesso, - del venditore da inserire

telefono, - del venditore da inserire

via, - del venditore da inserire

citta, - del venditore da inserire

cap, - del venditore da inserire

**Returns:**

ritorna oggetto Venditore appena aggiunto.

**Throws:**

java.sql.SQLException

#### getNome

public java.lang.String getNome()

**Returns:**

the nome

#### setNome

public void setNome(java.lang.String nome)

**Parameters:**

nome - the nome to set

#### getCognome

public java.lang.String getCognome()

**Returns:**

the cognome

#### setCognome

public void setCognome(java.lang.String cognome)

**Parameters:**

cognome - the cognome to set

#### getEmail

public java.lang.String getEmail()

**Returns:**

the email

#### setEmail

public void setEmail(java.lang.String email)

**Parameters:**

email - the email to set

#### getSesso

public java.lang.String getSesso()

**Returns:**

the sesso

#### setSesso

public void setSesso(java.lang.String sesso)

**Parameters:**

sesso - the sesso to set

#### getTelefono

public java.lang.String getTelefono()

**Returns:**

the telefono

#### setTelefono

public void setTelefono(java.lang.String telefono)

**Parameters:**

telefono - the telefono to set

#### getVia

public java.lang.String getVia()

**Returns:**

the via

#### setVia

public void setVia(java.lang.String via)

**Parameters:**

via - the via to set

#### getCitta

public java.lang.String getCitta()

**Returns:**

the citta

#### setCitta

public void setCitta(java.lang.String citta)

**Parameters:**

citta - the citta to set

#### getCap

public java.lang.String getCap()

**Returns:**

the cap

#### setCap

public void setCap(java.lang.String cap)

**Parameters:**

cap - the cap to set

# Package managernegozio

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) | la classe Categoria gestisce le operazioni di modifica,cancellazione e rimozione di una categoria costruzine del path per inserimento delle imagini |
| [**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) | La classe negozio permette di aggiungere un negozio, creare la cartella del negozio sul server restituire oggetto negozio di un determinato venditore |
| [**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) | la classe Prodotto gestisce le operazioni di modifica, cancellazione di un prodotto |

## Class Negozio

* java.lang.Object
  + managernegozio.Negozio
* **All Implemented Interfaces:**

java.io.Serializable

public class **Negozio**

extends java.lang.Object

implements java.io.Serializable

La classe negozio permette di aggiungere un negozio, creare la cartella del negozio sul server restituire oggetto negozio di un determinato venditore

**Author:**

cetra

**See Also:**

[Serialized Form](file:///D:\cetra\Users\doc\serialized-form.html#managernegozio.Negozio)

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#Negozio--)()  costruttore vuoto |
| [**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#Negozio-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String usernameVenditore, java.lang.String template, java.lang.String colore, java.lang.String partitaIva, java.lang.String dataIscrizione, java.lang.String descrizione, java.lang.String via, java.lang.String citta, java.lang.String cap, java.lang.String Logo)  Costruttore che crea un istanza di negozio |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| [**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) | [**addNegozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#addNegozio-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String usernameVenditore, java.lang.String template, java.lang.String colore, java.lang.String partitaIva, java.lang.String dataIscrizione, java.lang.String descrizione, java.lang.String via, java.lang.String citta, java.lang.String cap, java.lang.String Logo)  Aggiunge un nuovo negozio nel Database I parametri di input vengono controllati sia lato cliente nelle jsp che lato server nelle servlet. |
| java.lang.String | [**createCartellaNegozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#createCartellaNegozio-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String UPLOAD\_DIRECTORY)  Crea la cartella negozio se non esiste altrimenti la apre e ritorna il path |
| java.lang.String | [**createPathLogo**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#createPathLogo-java.util.List-java.lang.String-java.lang.String-)(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts, java.lang.String nomeNegozio, java.lang.String UPLOAD\_DIRECTORY)  Crea il la stringa del path per il logo del negozio |
| java.lang.String | [**getCap**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getCap--)() |
| java.lang.String | [**getCitta**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getCitta--)() |
| java.lang.String | [**getColore**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getColore--)() |
| java.lang.String | [**getDataIscrizione**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getDataIscrizione--)() |
| java.lang.String | [**getDescrizione**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getDescrizione--)() |
| java.lang.String | [**getDesign**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getDesign--)() |
| java.lang.String | [**getLogo**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getLogo--)() |
| [**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) | [**getNegozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getNegozio-java.lang.String-)(java.lang.String usernameVenditore)  Restituisce il negozio associato ad un venditore Il parametro usernameVenditore è controllato nella servlet |
| [**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) | [**getNegozioByName**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getNegozioByName-java.lang.String-)(java.lang.String nomeNegozio)  Restituire il negozio passato come parametro il nome del negozio |
| java.lang.String | [**getNomeNegozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getNomeNegozio--)() |
| java.lang.String | [**getPartitaIva**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getPartitaIva--)() |
| java.lang.String | [**getUsernameVenditore**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getUsernameVenditore--)() |
| java.lang.String | [**getVia**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#getVia--)() |
| void | [**setCap**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setCap-java.lang.String-)(java.lang.String cap) |
| void | [**setCitta**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setCitta-java.lang.String-)(java.lang.String citta) |
| void | [**setColore**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setColore-java.lang.String-)(java.lang.String colore) |
| void | [**setDataIscrizione**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setDataIscrizione-java.lang.String-)(java.lang.String dataIscrizione) |
| void | [**setDescrizione**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setDescrizione-java.lang.String-)(java.lang.String descrizione) |
| void | [**setDesign**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setDesign-java.lang.String-)(java.lang.String design) |
| void | [**setLogo**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setLogo-java.lang.String-)(java.lang.String logo) |
| void | [**setNomeNegozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setNomeNegozio-java.lang.String-)(java.lang.String nomeNegozio) |
| void | [**setPartitaIva**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setPartitaIva-java.lang.String-)(java.lang.String partitaIva) |
| void | [**setUsernameVenditore**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setUsernameVenditore-java.lang.String-)(java.lang.String usernameVenditore) |
| void | [**setVia**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#setVia-java.lang.String-)(java.lang.String via) |
| boolean | [**updateLogoNegozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html#updateLogoNegozio-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String urlLogo)  Modifica url del path del logo negozio |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### Negozio

public Negozio()

costruttore vuoto

#### Negozio

* + - public Negozio(java.lang.String nomeNegozio,
    - java.lang.String usernameVenditore,
    - java.lang.String template,
    - java.lang.String colore,
    - java.lang.String partitaIva,
    - java.lang.String dataIscrizione,
    - java.lang.String descrizione,
    - java.lang.String via,
    - java.lang.String citta,
    - java.lang.String cap,

java.lang.String Logo)

Costruttore che crea un istanza di negozio

**Parameters:**

nomeNegozio -

usernameVenditore -

template -

colore -

partitaIva -

dataIscrizione -

descrizione -

via -

citta -

cap -

Logo -

### *Method Detail*

#### getNegozio

* + - public [Negozio](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) getNegozio(java.lang.String usernameVenditore)
    - throws java.sql.SQLException,

[NegozioNonEsistenteException](file:///D:\cetra\Users\doc\eccezione\NegozioNonEsistenteException.html)

Restituisce il negozio associato ad un venditore

Il parametro usernameVenditore è controllato nella servlet

**Parameters:**

usernameVenditore, - venditore su cui fare la ricerca

**Returns:**

Negozio, oggetto negozio del determinato venditore, altrimenti lancia un eccezione

**Throws:**

java.sql.SQLException

[NegozioNonEsistenteException](file:///D:\cetra\Users\doc\eccezione\NegozioNonEsistenteException.html)

#### addNegozio

* + - public [Negozio](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) addNegozio(java.lang.String nomeNegozio,
    - java.lang.String usernameVenditore,
    - java.lang.String template,
    - java.lang.String colore,
    - java.lang.String partitaIva,
    - java.lang.String dataIscrizione,
    - java.lang.String descrizione,
    - java.lang.String via,
    - java.lang.String citta,
    - java.lang.String cap,
    - java.lang.String Logo)

throws java.sql.SQLException

Aggiunge un nuovo negozio nel Database

I parametri di input vengono controllati sia lato cliente nelle jsp che lato server nelle servlet.

per riferimento abbiamo un tabella dei formati nel RAD

**Parameters:**

nomeNegozio -

usernameVenditore -

template -

colore -

partitaIva -

dataIscrizione -

descrizione -

via -

citta -

cap -

Logo -

**Returns:**

Negozio, ritorna oggetto negozio appenna aggiunto, altrimenti si genera un eccezione in caso di errore.

**Throws:**

java.sql.SQLException

#### createCartellaNegozio

* + - public java.lang.String createCartellaNegozio(java.lang.String nomeNegozio,

java.lang.String UPLOAD\_DIRECTORY)

Crea la cartella negozio se non esiste altrimenti la apre e ritorna il path

**Parameters:**

nomeNegozio -

UPLOAD\_DIRECTORY -

**Returns:**

String UPLOAD\_DIRECTORY path della cartella

#### createPathLogo

* + - public java.lang.String createPathLogo(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts,
    - java.lang.String nomeNegozio,
    - java.lang.String UPLOAD\_DIRECTORY)

throws java.lang.Exception

Crea il la stringa del path per il logo del negozio

**Parameters:**

multiparts -

nomeNegozio -

UPLOAD\_DIRECTORY -

**Returns:**

stringa del path

**Throws:**

java.lang.Exception

#### updateLogoNegozio

* + - public boolean updateLogoNegozio(java.lang.String nomeNegozio,
    - java.lang.String urlLogo)

throws java.sql.SQLException

Modifica url del path del logo negozio

**Parameters:**

nomeNegozio -

urlLogo -

**Returns:**

boolean, true se andato a buon fine la modifica altrimenti false

**Throws:**

java.sql.SQLException

#### getNegozioByName

* + - public [Negozio](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) getNegozioByName(java.lang.String nomeNegozio)
    - throws java.sql.SQLException,

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

Restituire il negozio passato come parametro il nome del negozio

**Parameters:**

nomeNegozio - il nome del negozio

**Returns:**

negozio

**Throws:**

java.sql.SQLException

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

#### getNomeNegozio

public java.lang.String getNomeNegozio()

**Returns:**

the nomeNegozio

#### setNomeNegozio

public void setNomeNegozio(java.lang.String nomeNegozio)

**Parameters:**

nomeNegozio - the nomeNegozio to set

#### getUsernameVenditore

public java.lang.String getUsernameVenditore()

**Returns:**

the usernameVenditore

#### setUsernameVenditore

public void setUsernameVenditore(java.lang.String usernameVenditore)

**Parameters:**

usernameVenditore - the usernameVenditore to set

#### getDesign

public java.lang.String getDesign()

**Returns:**

the design

#### setDesign

public void setDesign(java.lang.String design)

**Parameters:**

design - the design to set

#### getColore

public java.lang.String getColore()

**Returns:**

the colore

#### setColore

public void setColore(java.lang.String colore)

**Parameters:**

colore - the colore to set

#### getPartitaIva

public java.lang.String getPartitaIva()

**Returns:**

the partitaIva

#### setPartitaIva

public void setPartitaIva(java.lang.String partitaIva)

**Parameters:**

partitaIva - the partitaIva to set

#### getDataIscrizione

public java.lang.String getDataIscrizione()

**Returns:**

the dataIscrizione

#### setDataIscrizione

public void setDataIscrizione(java.lang.String dataIscrizione)

**Parameters:**

dataIscrizione - the dataIscrizione to set

#### getDescrizione

public java.lang.String getDescrizione()

**Returns:**

the descrizione

#### setDescrizione

public void setDescrizione(java.lang.String descrizione)

**Parameters:**

descrizione - the descrizione to set

#### getVia

public java.lang.String getVia()

**Returns:**

the via

#### setVia

public void setVia(java.lang.String via)

**Parameters:**

via - the via to set

#### getCitta

public java.lang.String getCitta()

**Returns:**

the citta

#### setCitta

public void setCitta(java.lang.String citta)

**Parameters:**

citta - the citta to set

#### getCap

public java.lang.String getCap()

**Returns:**

the cap

#### setCap

public void setCap(java.lang.String cap)

**Parameters:**

cap - the cap to set

#### getLogo

public java.lang.String getLogo()

**Returns:**

the logo

#### setLogo

public void setLogo(java.lang.String logo)

**Parameters:**

logo - the logo to set

## Class Categoria

* java.lang.Object
  + managernegozio.Categoria

public class **Categoria**

extends java.lang.Object

la classe Categoria gestisce le operazioni di modifica,cancellazione e rimozione di una categoria costruzine del path per inserimento delle imagini

**Author:**

cetra

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#Categoria--)()  Costruttore vuoto |
| [**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#Categoria-java.lang.String-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String path, java.lang.String descrizione)  Costruttore crea istanza di Categoria |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| [**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) | [**addCategoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#addCategoria-java.lang.String-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String path, java.lang.String descrizione)  Aggiunge una nuova categoria per un determinato negozio Precondizione nomeNegozio formato: lettere e numeri min 3, max 16 nomeCategoria formato: lettere e numeri min 3, max 16 path percorso dove si trova image sul server inserita automaticamente descrizione formato: lettere cifre min 3 max 500 |
| java.lang.String | [**createPathCategoriaImage**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#createPathCategoriaImage-java.util.List-java.lang.String-java.lang.String-java.lang.String-)(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts, java.lang.String nomeNegozio, java.lang.String nomeCategoriaImage, java.lang.String UPLOAD\_DIRECTORY)  Crea il path per inserimento dell'image della categoria |
| boolean | [**deleteCategory**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#deleteCategory-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria)  Cancella la categoria di un determinato negozio |
| java.util.Collection<[**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html)> | [**getAllCategoryBySeller**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#getAllCategoryBySeller-java.lang.String-)(java.lang.String usernameVenditore)  Restituisce tutte le categorie di un determinato venditore, o lancia eccezione in caso venditore non esiste |
| [**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) | [**getCategoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#getCategoria-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria)  Restituisce oggetto categoria di un determinato negozio specificando il nome. |
| java.lang.String | [**getDescrizione**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#getDescrizione--)() |
| java.lang.String | [**getNomeCategoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#getNomeCategoria--)() |
| java.lang.String | [**getNomeNegozio**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#getNomeNegozio--)() |
| java.lang.String | [**getPath**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#getPath--)() |
| java.lang.String | [**openCartellaNegozio**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#openCartellaNegozio-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String UPLOAD\_DIRECTORY)  Crea la cartella nel server per un determinato negozio |
| void | [**setDescrizione**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#setDescrizione-java.lang.String-)(java.lang.String descrizione) |
| void | [**setNomeCategoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#setNomeCategoria-java.lang.String-)(java.lang.String nomeCategoria) |
| void | [**setNomeNegozio**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#setNomeNegozio-java.lang.String-)(java.lang.String nomeNegozio) |
| void | [**setPath**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#setPath-java.lang.String-)(java.lang.String path) |
| boolean | [**updateDescrizioneCategoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#updateDescrizioneCategoria-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String descrizione)  Modifica la descrizione di una categoria i parametri rispettano i formati e sono controllati sia da javascript che sul server |
| boolean | [**updatePathCategoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html#updatePathCategoria-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String urlLogo)  Modifica il path dell'imagine della categoria i parametri di input sono controllati dalla servlet che chiama il metodo |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### Categoria

public Categoria()

Costruttore vuoto

#### Categoria

* + - public Categoria(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - java.lang.String path,

java.lang.String descrizione)

Costruttore crea istanza di Categoria

**Parameters:**

nomeNegozio -

nomeCategoria -

path -

descrizione -

### *Method Detail*

#### getAllCategoryBySeller

* + - public java.util.Collection<[Categoria](file:///D:\cetra\Users\doc\managernegozio\Categoria.html)> getAllCategoryBySeller(java.lang.String usernameVenditore)
    - throws java.sql.SQLException,

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

Restituisce tutte le categorie di un determinato venditore, o lancia eccezione in caso venditore non esiste

**Parameters:**

usernameVenditore, - del venditore per restituire tutte le categorie associate

**Returns:**

ritorna la collezione delle categorie di un determinato venditore

**Throws:**

java.sql.SQLException

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

#### addCategoria

* + - public [Categoria](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) addCategoria(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - java.lang.String path,
    - java.lang.String descrizione)

throws java.sql.SQLException

Aggiunge una nuova categoria per un determinato negozio

## Precondizione

nomeNegozio formato: lettere e numeri min 3, max 16

nomeCategoria formato: lettere e numeri min 3, max 16

path percorso dove si trova image sul server inserita automaticamente

descrizione formato: lettere cifre min 3 max 500

**Parameters:**

nomeNegozio -

nomeCategoria -

path -

descrizione -

**Returns:**

categoria aggiunta

**Throws:**

java.sql.SQLException

#### openCartellaNegozio

* + - public java.lang.String openCartellaNegozio(java.lang.String nomeNegozio,

java.lang.String UPLOAD\_DIRECTORY)

Crea la cartella nel server per un determinato negozio

**Parameters:**

nomeNegozio, - nome che viene dato alal cartella

UPLOAD\_DIRECTORY, - stringa path base del server

**Returns:**

String del path della cartella appena creato

#### createPathCategoriaImage

* + - public java.lang.String createPathCategoriaImage(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts,
    - java.lang.String nomeNegozio,
    - java.lang.String nomeCategoriaImage,
    - java.lang.String UPLOAD\_DIRECTORY)

throws java.lang.Exception

Crea il path per inserimento dell'image della categoria

**Parameters:**

multiparts, - lista

nomeNegozio, - serve per la composizione del path

nomeCategoriaImage, - serve per la composizione del path

UPLOAD\_DIRECTORY, - stringa base del path per la costruzione

**Returns:**

String del path dove per image della categoria

**Throws:**

java.lang.Exception

#### deleteCategory

* + - public boolean deleteCategory(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria)
    - throws [ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html),

java.sql.SQLException

Cancella la categoria di un determinato negozio

**Parameters:**

nomeNegozio, - nome del negozio di cui cancellare la categoria

nomeCategoria, - nome della categoria da cancellare

**Returns:**

boolean true se la cancellazione avviene con successo altrimenti lancia un eccezione

**Throws:**

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

java.sql.SQLException

#### updatePathCategoria

* + - public boolean updatePathCategoria(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - java.lang.String urlLogo)

throws java.sql.SQLException

Modifica il path dell'imagine della categoria

i parametri di input sono controllati dalla servlet che chiama il metodo

**Parameters:**

nomeNegozio -

nomeCategoria -

urlLogo -

**Returns:**

boolean True se andato a buon fine

**Throws:**

java.sql.SQLException

#### getCategoria

* + - public [Categoria](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) getCategoria(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria)

throws java.sql.SQLException

Restituisce oggetto categoria di un determinato negozio specificando il nome.

I paramtri di input sono controllati dalla servlet, prelevati dalla sessione

**Parameters:**

nomeNegozio -

nomeCategoria -

**Returns:**

Categoria, dato il nome negozio e il nome categoria

**Throws:**

java.sql.SQLException

#### updateDescrizioneCategoria

* + - public boolean updateDescrizioneCategoria(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - java.lang.String descrizione)

throws java.sql.SQLException

Modifica la descrizione di una categoria

i parametri rispettano i formati e sono controllati sia da javascript che sul server

**Parameters:**

nomeNegozio -

nomeCategoria -

descrizione -

**Returns:**

boolean True in caso di successo della modifica altrimenti false

**Throws:**

java.sql.SQLException

#### getNomeNegozio

public java.lang.String getNomeNegozio()

**Returns:**

the nomeNegozio

#### setNomeNegozio

public void setNomeNegozio(java.lang.String nomeNegozio)

**Parameters:**

nomeNegozio - the nomeNegozio to set

#### getNomeCategoria

public java.lang.String getNomeCategoria()

**Returns:**

the nomeCategoria

#### setNomeCategoria

public void setNomeCategoria(java.lang.String nomeCategoria)

**Parameters:**

nomeCategoria - the nomeCategoria to set

#### getDescrizione

public java.lang.String getDescrizione()

**Returns:**

the descrizione

#### setDescrizione

public void setDescrizione(java.lang.String descrizione)

**Parameters:**

descrizione - the descrizione to set

#### getPath

public java.lang.String getPath()

**Returns:**

the path

#### setPath

public void setPath(java.lang.String path)

**Parameters:**

path - the path to set

## Class Prodotto

* java.lang.Object
  + managernegozio.Prodotto

public class **Prodotto**

extends java.lang.Object

la classe Prodotto gestisce le operazioni di modifica, cancellazione di un prodotto

**Author:**

cetra

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#Prodotto--)()  Costruttore vuoto |
| [**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#Prodotto-int-java.lang.String-java.lang.String-java.lang.String-int-java.lang.String-float-int-int-java.lang.String-)(int idProdotto, java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String nome, int iva, java.lang.String path, float prezzo, int qta, int sconto, java.lang.String descrizione)  Costruttore crea un oggetto Prodotto |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| [**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) | [**addProdotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#addProdotto-managernegozio.Prodotto-)([**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) prodotto)  Aggiunge un prodotto nel database il prodotto passato in input non deve essere null |
| java.lang.String | [**createPathProdottoImage**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#createPathProdottoImage-java.util.List-java.lang.String-java.lang.String-int-java.lang.String-)(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts, java.lang.String nomeNegozio, java.lang.String nomeCategoria, int nomeProdottoImage, java.lang.String UPLOAD\_DIRECTORY)  Crea il path del image prodotto |
| boolean | [**deleteProduct**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#deleteProduct-int-)(int id)  Cancella il prodotto tramite il suo id dal database |
| java.util.Collection<[**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html)> | [**getAllProductBySellerCategory**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getAllProductBySellerCategory-java.lang.String-java.lang.String-)(java.lang.String venditore, java.lang.String categoria)  restituisce tutti i prodotti relativa a quella categoria di quel venditore. |
| java.lang.String | [**getDescrizione**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getDescrizione--)() |
| int | [**getIdProdotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getIdProdotto--)() |
| int | [**getIva**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getIva--)() |
| java.lang.String | [**getNome**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getNome--)() |
| java.lang.String | [**getNomeCategoria**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getNomeCategoria--)() |
| java.lang.String | [**getNomeNegozio**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getNomeNegozio--)() |
| java.lang.String | [**getPath**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getPath--)() |
| java.lang.String | [**getPathByID**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getPathByID-int-)(int id)  Crea il path per inserire image del prodotto |
| float | [**getPrezzo**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getPrezzo--)() |
| [**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) | [**getProductById**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getProductById-int-)(int idProdotto)  restituisce il prodotto tramite il suo id |
| int | [**getQuantita**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getQuantita--)() |
| int | [**getSconto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#getSconto--)() |
| java.lang.String | [**openCartellaNegozio**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#openCartellaNegozio-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String UPLOAD\_DIRECTORY)  crea la cartella se non esiste se no preleva il path per restituirlo |
| void | [**setDescrizione**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setDescrizione-java.lang.String-)(java.lang.String descrizione)  questo metodo setta la descrizione di un prodotto |
| void | [**setIdProdotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setIdProdotto-int-)(int idProdotto)  questo metodo setta l'id di un prodotto |
| void | [**setIva**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setIva-int-)(int iva)  questo metodo setta l'iva di un prodotto |
| void | [**setNome**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setNome-java.lang.String-)(java.lang.String nome) |
| void | [**setNomeCategoria**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setNomeCategoria-java.lang.String-)(java.lang.String nomeCategoria) |
| void | [**setNomeNegozio**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setNomeNegozio-java.lang.String-)(java.lang.String nomeNegozio) |
| void | [**setPath**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setPath-java.lang.String-)(java.lang.String path)  questo metodo setta il path dove è contenuta l'immagine del prodotto |
| void | [**setPrezzo**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setPrezzo-float-)(float prezzo)  questo metodo setta il prezzo di un prodotto |
| void | [**setQuantita**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setQuantita-int-)(int qta)  questo metodo setta la quantità di un prodotto |
| void | [**setSconto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#setSconto-int-)(int sconto)  questo metodo setta lo sconto di un prodotto |
| boolean | [**updatePathProdotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#updatePathProdotto-java.lang.String-java.lang.String-int-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria, int id, java.lang.String logo)  Modifica il path dell'image del prodotto |
| boolean | [**updateProdotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html#updateProdotto-managernegozio.Prodotto-)([**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) bean)  Modifica il prodotto |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### Prodotto

public Prodotto()

Costruttore vuoto

#### Prodotto

* + - public Prodotto(int idProdotto,
    - java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - java.lang.String nome,
    - int iva,
    - java.lang.String path,
    - float prezzo,
    - int qta,
    - int sconto,

java.lang.String descrizione)

Costruttore crea un oggetto Prodotto

**Parameters:**

idProdotto -

nomeNegozio -

nomeCategoria -

nome -

iva -

path -

prezzo -

qta -

sconto -

descrizione -

### *Method Detail*

#### getPathByID

* + - public java.lang.String getPathByID(int id)

throws java.sql.SQLException

Crea il path per inserire image del prodotto

**Parameters:**

id -

**Returns:**

string, path dell'image del prodotto

**Throws:**

java.sql.SQLException

#### getAllProductBySellerCategory

* + - public java.util.Collection<[Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html)> getAllProductBySellerCategory(java.lang.String venditore,
    - java.lang.String categoria)
    - throws java.sql.SQLException,

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

restituisce tutti i prodotti relativa a quella categoria di quel venditore.

## Precondizione

venditore, categoria devono non essere null

**Parameters:**

venditore -

categoria -

**Returns:**

Collection di prodotti relativi a quella categoria di quel venditore

**Throws:**

java.sql.SQLException

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

#### addProdotto

* + - public [Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) addProdotto([Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) prodotto)
    - throws java.sql.SQLException,

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

Aggiunge un prodotto nel database

il prodotto passato in input non deve essere null

**Parameters:**

prodotto -

**Returns:**

Prodotto, restituisce il prodotto appena aggiunto o un eccezione

**Throws:**

java.sql.SQLException

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

#### openCartellaNegozio

* + - public java.lang.String openCartellaNegozio(java.lang.String nomeNegozio,

java.lang.String UPLOAD\_DIRECTORY)

crea la cartella se non esiste se no preleva il path per restituirlo

**Parameters:**

nomeNegozio -

UPLOAD\_DIRECTORY -

**Returns:**

String, url path dove inserisce image del prodotto

#### createPathProdottoImage

* + - public java.lang.String createPathProdottoImage(java.util.List<org.apache.tomcat.util.http.fileupload.FileItem> multiparts,
    - java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - int nomeProdottoImage,
    - java.lang.String UPLOAD\_DIRECTORY)

throws java.lang.Exception

Crea il path del image prodotto

**Parameters:**

multiparts -

nomeNegozio -

nomeCategoria -

nomeProdottoImage - id del prodotto

UPLOAD\_DIRECTORY -

**Returns:**

String path del logo

**Throws:**

java.lang.Exception

#### updatePathProdotto

* + - public boolean updatePathProdotto(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - int id,
    - java.lang.String logo)

throws java.sql.SQLException

Modifica il path dell'image del prodotto

**Parameters:**

nomeNegozio -

nomeCategoria -

id -

logo -

**Returns:**

boolean true in caso di successo

**Throws:**

java.sql.SQLException

#### deleteProduct

* + - public boolean deleteProduct(int id)

throws java.sql.SQLException

Cancella il prodotto tramite il suo id dal database

**Parameters:**

id -

**Returns:**

boolean true se la cancellazione ha avuto successo

**Throws:**

java.sql.SQLException

#### getProductById

* + - public [Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) getProductById(int idProdotto)

throws java.sql.SQLException

restituisce il prodotto tramite il suo id

**Parameters:**

idProdotto -

**Returns:**

Prodotto

**Throws:**

java.sql.SQLException

#### updateProdotto

* + - public boolean updateProdotto([Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) bean)

throws java.sql.SQLException

Modifica il prodotto

**Parameters:**

bean -

**Returns:**

boolean true se avvenuta la modifica

**Throws:**

java.sql.SQLException

#### getIdProdotto

public int getIdProdotto()

**Returns:**

int

#### setIdProdotto

public void setIdProdotto(int idProdotto)

questo metodo setta l'id di un prodotto

**Parameters:**

id - è l'id da assegnare

#### getIva

public int getIva()

**Returns:**

int

#### setIva

public void setIva(int iva)

questo metodo setta l'iva di un prodotto

**Parameters:**

iva - è l'iva da assegnare

#### getPath

public java.lang.String getPath()

**Returns:**

string

#### setPath

public void setPath(java.lang.String path)

questo metodo setta il path dove è contenuta l'immagine del prodotto

**Parameters:**

path - è il path da assegnare

#### getPrezzo

public float getPrezzo()

**Returns:**

float

#### setPrezzo

public void setPrezzo(float prezzo)

questo metodo setta il prezzo di un prodotto

**Parameters:**

prezzo - è il prezzo da assegnare

#### getQuantita

public int getQuantita()

**Returns:**

int

#### setQuantita

public void setQuantita(int qta)

questo metodo setta la quantità di un prodotto

**Parameters:**

qtà - è la quantità da assegnare

#### getSconto

public int getSconto()

**Returns:**

int

#### setSconto

public void setSconto(int sconto)

questo metodo setta lo sconto di un prodotto

**Parameters:**

sconto - è la sconto da assegnare

#### getDescrizione

public java.lang.String getDescrizione()

**Returns:**

string

#### setDescrizione

public void setDescrizione(java.lang.String descrizione)

questo metodo setta la descrizione di un prodotto

**Parameters:**

descrizione - è la descrizione da assegnare

#### getNomeNegozio

public java.lang.String getNomeNegozio()

**Returns:**

the nomeNegozio

#### setNomeNegozio

public void setNomeNegozio(java.lang.String nomeNegozio)

**Parameters:**

nomeNegozio - the nomeNegozio to set

#### getNomeCategoria

public java.lang.String getNomeCategoria()

**Returns:**

the nomeCategoria

#### setNomeCategoria

public void setNomeCategoria(java.lang.String nomeCategoria)

**Parameters:**

nomeCategoria - the nomeCategoria to set

#### getNome

public java.lang.String getNome()

**Returns:**

the nome

#### setNome

public void setNome(java.lang.String nome)

**Parameters:**

nome - the nome to set

## Class ArrayFattura

* java.lang.Object
  + managerordine.ArrayFattura

public class **ArrayFattura**

extends java.lang.Object

Permette di gestire una lista di fatture con operazioni aggiunta

**Author:**

cetra

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**ArrayFattura**](file:///D:\cetra\Users\doc\managerordine\ArrayFattura.html#ArrayFattura--)()  costruttore che crea una arrylist di fatture vuoto |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| void | [**add**](file:///D:\cetra\Users\doc\managerordine\ArrayFattura.html#add-managerordine.Fattura-)([**Fattura**](file:///D:\cetra\Users\doc\managerordine\Fattura.html) f)  Aggiunge una fattura alla lista |
| java.util.ArrayList<[**Fattura**](file:///D:\cetra\Users\doc\managerordine\Fattura.html)> | [**getAllFatture**](file:///D:\cetra\Users\doc\managerordine\ArrayFattura.html#getAllFatture--)()  restituisce la lista delle fatture |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### ArrayFattura

public ArrayFattura()

costruttore che crea una arrylist di fatture vuoto

### *Method Detail*

#### add

public void add([Fattura](file:///D:\cetra\Users\doc\managerordine\Fattura.html) f)

Aggiunge una fattura alla lista

**Parameters:**

f - Fattura

#### getAllFatture

public java.util.ArrayList<[Fattura](file:///D:\cetra\Users\doc\managerordine\Fattura.html)> getAllFatture()

restituisce la lista delle fatture

**Returns: Restituisce tutte le fatture**

## Class ArrayRiferimento

* java.lang.Object
  + managerordine.ArrayRiferimento

public class **ArrayRiferimento**

extends java.lang.Object

Ha le funzionalità per gestire un array di oggetti riferimento

**Author:**

cetra

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**ArrayRiferimento**](file:///D:\cetra\Users\doc\managerordine\ArrayRiferimento.html#ArrayRiferimento--)()  Costruttore crea un lista riferimento vuota |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| void | [**add**](file:///D:\cetra\Users\doc\managerordine\ArrayRiferimento.html#add-managerordine.Riferimento-)([**Riferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html) f)  Aggiunge un riferimento alla lista |
| java.util.ArrayList<[**Riferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html)> | [**getAllRiferimento**](file:///D:\cetra\Users\doc\managerordine\ArrayRiferimento.html#getAllRiferimento--)()  restituisce la lista riferimento |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### ArrayRiferimento

public ArrayRiferimento()

Costruttore crea un lista riferimento vuota

### *Method Detail*

#### add

public void add([Riferimento](file:///D:\cetra\Users\doc\managerordine\Riferimento.html) f)

Aggiunge un riferimento alla lista

**Parameters:**

f -

#### getAllRiferimento

public java.util.ArrayList<[Riferimento](file:///D:\cetra\Users\doc\managerordine\Riferimento.html)> getAllRiferimento()

restituisce la lista riferimento

**Returns:**

ArrayList

## Class Fattura

* java.lang.Object
  + managerordine.Fattura

public class **Fattura**

extends java.lang.Object

**Author:**

manlio la classe Fattura gestisce le operazioni di un ordine

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Fattura**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#Fattura--)()  Costruttore vuoto |
| [**Fattura**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#Fattura-int-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-java.lang.String-)(int numFattura, java.lang.String username\_Cliente, java.lang.String nomeNegozio, java.lang.String dataOrdine, java.lang.String descrizione, java.lang.String viaPartenza, java.lang.String cittaPartenza, java.lang.String capPartenza, java.lang.String viaDestinazione, java.lang.String cittaDestinazione, java.lang.String capDestinazione)  Costruttore che crea una fattura |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| java.lang.String | [**getCapDestinazione**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#getCapDestinazione--)() |
| java.lang.String | [**getCittaDestinazione**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#getCittaDestinazione--)() |
| java.lang.String | [**getDataOrdine**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#getDataOrdine--)() |
| [**Riferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html) | [**getListaRiferimento**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#getListaRiferimento--)() |
| java.lang.String | [**getNomeNegozio**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#getNomeNegozio--)() |
| int | [**getNumFattura**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#getNumFattura--)() |
| java.lang.String | [**getUsername\_Cliente**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#getUsername_Cliente--)() |
| java.lang.String | [**getViaDestinazione**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#getViaDestinazione--)() |
| void | [**setCapDestinazione**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#setCapDestinazione-java.lang.String-)(java.lang.String capDestinazione) |
| void | [**setCittaDestinazione**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#setCittaDestinazione-java.lang.String-)(java.lang.String cittaDestinazione) |
| void | [**setDataOrdine**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#setDataOrdine-java.lang.String-)(java.lang.String dataOrdine) |
| void | [**setListaRiferimento**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#setListaRiferimento-managerordine.Riferimento-)([**Riferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html) listaRiferimento) |
| void | [**setNomeNegozio**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#setNomeNegozio-java.lang.String-)(java.lang.String nomeNegozio) |
| void | [**setNumFattura**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#setNumFattura-int-)(int numFattura) |
| void | [**setUsername\_Cliente**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#setUsername_Cliente-java.lang.String-)(java.lang.String username\_Cliente) |
| void | [**setViaDestinazione**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#setViaDestinazione-java.lang.String-)(java.lang.String viaDestinazione) |
| java.lang.String | [**toString**](file:///D:\cetra\Users\doc\managerordine\Fattura.html#toString--)() |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

### *Constructor Detail*

#### Fattura

public Fattura()

Costruttore vuoto

#### Fattura

* + - public Fattura(int numFattura,
    - java.lang.String username\_Cliente,
    - java.lang.String nomeNegozio,
    - java.lang.String dataOrdine,
    - java.lang.String descrizione,
    - java.lang.String viaPartenza,
    - java.lang.String cittaPartenza,
    - java.lang.String capPartenza,
    - java.lang.String viaDestinazione,
    - java.lang.String cittaDestinazione,

java.lang.String capDestinazione)

Costruttore che crea una fattura

**Parameters:**

numFattura -

username\_Cliente -

nomeNegozio -

dataOrdine -

descrizione -

viaPartenza -

cittaPartenza -

capPartenza -

viaDestinazione -

cittaDestinazione -

capDestinazione -

### *Method Detail*

#### getUsername\_Cliente

public java.lang.String getUsername\_Cliente()

**Returns:**

the username\_Cliente

#### setUsername\_Cliente

public void setUsername\_Cliente(java.lang.String username\_Cliente)

**Parameters:**

username\_Cliente - the username\_Cliente to set

#### getNomeNegozio

public java.lang.String getNomeNegozio()

**Returns:**

the nomeNegozio

#### setNomeNegozio

public void setNomeNegozio(java.lang.String nomeNegozio)

**Parameters:**

nomeNegozio - the nomeNegozio to set

#### getListaRiferimento

public [Riferimento](file:///D:\cetra\Users\doc\managerordine\Riferimento.html) getListaRiferimento()

**Returns:**

the listaRiferimento

#### setListaRiferimento

public void setListaRiferimento([Riferimento](file:///D:\cetra\Users\doc\managerordine\Riferimento.html) listaRiferimento)

**Parameters:**

listaRiferimento - the listaRiferimento to set

#### getDataOrdine

public java.lang.String getDataOrdine()

**Returns:**

the dataOrdine

#### setDataOrdine

public void setDataOrdine(java.lang.String dataOrdine)

**Parameters:**

dataOrdine - the dataOrdine to set

#### getViaDestinazione

public java.lang.String getViaDestinazione()

**Returns:**

the viaDestinazione

#### setViaDestinazione

public void setViaDestinazione(java.lang.String viaDestinazione)

**Parameters:**

viaDestinazione - the viaDestinazione to set

#### getCittaDestinazione

public java.lang.String getCittaDestinazione()

**Returns:**

the cittaDestinazione

#### setCittaDestinazione

public void setCittaDestinazione(java.lang.String cittaDestinazione)

**Parameters:**

cittaDestinazione - the cittaDestinazione to set

#### getCapDestinazione

public java.lang.String getCapDestinazione()

**Returns:**

the capDestinazione

#### setCapDestinazione

public void setCapDestinazione(java.lang.String capDestinazione)

**Parameters:**

capDestinazione - the capDestinazione to set

#### getNumFattura

public int getNumFattura()

**Returns:**

the numFattura

#### setNumFattura

public void setNumFattura(int numFattura)

**Parameters:**

numFattura - the numFattura to set

#### toString

public java.lang.String toString()

**Overrides:**

toString in class java.lang.Object

## Class Riferimento

* java.lang.Object
  + managerordine.Riferimento

public class **Riferimento**

extends java.lang.Object

Contiene i dati permenenti di una fattura, ha i metodi getter e setter per modificare l'oggetto riferimento inoltre permette di gestire arrayriferimento.

**Author:**

cetra

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Riferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#Riferimento--)()  costruttore vuoto |
| [**Riferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#Riferimento-int-int-java.lang.String-java.lang.String-int-int-float-int-)(int id\_prodotto, int numero\_Fattura, java.lang.String nome\_Negozio, java.lang.String nome\_Categoria, int qtaOrdinata, int sconto, float prezzoUnitario, int iva)  Costruttore che crea un oggetto Riferimento |
| [**Riferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#Riferimento-int-int-java.lang.String-java.lang.String-int-int-float-int-java.lang.String-)(int id\_prodotto, int numero\_Fattura, java.lang.String nome\_Negozio, java.lang.String nome\_Categoria, int qtaOrdinata, int sconto, float prezzoUnitario, int iva, java.lang.String note)  Costruttore che crea un oggetto riferimento con nota |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| [**ArrayRiferimento**](file:///D:\cetra\Users\doc\managerordine\ArrayRiferimento.html) | [**getArrayRiferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getArrayRiferimento-java.lang.String-)(java.lang.String nomeNegozio)  Restituisce una lista di oggetti riferimenti associati ad un negozio nomeNegozio diverso da null |
| [**Fattura**](file:///D:\cetra\Users\doc\managerordine\Fattura.html) | [**getFatturaRiferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getFatturaRiferimento--)()  restituisce la fattura referita |
| int | [**getId\_prodotto**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getId_prodotto--)() |
| int | [**getIva**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getIva--)() |
| java.lang.String | [**getNome\_Categoria**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getNome_Categoria--)() |
| java.lang.String | [**getNome\_Negozio**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getNome_Negozio--)() |
| java.lang.String | [**getNote**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getNote--)() |
| int | [**getNumero\_Fattura**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getNumero_Fattura--)() |
| float | [**getPrezzoUnitario**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getPrezzoUnitario--)() |
| int | [**getQtaOrdinata**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getQtaOrdinata--)() |
| int | [**getSconto**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#getSconto--)() |
| void | [**setFatturaRiferimento**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setFatturaRiferimento-managerordine.Fattura-)([**Fattura**](file:///D:\cetra\Users\doc\managerordine\Fattura.html) fatturaRiferimento)  setta la fattura riferimento |
| void | [**setId\_prodotto**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setId_prodotto-int-)(int id\_prodotto) |
| void | [**setIva**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setIva-int-)(int iva) |
| void | [**setNome\_Categoria**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setNome_Categoria-java.lang.String-)(java.lang.String nome\_Categoria) |
| void | [**setNome\_Negozio**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setNome_Negozio-java.lang.String-)(java.lang.String nome\_Negozio) |
| void | [**setNote**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setNote-java.lang.String-)(java.lang.String note) |
| void | [**setNumero\_Fattura**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setNumero_Fattura-int-)(int numero\_Fattura) |
| void | [**setPrezzoUnitario**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setPrezzoUnitario-float-)(float prezzoUnitario) |
| void | [**setQtaOrdinata**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setQtaOrdinata-int-)(int qtaOrdinata) |
| void | [**setSconto**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#setSconto-int-)(int sconto) |
| java.lang.String | [**toString**](file:///D:\cetra\Users\doc\managerordine\Riferimento.html#toString--)() |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

### *Constructor Detail*

#### Riferimento

public Riferimento()

costruttore vuoto

#### Riferimento

* + - public Riferimento(int id\_prodotto,
    - int numero\_Fattura,
    - java.lang.String nome\_Negozio,
    - java.lang.String nome\_Categoria,
    - int qtaOrdinata,
    - int sconto,
    - float prezzoUnitario,

int iva)

Costruttore che crea un oggetto Riferimento

**Parameters:**

id\_prodotto -

numero\_Fattura -

nome\_Negozio -

nome\_Categoria -

qtaOrdinata -

sconto -

prezzoUnitario -

iva -

#### Riferimento

* + - public Riferimento(int id\_prodotto,
    - int numero\_Fattura,
    - java.lang.String nome\_Negozio,
    - java.lang.String nome\_Categoria,
    - int qtaOrdinata,
    - int sconto,
    - float prezzoUnitario,
    - int iva,

java.lang.String note)

Costruttore che crea un oggetto riferimento con nota

**Parameters:**

id\_prodotto -

numero\_Fattura -

nome\_Negozio -

nome\_Categoria -

qtaOrdinata -

sconto -

prezzoUnitario -

iva -

note -

### *Method Detail*

#### getFatturaRiferimento

public [Fattura](file:///D:\cetra\Users\doc\managerordine\Fattura.html) getFatturaRiferimento()

restituisce la fattura referita

**Returns:**

Fattura

#### setFatturaRiferimento

public void setFatturaRiferimento([Fattura](file:///D:\cetra\Users\doc\managerordine\Fattura.html) fatturaRiferimento)

setta la fattura riferimento

**Parameters:**

fatturaRiferimento -

#### toString

public java.lang.String toString()

**Overrides:**

toString in class java.lang.Object

#### getArrayRiferimento

* + - public [ArrayRiferimento](file:///D:\cetra\Users\doc\managerordine\ArrayRiferimento.html) getArrayRiferimento(java.lang.String nomeNegozio)
    - throws [ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html),

java.sql.SQLException

Restituisce una lista di oggetti riferimenti associati ad un negozio

nomeNegozio diverso da null

**Parameters:**

nomeNegozio -

**Returns:**

**Throws:**

[ParametroNonCorrettoException](file:///D:\cetra\Users\doc\eccezione\ParametroNonCorrettoException.html)

java.sql.SQLException

#### getId\_prodotto

public int getId\_prodotto()

**Returns:**

the id\_prodotto

#### setId\_prodotto

public void setId\_prodotto(int id\_prodotto)

**Parameters:**

id\_prodotto - the id\_prodotto to set

#### getNumero\_Fattura

public int getNumero\_Fattura()

**Returns:**

the numero\_Fattura

#### setNumero\_Fattura

public void setNumero\_Fattura(int numero\_Fattura)

**Parameters:**

numero\_Fattura - the numero\_Fattura to set

#### getNome\_Negozio

public java.lang.String getNome\_Negozio()

**Returns:**

the nome\_Negozio

#### setNome\_Negozio

public void setNome\_Negozio(java.lang.String nome\_Negozio)

**Parameters:**

nome\_Negozio - the nome\_Negozio to set

#### getNome\_Categoria

public java.lang.String getNome\_Categoria()

**Returns:**

the nome\_Categoria

#### setNome\_Categoria

public void setNome\_Categoria(java.lang.String nome\_Categoria)

**Parameters:**

nome\_Categoria - the nome\_Categoria to set

#### getQtaOrdinata

public int getQtaOrdinata()

**Returns:**

the qtaOrdinata

#### setQtaOrdinata

public void setQtaOrdinata(int qtaOrdinata)

**Parameters:**

qtaOrdinata - the qtaOrdinata to set

#### getSconto

public int getSconto()

**Returns:**

the sconto

#### setSconto

public void setSconto(int sconto)

**Parameters:**

sconto - the sconto to set

#### getPrezzoUnitario

public float getPrezzoUnitario()

**Returns:**

the prezzoUnitario

#### setPrezzoUnitario

public void setPrezzoUnitario(float prezzoUnitario)

**Parameters:**

prezzoUnitario - the prezzoUnitario to set

#### getIva

public int getIva()

**Returns:**

the iva

#### setIva

public void setIva(int iva)

**Parameters:**

iva - the iva to set

#### getNote

public java.lang.String getNote()

**Returns:**

the note

#### setNote

public void setNote(java.lang.String note)

**Parameters:**

note - the note to set

# Package model

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**CategoriaDAO**](file:///D:\cetra\Users\doc\model\CategoriaDAO.html) | Posside le operazioni di gestione la table Categoria all'interno del database |
| [**DriverManagerConnectionPool**](file:///D:\cetra\Users\doc\model\DriverManagerConnectionPool.html) | Connessione con il db tramite ConnectionPool |
| [**NegozioDAO**](file:///D:\cetra\Users\doc\model\NegozioDAO.html) | Realizza le operazioni da effettuare con li databese riferite alla table Negozio |
| [**ProdottoDAO**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html) | Permette di gestire la table Prodotto nel database con le operazioni piů comuni |
| [**RiferimentoDAO**](file:///D:\cetra\Users\doc\model\RiferimentoDAO.html) | Permette di operare sulla table riferimento per operazioni CRUD |
| [**VenditoreDAO**](file:///D:\cetra\Users\doc\model\VenditoreDAO.html) | Permette di effettuare le operazioni CRUD per la table Venditore |

## Class CategoriaDAO

* java.lang.Object
  + model.CategoriaDAO
* **All Implemented Interfaces:**

java.io.Serializable

public class **CategoriaDAO**

extends java.lang.Object

implements java.io.Serializable

Posside le operazioni di gestione la table Categoria all'interno del database

**Author:**

cetra

**See Also:**

[Serialized Form](file:///D:\cetra\Users\doc\serialized-form.html#model.CategoriaDAO)

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**CategoriaDAO**](file:///D:\cetra\Users\doc\model\CategoriaDAO.html#CategoriaDAO--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| void | [**addCategoria**](file:///D:\cetra\Users\doc\model\CategoriaDAO.html#addCategoria-managernegozio.Categoria-)([**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) categoria)  Aggiunge una nuova categoria Il nome del negozio, nome venditore deve essere già presente nel database Il nome della categoria non può essere inserito in caso sia già presente con lo stesso nome, nome del venditore, negozio |
| boolean | [**deleteCategory**](file:///D:\cetra\Users\doc\model\CategoriaDAO.html#deleteCategory-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria)  Cancella la categoria di un determinato negozio il nome negozio e la categoria associata, da cancellare devono essere presenti nel db |
| java.util.Collection<[**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html)> | [**getAllCategoryBySeller**](file:///D:\cetra\Users\doc\model\CategoriaDAO.html#getAllCategoryBySeller-java.lang.String-)(java.lang.String venditore)  Restituisce una lista di categorie di un determinato venditore |
| [**Categoria**](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) | [**getCategoria**](file:///D:\cetra\Users\doc\model\CategoriaDAO.html#getCategoria-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria)  Restituisce la categoria di un determinato negozio nel database deve essere presente il negozio, e la categoria associata a quel negozio |
| boolean | [**updateDescrizioneCategoria**](file:///D:\cetra\Users\doc\model\CategoriaDAO.html#updateDescrizioneCategoria-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String descrizione)  Modifica la descrizione di una categoria nel database devono essere presente nome negozio e il nome categoria associato al negozio |
| boolean | [**updatePathCategoria**](file:///D:\cetra\Users\doc\model\CategoriaDAO.html#updatePathCategoria-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria, java.lang.String logo)  Modifica il path della categoria nel caso il path non sia già aggiunto lo inserisce il parametro nomeNegozio deve essere presente nel database il parametro nomeCategoria deve essere riferito al nomeNegozio è deve essere presente nel db |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### CategoriaDAO

public CategoriaDAO()

### *Method Detail*

#### getAllCategoryBySeller

* + - public java.util.Collection<[Categoria](file:///D:\cetra\Users\doc\managernegozio\Categoria.html)> getAllCategoryBySeller(java.lang.String venditore)

throws java.sql.SQLException

Restituisce una lista di categorie di un determinato venditore

**Parameters:**

venditore -

**Returns:**

Collection, lista di categoria

**Throws:**

java.sql.SQLException

#### addCategoria

* + - public void addCategoria([Categoria](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) categoria)

throws java.sql.SQLException

Aggiunge una nuova categoria

Il nome del negozio, nome venditore deve essere già presente nel database

Il nome della categoria non può essere inserito in caso sia già presente con lo stesso nome, nome del venditore, negozio

**Parameters:**

categoria -

**Throws:**

java.sql.SQLException

#### updatePathCategoria

* + - public boolean updatePathCategoria(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - java.lang.String logo)

throws java.sql.SQLException

Modifica il path della categoria

nel caso il path non sia già aggiunto lo inserisce

il parametro nomeNegozio deve essere presente nel database

il parametro nomeCategoria deve essere riferito al nomeNegozio è deve essere presente nel db

**Parameters:**

nomeNegozio -

nomeCategoria -

logo -

**Returns:**

boolean true in caso di successo della modifica

**Throws:**

java.sql.SQLException

#### getCategoria

* + - public [Categoria](file:///D:\cetra\Users\doc\managernegozio\Categoria.html) getCategoria(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria)

throws java.sql.SQLException

Restituisce la categoria di un determinato negozio

nel database deve essere presente il negozio, e la categoria associata a quel negozio

**Parameters:**

nomeNegozio -

nomeCategoria -

**Returns:**

restituisce la categoria

**Throws:**

java.sql.SQLException

#### updateDescrizioneCategoria

* + - public boolean updateDescrizioneCategoria(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - java.lang.String descrizione)

throws java.sql.SQLException

Modifica la descrizione di una categoria

nel database devono essere presente nome negozio e il nome categoria associato al negozio

**Parameters:**

nomeNegozio -

nomeCategoria -

descrizione -

**Returns:**

flag boolean true se la modifica avviene altrimenti false

**Throws:**

java.sql.SQLException

#### deleteCategory

* + - public boolean deleteCategory(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria)

throws java.sql.SQLException

Cancella la categoria di un determinato negozio

il nome negozio e la categoria associata, da cancellare devono essere presenti nel db

**Parameters:**

nomeNegozio -

nomeCategoria -

**Returns:**

boolean true se la cancellazione avviene con successo

**Throws:**

java.sql.SQLException

## Class DriverManagerConnectionPool

* java.lang.Object
  + model.DriverManagerConnectionPool

public class **DriverManagerConnectionPool**

extends java.lang.Object

Connessione con il db tramite ConnectionPool

**Author:**

cetra

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**DriverManagerConnectionPool**](file:///D:\cetra\Users\doc\model\DriverManagerConnectionPool.html#DriverManagerConnectionPool--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Static Methods**](javascript:show(1);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| static java.sql.Connection | [**createDBConnection**](file:///D:\cetra\Users\doc\model\DriverManagerConnectionPool.html#createDBConnection--)() |
| static java.sql.Connection | [**getDbConnection**](file:///D:\cetra\Users\doc\model\DriverManagerConnectionPool.html#getDbConnection--)() |
| static void | [**releaseConnection**](file:///D:\cetra\Users\doc\model\DriverManagerConnectionPool.html#releaseConnection-java.sql.Connection-)(java.sql.Connection connection) |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### DriverManagerConnectionPool

public DriverManagerConnectionPool()

### *Method Detail*

#### createDBConnection

* + - public static java.sql.Connection createDBConnection()

throws java.sql.SQLException

**Throws:**

java.sql.SQLException

#### getDbConnection

* + - public static java.sql.Connection getDbConnection()

throws java.sql.SQLException

**Throws:**

java.sql.SQLException

#### releaseConnection

public static void releaseConnection(java.sql.Connection connection)

## Class NegozioDAO

* java.lang.Object
  + model.NegozioDAO
* **All Implemented Interfaces:**

java.io.Serializable

public class **NegozioDAO**

extends java.lang.Object

implements java.io.Serializable

Realizza le operazioni da effettuare con li databese riferite alla table Negozio

**Author:**

cetra

**See Also:**

[Serialized Form](file:///D:\cetra\Users\doc\serialized-form.html#model.NegozioDAO)

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**NegozioDAO**](file:///D:\cetra\Users\doc\model\NegozioDAO.html#NegozioDAO--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| void | [**addNegozio**](file:///D:\cetra\Users\doc\model\NegozioDAO.html#addNegozio-managernegozio.Negozio-)([**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) negozio)  Aggiunge un negozio nel db il negozio deve rispettare il formato per essere inserito |
| void | [**deleteShop**](file:///D:\cetra\Users\doc\model\NegozioDAO.html#deleteShop-java.lang.String-)(java.lang.String nomeNegozio)  Cancella un negozio |
| [**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) | [**getNegozio**](file:///D:\cetra\Users\doc\model\NegozioDAO.html#getNegozio-java.lang.String-)(java.lang.String usernameVenditore)  Restituisce il negozio di un venditore Il venditore deve essere presente nel db e deve aver creato un negozio |
| [**Negozio**](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) | [**getNegozioByName**](file:///D:\cetra\Users\doc\model\NegozioDAO.html#getNegozioByName-java.lang.String-)(java.lang.String negozio)  Restituisce il nome del negozio Il negozio deve essere già presente con nome uguale al parametro negozio |
| boolean | [**updateLogoNegozio**](file:///D:\cetra\Users\doc\model\NegozioDAO.html#updateLogoNegozio-java.lang.String-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String logo)  Modifica il path del logo del negozio Il negozio deve essere già presente |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### NegozioDAO

public NegozioDAO()

### *Method Detail*

#### getNegozio

* + - public [Negozio](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) getNegozio(java.lang.String usernameVenditore)

throws java.sql.SQLException

Restituisce il negozio di un venditore

Il venditore deve essere presente nel db e deve aver creato un negozio

**Parameters:**

usernameVenditore -

**Returns:**

Negozio, riferito ad un venditore

**Throws:**

java.sql.SQLException

#### addNegozio

* + - public void addNegozio([Negozio](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) negozio)

throws java.sql.SQLException

Aggiunge un negozio nel db

il negozio deve rispettare il formato per essere inserito

**Parameters:**

negozio -

**Throws:**

java.sql.SQLException

#### updateLogoNegozio

* + - public boolean updateLogoNegozio(java.lang.String nomeNegozio,
    - java.lang.String logo)

throws java.sql.SQLException

Modifica il path del logo del negozio

Il negozio deve essere già presente

**Parameters:**

nomeNegozio -

logo -

**Returns:**

boolean, true se avvenuto la modifica

**Throws:**

java.sql.SQLException

#### getNegozioByName

* + - public [Negozio](file:///D:\cetra\Users\doc\managernegozio\Negozio.html) getNegozioByName(java.lang.String negozio)

throws java.sql.SQLException

Restituisce il nome del negozio

Il negozio deve essere già presente con nome uguale al parametro negozio

**Parameters:**

negozio -

**Returns:**

Negozio

**Throws:**

java.sql.SQLException

#### deleteShop

* + - public void deleteShop(java.lang.String nomeNegozio)

throws java.sql.SQLException

Cancella un negozio

**Parameters:**

nomeNegozio -

**Throws:**

java.sql.SQLException

## Class ProdottoDAO

* java.lang.Object
  + model.ProdottoDAO
* **All Implemented Interfaces:**

java.io.Serializable

public class **ProdottoDAO**

extends java.lang.Object

implements java.io.Serializable

Permette di gestire la table Prodotto nel database con le operazioni più comuni

**Author:**

cetra

**See Also:**

[Serialized Form](file:///D:\cetra\Users\doc\serialized-form.html#model.ProdottoDAO)

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**ProdottoDAO**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#ProdottoDAO--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| [**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) | [**addProdotto**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#addProdotto-managernegozio.Prodotto-)([**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) prodotto)  Aggiunge un prodotto alla categoria specificata La categoria, negozio devono essere presenti nel database. |
| boolean | [**deleteProduct**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#deleteProduct-int-)(int id)  Cancella il prodotto tramite id del prodotto id del prodotto deve essere presente nel database |
| java.util.Collection<[**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html)> | [**getAllProductBySellerCategory**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#getAllProductBySellerCategory-java.lang.String-java.lang.String-)(java.lang.String venditore, java.lang.String categoria)  Restituisce la collezione di prodotti in base al venditore e categoria i parametri venditori e categoria devono essere presenti nel database. |
| int | [**getIDProd**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#getIDProd-java.lang.String-java.lang.String-java.lang.String-)(java.lang.String cat, java.lang.String neg, java.lang.String nomep)  Restituisce id del negozio tramite la categoria negozio e prodotto Negozio deve essere presente avere la categoria indicata nel parametro cat. |
| java.lang.String | [**getPathByID**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#getPathByID-int-)(int id)  Restituisc il path (url dove sono salvate le immagini del negozio) del prodotto tramite id ID del negozio deve essere presente nel database |
| [**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) | [**getProductById**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#getProductById-int-)(int idProdotto)  Restituisce il prodotto, tramite il suo id id deve essere presente nel database |
| boolean | [**updatePathProdotto**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#updatePathProdotto-java.lang.String-java.lang.String-int-java.lang.String-)(java.lang.String nomeNegozio, java.lang.String nomeCategoria, int id, java.lang.String logo)  Ritorna true o false a seconda se il path è stato modificato Nome negozio deve essere presente nel db , la categoria deve essere associata al negozio e deve essere presente nel db table categoria, id del prodotto deve essere riferito allo stesso negozio e categoria. |
| boolean | [**updateProdotto**](file:///D:\cetra\Users\doc\model\ProdottoDAO.html#updateProdotto-managernegozio.Prodotto-)([**Prodotto**](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) bean)  Modifica il prodotto Id del prodotto deve essere presente nel database, nome negozio e nome categoria devono essere associati nel database allo stesso id prodotto |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### ProdottoDAO

public ProdottoDAO()

### *Method Detail*

#### getPathByID

* + - public java.lang.String getPathByID(int id)

throws java.sql.SQLException

Restituisc il path (url dove sono salvate le immagini del negozio) del prodotto tramite id

ID del negozio deve essere presente nel database

**Parameters:**

id -

**Returns:**

String, path del negozio

**Throws:**

java.sql.SQLException

#### getIDProd

* + - public int getIDProd(java.lang.String cat,
    - java.lang.String neg,
    - java.lang.String nomep)

throws java.sql.SQLException

Restituisce id del negozio tramite la categoria negozio e prodotto

Negozio deve essere presente avere la categoria indicata nel parametro cat.

Nella categoria deve essere presente il prodotto riferito con nome nomep.

**Parameters:**

cat -

neg -

nomep -

**Returns:**

int, id del negozio

**Throws:**

java.sql.SQLException

#### getAllProductBySellerCategory

* + - public java.util.Collection<[Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html)> getAllProductBySellerCategory(java.lang.String venditore,
    - java.lang.String categoria)

throws java.sql.SQLException

Restituisce la collezione di prodotti in base al venditore e categoria

i parametri venditori e categoria devono essere presenti nel database.

la categoria deve essere associata al negozio del venditore passato per parametro.

**Parameters:**

venditore -

categoria -

**Returns:**

Collection, lista di prodotti

**Throws:**

java.sql.SQLException

#### addProdotto

* + - public [Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) addProdotto([Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) prodotto)

throws java.sql.SQLException

Aggiunge un prodotto alla categoria specificata

La categoria, negozio devono essere presenti nel database.

Il negozio deve avere la categoria, specificata per insirire il prodotto.

**Parameters:**

prodotto -

**Returns:**

Prodotto aggiunto

**Throws:**

java.sql.SQLException

#### updatePathProdotto

* + - public boolean updatePathProdotto(java.lang.String nomeNegozio,
    - java.lang.String nomeCategoria,
    - int id,
    - java.lang.String logo)

throws java.sql.SQLException

Ritorna true o false a seconda se il path è stato modificato

Nome negozio deve essere presente nel db ,

la categoria deve essere associata al negozio e deve essere presente nel db table categoria,

id del prodotto deve essere riferito allo stesso negozio e categoria.

**Parameters:**

nomeNegozio -

nomeCategoria -

id -

logo -

**Returns:**

boolean, true se la modifica è avvenuta

**Throws:**

java.sql.SQLException

#### deleteProduct

* + - public boolean deleteProduct(int id)

throws java.sql.SQLException

Cancella il prodotto tramite id del prodotto

id del prodotto deve essere presente nel database

**Parameters:**

id -

**Returns:**

**Throws:**

java.sql.SQLException

#### getProductById

* + - public [Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) getProductById(int idProdotto)

throws java.sql.SQLException

Restituisce il prodotto, tramite il suo id

id deve essere presente nel database

**Parameters:**

idProdotto -

**Returns:**

Prodotto

**Throws:**

java.sql.SQLException

#### updateProdotto

* + - public boolean updateProdotto([Prodotto](file:///D:\cetra\Users\doc\managernegozio\Prodotto.html) bean)

throws java.sql.SQLException

Modifica il prodotto

Id del prodotto deve essere presente nel database,

nome negozio e nome categoria devono essere associati nel database allo stesso id prodotto

**Parameters:**

bean -

**Returns:**

booleam, true se il prodotto viene modificato

**Throws:**

java.sql.SQLException

## Class RiferimentoDAO

* java.lang.Object
  + model.RiferimentoDAO
* **All Implemented Interfaces:**

java.io.Serializable

public class **RiferimentoDAO**

extends java.lang.Object

implements java.io.Serializable

Permette di operare sulla table riferimento per operazioni CRUD

**Author:**

cetra

**See Also:**

[Serialized Form](file:///D:\cetra\Users\doc\serialized-form.html#model.RiferimentoDAO)

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**RiferimentoDAO**](file:///D:\cetra\Users\doc\model\RiferimentoDAO.html#RiferimentoDAO--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| [**ArrayRiferimento**](file:///D:\cetra\Users\doc\managerordine\ArrayRiferimento.html) | [**getArrayRiferimento**](file:///D:\cetra\Users\doc\model\RiferimentoDAO.html#getArrayRiferimento-java.lang.String-)(java.lang.String nomeNegozio)  Restituisce la lista di riferimento di un determinato negozio Il negozio deve essere presente nel db |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### RiferimentoDAO

public RiferimentoDAO()

### *Method Detail*

#### getArrayRiferimento

* + - public [ArrayRiferimento](file:///D:\cetra\Users\doc\managerordine\ArrayRiferimento.html) getArrayRiferimento(java.lang.String nomeNegozio)

throws java.sql.SQLException

Restituisce la lista di riferimento di un determinato negozio

Il negozio deve essere presente nel db

**Parameters:**

nomeNegozio -

**Returns:**

ArrayRiferimento lista che contiene tutte le fatture del determinato negozio passato come input

**Throws:**

java.sql.SQLException

## Class VenditoreDAO

* java.lang.Object
  + model.VenditoreDAO
* **All Implemented Interfaces:**

java.io.Serializable

public class **VenditoreDAO**

extends java.lang.Object

implements java.io.Serializable

Permette di effettuare le operazioni CRUD per la table Venditore

**Author:**

cetra

**See Also:**

[Serialized Form](file:///D:\cetra\Users\doc\serialized-form.html#model.VenditoreDAO)

### *Constructor Summary*

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**VenditoreDAO**](file:///D:\cetra\Users\doc\model\VenditoreDAO.html#VenditoreDAO--)() |

### *Method Summary*

|  |  |
| --- | --- |
| **All Methods**[**Instance Methods**](javascript:show(2);)[**Concrete Methods**](javascript:show(8);) | |
| **Modifier and Type** | **Method and Description** |
| void | [**addVenditore**](file:///D:\cetra\Users\doc\model\VenditoreDAO.html#addVenditore-manageraccouting.Venditore-)([**Venditore**](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html) venditore)  Aggiunge un venditore nel database Username e password non devono essere giù presenti nel db |
| [**Utente**](file:///D:\cetra\Users\doc\manageraccouting\Utente.html) | [**checkLoginSeller**](file:///D:\cetra\Users\doc\model\VenditoreDAO.html#checkLoginSeller-java.lang.String-java.lang.String-)(java.lang.String username, java.lang.String password)  Verifica la presenza di un venditore nel database Controlla se quello restituito dal database è null |
| void | [**deleteVenditore**](file:///D:\cetra\Users\doc\model\VenditoreDAO.html#deleteVenditore-java.lang.String-)(java.lang.String username)  Cancella il venditore dal database tramite username Username deve essere presente nel database |

### Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

### *Constructor Detail*

#### VenditoreDAO

public VenditoreDAO()

### *Method Detail*

#### checkLoginSeller

* + - public [Utente](file:///D:\cetra\Users\doc\manageraccouting\Utente.html) checkLoginSeller(java.lang.String username,
    - java.lang.String password)

throws java.sql.SQLException

Verifica la presenza di un venditore nel database

Controlla se quello restituito dal database è null

**Parameters:**

username -

password -

**Returns:**

Utente

**Throws:**

java.sql.SQLException

#### addVenditore

* + - public void addVenditore([Venditore](file:///D:\cetra\Users\doc\manageraccouting\Venditore.html) venditore)

throws java.sql.SQLException

Aggiunge un venditore nel database

Username e password non devono essere giù presenti nel db

**Parameters:**

venditore -

**Throws:**

java.sql.SQLException

#### deleteVenditore

* + - public void deleteVenditore(java.lang.String username)

throws java.sql.SQLException

Cancella il venditore dal database tramite username

Username deve essere presente nel database

**Parameters:**

username -

**Throws:**

java.sql.SQLException

# Package systemtesting

|  |  |
| --- | --- |
| **Class Summary** | |
| **Class** | **Description** |
| [**LoginVenditoreTest**](file:///D:\cetra\Users\doc\systemtesting\LoginVenditoreTest.html) |  |
| [**RegistrazioneVenditoreTest**](file:///D:\cetra\Users\doc\systemtesting\RegistrazioneVenditoreTest.html) |  |