

404NotFound

Premi: better than Prezi.



Definizione di Prodotto

Versione	1.0
Redazione	Gobbo Ismaele Rettore Andrea Vegro Federico Manuto Monica De Lazzari Enrico Cossu Mattia Camborata Marco
Verifica	Vegro Federico Camborata Marco
Responsabile	Cossu Mattia
Uso	Esterno
Stato	Formale
Ultima modifica	21 agosto 2015
Lista di distribuzione	404NotFound prof. Tullio Vardanega prof. Riccardo Cardin Zucchetti S.p.a.

Registro delle modifiche

Versione	Autore	Data	Descrizione
1.0	Cossu Mattia	21-08-2015	Approvazione documento
0.1	Gobbo Ismaele	18-06-2015	Stesura scheletro, scrittura introduzione al documento

Tabella 1: Storico versioni del documento.

Indice

1	Introduzione	4
1.1	Scopo del documento	4
1.2	Scopo del Progetto	4
1.3	Glossario	4
1.4	Riferimenti	4
1.4.1	Normativi	4
1.4.2	Informativi	4
2	Standard di Progetto	5
3	Specifica componenti	5
3.1	premi/server	5
3.1.1	premi/server/publish	5
3.2	premi/client	19
3.3	premi/client/presentation	19
3.4	premi/client/presentationManager	19
3.4.1	premi/client/presentationManager/controllers/editPresentationCtrl	19
3.4.2	premi/client/presentationManager/controllers/newPresentationCtrl	21
3.4.3	premi/client/presentationManager/controllers/presentationManagerCtrl	22
3.4.4	premi/client/presentationManager/controllers/presentationsCtrl	22
3.4.5	premi/client/presentationManager/controllers/removePresentationCtrl	23
3.4.6	premi/client/presentationManager/views/editPresentation.ng . .	24
3.4.7	premi/client/presentationManager/views/newPresentation.ng .	24
3.4.8	premi/client/presentationManager/views/presentationManager.ng	24
3.4.9	premi/client/presentationManager/views/presentations.ng . . .	25
3.4.10	premi/client/presentationManager/views/removePresentation.ng	25
3.5	premi/client/editor	25
3.5.1	premi/client/editor/lib/GObject	25
3.5.2	premi/client/editor/lib/GOProvider	28
3.5.3	premi/client/editor/lib/frame	29
3.6	premi/client/frameEditor	31
3.7	premi/client/infographicEditor	31
3.8	premi/client/trailsEditor	31
3.9	premi/client/userManager	31
3.10	premi/client/viewer	31
4	Tracciamento	31
5	Diagrammi di Sequenza	31

Elenco delle tabelle

1	Storico versioni del documento.	1
---	---	---

Elenco delle figure

1	Diagramma del package premi/client	5
2	Diagramma della classe premi/server/publish	5
3	Diagramma del package premi/client/presentationManager	19
4	Diagramma della classe premi/client/presentationManager/editPresentationCtrl	20
5	Diagramma della classe premi/client/presentationManager/newPresentationCtrl	21
6	Diagramma della classe premi/client/presentationManager/controllers/presentationManagerCtrl	22
7	Diagramma della classe premi/client/presentationManager/controllers/presentationsCtrl	22
8	Diagramma della classe premi/client/presentationManager/removePresentationCtrl	23
9	Diagramma della classe premi/client/editor	26
10	Diagramma della classe premi/client/editor/GObject	26
11	Diagramma della classe premi/client/editor/GObject	29
12	Diagramma della classe premi/client/editor/GObject	29

1 Introduzione

1.1 Scopo del documento

Questo documento approfondisce la definizione della struttura e dei componenti di Premi già discussa nella *Specifica Tecnica v2.0*. Ogni componente verrà descritto in modo sufficientemente dettagliato da consentire ai programmatori di sviluppare il software in modo coerente con quanto progettato finora.

1.2 Scopo del Progetto

Lo scopo del progetto è la realizzazione di un software di presentazione di slide non basato sul modello di PowerPoint_G, sviluppato in tecnologia HTML5_G e che funzioni sia su desktop che su dispositivo mobile. Il software dovrà permettere la creazione da parte dell'autore e la successiva presentazione del lavoro, fornendo effetti grafici di supporto allo storytelling e alla creazione di mappe mentali.

1.3 Glossario

Al fine di evitare ogni ambiguità relativa al linguaggio e ai termini utilizzati nei documenti formali tutti i termini e gli acronimi presenti nel seguente documento che necessitano di definizione saranno seguiti da una "G" in pedice e saranno riportati in un documento esterno denominato Glossario.pdf. Tale documento accompagna e completa il presente e consiste in un listato ordinato di termini e acronimi con le rispettive definizioni e spiegazioni.

1.4 Riferimenti

1.4.1 Normativi

- **Norme di Progetto:** *NormeDiProgetto_v2.0.pdf*;
- **Capitolato d'appalto C4:** Premi: Software di presentazione "better than Prezi" - <http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C4.pdf>.

1.4.2 Informativi

- **Slide dell'insegnamento Ingegneria del Software modulo A:**
<http://www.math.unipd.it/~tullio/IS-1/2014/>;
- **Ingegneria del software - Ian Sommerville - 8a Edizione (2007):**
 - Part 4: Software Management.

2 Standard di Progetto

3 Specifica componenti

3.1 premi/server

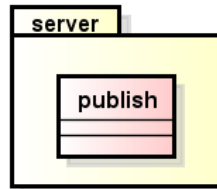


Figura 1: Diagramma del package premi/client

3.1.1 premi/server/publish

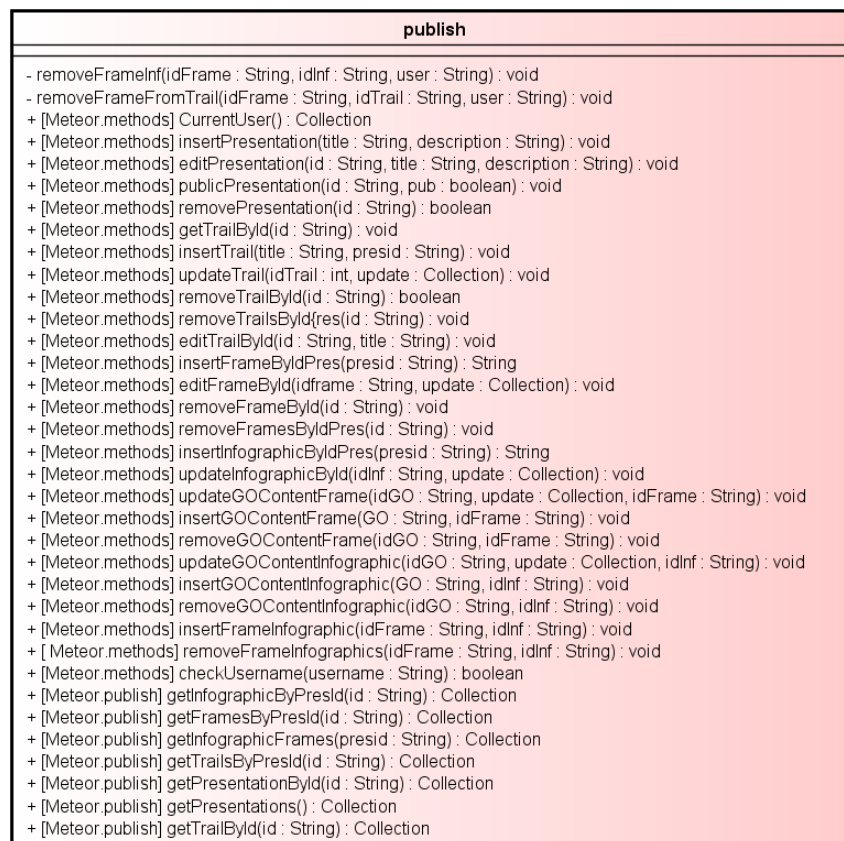


Figura 2: Diagramma della classe premi/server/publish

Descrizione

Lista di metodi che permettono al client di interagire con il database del server. I metodi marcati [**Meteor.methods**] devono essere inseriti nel servizio \$meteor per permettere poi al client di accedervi attraverso il pattern Dependency Injection_G, mentre quelli marcati [**Meteor.publish**] hanno lo scopo di pubblicare al client soltanto le informazioni a cui esso può accedere.

Metodi

- **removeFrameInf(idFrame : String, idInf : String, user : String) : void**

Metodo privato di utilità che rimuove ogni occorrenza di un frame da un'infografica.

Argomenti

- **idFrame : String**
Codice identificativo del frame da rimuovere
- **idInf : String**
Codice identificativo dell'infografica che possiede il frame
- **user : String**
Codice identificativo dell'utente che sta effettuando la rimozione

Note

- Il client non deve poter accedere direttamente a questo metodo

- **removeFrameFromTrail(idFrame : String, idTrail : String, user : String) : void**

Metodo privato di utilità che rimuove ogni occorrenza di un frame da un trail

Argomenti

- **idFrame : String**
Codice identificativo del frame da rimuovere
- **idTrail : String**
Codice identificativo del trail che possiede il frame
- **user : String**
Codice identificativo dell'utente che sta effettuando la rimozione

Note

- Il client non deve poter accedere direttamente a questo metodo

+ **currentUser() : Collection**

Restituisce le informazioni riguardanti l'utente che sta interrogando il database attraverso una collezione di documenti di MongoDB

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`

+ **insertPresentation(title : String, description : String) : String**

Aggiunge una nuova presentazione nel database di proprietà dell'utente, e restituisce il suo codice identificativo

Argomenti

- **title : String**
Titolo della nuova presentazione
- **description : String**
Descrizione della nuova presentazione

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`
- Inizializzare ogni attributo della presentazione:
 - *title*: con il titolo ricevuto
 - *description*: con la descrizione ricevuta
 - *owner*: con l'id dell'utente che sta creando la presentazione
 - *isPublic*: **false**, la presentazione inizialmente è sempre privata
- Ogni presentazione possiede almeno un'infografica, che andrà creata e inizializzata nel database con i seguenti campi dato:
 - *dataX*: -5000
 - *dataY*: -4000
 - *dataZ*: 0
 - *scale*: 0
 - *height*: 7920
 - *width*: 10240
 - *zoom*: 0
 - *presid*: il codice univoco della presentazione
 - *owner*: l'id dell'utente che sta creando la presentazione

- *background*: Collezione(JSON) vuota
- *content*: Collezione(JSON) vuota
- *framesId*: array vuoto
- *type*: infographics

+ **editPresentation(id : String, title : String, description : String) : void**

Modifica le informazioni di una presentazione (titolo e descrizione)

Argomenti

- **title : String**
Nuovo titolo della presentazione
- **description : String**
Nuova descrizione della presentazione

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ **publicPresentation(id : String, pub : boolean) : void**

Rende una presentazione pubblica o privata

Argomenti

- **id : String**
Codice identificativo della presentazione
- **pub : boolean**
Indica se la presentazione è pubblica (**true**) o privata (**false**)

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ **removePresentation(id : String) : boolean**

Rimuove una presentazione dal database. Restituisce un valore che indica se l'operazione è avvenuta con successo

Argomenti

- **id : String**
Codice identificativo della presentazione.

Note

- Devono essere rimossi, assieme alla presentazione, anche tutti gli altri dati ad essa associati (frame, infografica e trails)
- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`

+ getTrailById(id : String) : Collection

Restituisce il trail corrispondente al codice identificativo fornito, attraverso una collezione di documenti di MongoDB

Argomenti

- **id : String**
Codice identificativo del trail che si sta cercando

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`

+ insertTrail(title : String, presid : String) : void

Inserisce un nuovo trail associato ad una presentazione all'interno del database

Argomenti

- **title : String**
Titolo del nuovo trail
- **presid : String**
Codice identificativo della presentazione a cui il trail è associato

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`
- inizializzare il trail con i seguenti campi:
 - *title*: con il titolo ricevuto
 - *owner*: con l'id dell'utente che sta chiamando il metodo
 - *presid*: con il codice identificativo della presentazione

– *trail*: è una matrice vuota (`[]`)

+ **updateTrail(idTrail : int, update : Collection) : void**

Aggiorna i dati di un trail con quelli forniti

Argomenti

- **idTrail : String**
Codice identificativo del trail
- **update : Collection**
Insieme dei dati del trail modificati dall'utente

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`

+ **removeTrailById(id : String) : boolean**

Rimuove dal database il trail associato al codice identificativo fornito. Restituisce **true** se l'operazione ha avuto successo, **false** altrimenti.

Argomenti

- **id : String**
Codice identificativo del trail da rimuovere

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`

+ **removeTrailsByIdPres(id : String) : void**

Rimuove dal database ogni trail associato ad una presentazione

Argomenti

- **id : String**
Codice identificativo della presentazione da cui rimuovere ogni trail

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`

+ `editTrailById(id : String, title : String) : void`

Permette all'utente di rinominare un trail in suo possesso

Argomenti

- **id : String**
Codice identificativo del trail da rinominare
- **title : String**
Nuovo titolo, o nome, del trail

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`

+ `insertFrameByIdPres(presid : String) : String`

Inserisce un nuovo frame all'interno di una presentazione, e restituisce il suo id

Argomenti

- **presid : String**
Codice identificativo della presentazione a cui assegnare il nuovo frame

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`
- Inizializzare il frame con i seguenti campi dato:
 - *presid*: con il codice identificativo della presentazione
 - *owner*: con l'id dell'utente che ha effettuato la chiamata al metodo
 - *dataX*: 0
 - *dataY*: 0
 - *dataZ*: 0
 - *height*: 792
 - *width*: 1024
 - *scale*: 1
 - *backgroundColor*: #FFFFFF
 - *content*: Collezione(JSON) vuota
 - *type*: frame
 - *lvl*: 0

+ `editFrameById(idframe : String, update : Collection) : void`

Modifica un frame con nuovi dati inseriti dall'utente

Argomenti

- **idframe : String**
Codice identificativo del frame
- **update : Collection**
Collezione di dati modificati del frame da salvare sul database

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`

+ `removeFrameById(id : String) : void`

Rimuove dal database il frame corrispondente al codice identificativo inviato

Argomenti

- **id : String**
Codice identificativo del frame da rimuovere

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando `Meteor.methods`
- Il frame dev'essere rimosso anche dai trail e dalle infografiche in cui è stato utilizzato. Servirsi dei metodi `removeFrameFromTrail` e `RemoveFramInf`

+ `removeFramesByIdPres(id : String) : void`

Rimuove dal database tutti i frame associati ad una presentazione.

Argomenti

- **id : String**
Codice identificativo della presentazione

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ insertInfographicByIdPres(presid : String) : String

Inserisce un'infografica nel database associandola ad una presentazione, e restituisce il suo codice identificativo

Argomenti

- **presid : String**
Codice identificativo della presentazione

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods
- L'Infografica dev'essere inizializzata con i seguenti campi dato:
 - *presid*: il codice identificativo della presentazione
 - *owner*: Il codice identificativo dell'utente che ha chiamato il metodo
 - *content*: Collezione(JSON) vuota
 - *frames*: Collezione(JSON) vuota
 - *type*: infographic

+ updateInfographicById(idInf : String, update : Collection) : void

Aggiorna un'infografica con campi dati modificati dall'utente

Argomenti

- **idInf : String**
Codice identificativo dell'infografica da aggiornare
- **update : Collection**
Collezione di dati con cui aggiornare l'infografica

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ updateGOContentFrame(idGO : String, update : Collection, idFrame : String) : void

Aggiorna un oggetto grafico contenuto all'interno di una presentazione con campi dati modificati dall'utente

Argomenti

- **idGO : String**
Codice identificativo dell'oggetto grafico da modificare
- **update : Collection**
Collezione di dati modificati dell'oggetto grafico
- **idFrame : String**
Codice identificativo del Frame che contiene l'oggetto grafico

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ **insertGOContentFrame(GO : Collection, idFrame : String) : void**

Inserisce un oggetto grafico all'interno di un frame

Argomenti

- **GO : Collection**
Oggetto grafico convertito in JSON
- **idFrame : String**
Codice identificativo del frame in cui inserire l'oggetto grafico

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods
- Ogni oggetto grafico possiede dei metodi per la conversione in JSON. Quest'operazione va sempre effettuata dal client

+ **removeGOContentFrame(idGO : String, idFrame : String) : void**

Rimuove un oggetto grafico dal frame che lo contiene

Argomenti

- **idGO : String**
Codice identificativo dell'oggetto grafico
- **idFrame : String**
Codice identificativo del frame che contiene l'oggetto grafico

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ **updateGOContentInfographic(idGO : String, update : Collection, idInf : String) : void**

Aggiorna un oggetto grafico contenuto all'interno di un'infografica

Argomenti

- **idGO : String**
Codice identificativo dell'oggetto grafico
- **update : Collection**
Collezione di dati dell'oggetto grafico modificati dall'utente
- **idInf : String**
Codice identificativo dell'infografica

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ **insertGOContentInfographic(GO : Collection, idInf : String) : void**

Inserisce un oggetto grafico all'interno di un'infografica

Argomenti

- **GO : Collection**
Oggetto grafico convertito in JSON
- **idInf : String**
Codice identificativo dell'infografica

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods
- Ogni oggetto grafico possiede dei metodi per la conversione in JSON. Quest'operazione va sempre effettuata dal client

+ **removeGOContentInfographic(idGO : String, idInf : String) : void**

Rimuove un oggetto grafico da un'infografica

Argomenti

- **idGO : String**
Codice identificativo dell'oggetto grafico
- **idInf : String**
Codice identificativo dell'infografica

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ insertFrameInfographic(idFrame : String, idInf : String) : void

Inserisce un frame all'interno di un'infografica

Argomenti

- **idFrame : String**
Codice identificativo del frame
- **idInf : String**
Codice identificativo dell'infografica

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ removeFrameInfographics(idFrame : String, idInf : String) : void

Rimuove un frame da un'infografica

Argomenti

- **idFrame : String**
Codice identificativo del frame
- **idInf : String**
Codice identificativo dell'infografica

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ checkUsername(username : String) : boolean

Verifica che l'username ricevuto sia presente tra quelli registrati nel database.

Argomenti

- **username : String**

Il codice identificativo dell'username da verificare

Note

- Dev'essere inserito nel servizio **\$meteor** attraverso il comando Meteor.methods

+ **getInfographicByPresId(id : String) : Collection**

Pubblica una lista di infografiche associate ad una presentazione

Argomenti

- **id : String**

Il codice identificativo della presentazione associata alle infografiche

Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish

+ **getFramesByPresId(id : String) : Collection**

Pubblica una lista di frames associati ad una presentazione

Argomenti

- **id : String**

Il codice identificativo della presentazione associata ai frame

Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish
- La presentazione deve appartenere all'utente che ha effettuato la richiesta

+ **getInfographicFrames(presid : String) : Collection**

Pubblica una lista di tutti i frames delle infografiche di una presentazione

Argomenti

- **presid : String**

Il codice identificativo della presentazione associata alle infografiche

Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish
- La presentazione deve appartenere all'utente che ne ha effettuato la richiesta

+ **getTrailsByPresId(id : String) : Collection**

Pubblica una lista di trails associati ad una presentazione

Argomenti

- **id : String**

Il codice identificativo della presentazione associata ai trails

Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish
- La presentazione deve appartenere all'utente che ne ha effettuato la richiesta

+ **getPresentationsById(id : String) : Collection**

Pubblica una presentazione all'utente

Argomenti

- **id : String**

Il codice identificativo della presentazione

Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish
- La presentazione deve appartenere all'utente che ne ha effettuato la richiesta

+ **getPresentations() : Collection**

Pubblica una lista di tutte le presentazioni che l'utente possiede

Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish
- Le presentazioni devono appartenere all'utente che ne ha effettuato la richiesta

+ getTrailsById(id : String) : Collection

Pubblica un trail all'utente che ne ha effettuato la richiesta

Argomenti

- **id : String**
Il codice identificativo del trail da pubblicare

Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish
- Il trail deve appartenere all'utente che ne ha effettuato la richiesta

3.2 premi/client

3.3 premi/client/presentation

3.4 premi/client/presentationManager

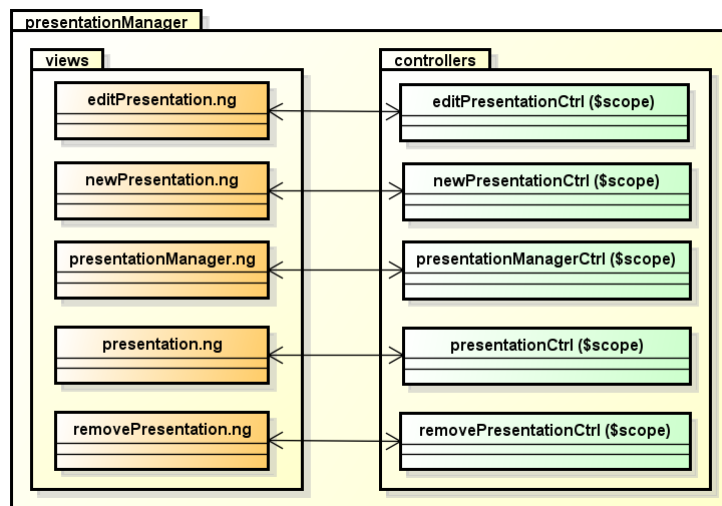


Figura 3: Diagramma del package premi/client/presentationManager

3.4.1 premi/client/presentationManager/views/editPresentation.ng

Descrizione

Template della vista associata allo *\$scope* di *editPresentationCtrl*. Permette all'utente di modificare titolo e descrizione di una presentazione.

Note

- Mostra il titolo della presentazione in un input `HTMLG`, modificabile, attraverso l'attributo dello *\$scope* `Presentation.title`
- Mostra la descrizione della presentazione in un input `HTMLG`, modificabile, attraverso l'attributo dello *scope* `Presentation.description`
- Possiede un bottone associato al metodo *save()* dello *\$scope* per salvare la presentazione
- Possiede un bottone associato al metodo *discard()* dello *\$scope* per annullare le modifiche effettuate sulla presentazione

3.4.2 premi/client/presentationManager/views/newPresentation.ng

Descrizione

Template della vista associata allo *\$scope* di *newPresentationCtrl*. Permette all'utente di creare una nuova presentazione, fornendo un titolo e una descrizione.

Note

- Mostra un input HTML nel quale inserire il titolo della presentazione, che va associato all'attributo *title* dello *\$scope*
- Mostra un input HTML nel quale inserire la descrizione della presentazione, che va associata all'attributo *description* dello *\$scope*
- Possiede un bottone associato al metodo *save()* dello *\$scope* per salvare la presentazione
- Possiede un bottone associato al metodo *discard()* dello *\$scope* per annullare la creazione della presentazione

3.4.3 premi/client/presentationManager/views/presentationManager.ng

Descrizione

Template della vista associata allo *\$scope* di *presentationmagerCtrl*. Fornisce uno scheletro per le altre viste dedicate alla gestione delle presentazioni

3.4.4 premi/client/presentationManager/views/presentations.ng

Descrizione

Template della vista associata allo *\$scope* di *presentationsCtrl*. Mostra una lista di tutte le presentazioni dell'utente

Note

- Mostra la lista delle presentazioni, le quali sono contenute nell'attributo *Presentations* dello *\$scope*
- Per ogni presentazione fornisce dei link per la sua modifica e rimozione

3.4.5 premi/client/presentationManager/views/removePresentation.ng

Descrizione

Template della vista associata allo *\$scope* di *removePresentationCtrl*. Permette all'utente di rimuovere una presentazione dal database.

Note

- Mostra un messaggio di conferma prima di rimuovere la presentazione
- Dopo il messaggio di conferma mostra un tasto associato al metodo *remove()* dello *\$scope* per la rimozione definitiva della presentazione
- Dopo il messaggio di conferma mostra un tasto associato al metodo *discard()* dello *\$scope* per annullare il processo di rimozione della presentazione

3.4.6 premi/client/presentationManager/controllers/editPresentationCtrl

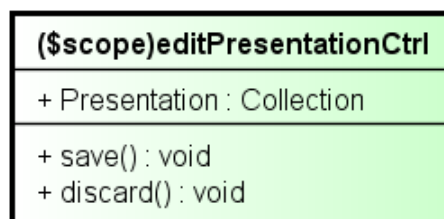


Figura 4: Diagramma della classe premi/client/presentationManager/editPresentationCtrl

Descrizione

Controller della view *editPresentation.ng*. Permette all'utente di modificare titolo e descrizione di una presentazione.

Associazioni

- **client/presentation/lib/databaseAPI**: per salvare la presentazione modificata

Attributi

+ Presentation : Collection

Presentazione che l'utente intende modificare. Viene inizializzata dal controller tramite il codice identificativo passato come parametro dal browser (servizio *\$stateParams* di AngularJS)

Metodi

+ save() : void

Utilizza il metodo *+ updatePresentation(id, title, description)* di *databaseAPI* per il salvataggio della presentazione nel database. Rimanda poi alla lista delle presentazioni utilizzando l'oggetto *\$state* di *\$stateProvider*

+ discard() : void

Annulla le modifiche effettuate dall'utente sul titolo e sulla descrizione della presentazione. Rimanda poi alla lista delle presentazioni utilizzando l'oggetto *\$state* di *\$stateProvider*

3.4.7 premi/client/presentationManager/controllers/newPresentationCtrl

(\$scope)newPresentationCtrl
+ title : String + description : String
+ save() : void + discard() : void

Figura 5: Diagramma della classe premi/client/presentationManager/newPresentationCtrl

Descrizione

Controller della view *editPresentation.ng*. Permette all'utente di creare una nuova presentazione e di salvarla nel database.

Associazioni

- **client/presentation/lib/databaseAPI**: per salvare la presentazione nel database

Attributi

+ **title** : **String**

Il titolo della nuova presentazione

+ **description** : **String**

La descrizione della nuova presentazione

Metodi

+ **save()** : **void**

Utilizza il metodo + *insertNewPresentation(title, description)* di databaseAPI per il salvataggio della presentazione nel database. Rimanda poi alla lista delle presentazioni utilizzando l'oggetto *\$state* di *\$stateProvider*

+ **discard()** : **void**

Annulla le modifiche effettuate dall'utente sul titolo e sulla descrizione della presentazione. Rimanda poi alla lista delle presentazioni utilizzando l'oggetto *\$state* di *\$stateProvider*

3.4.8 premi/client/presentationManager/controllers/presentationManagerCtrl

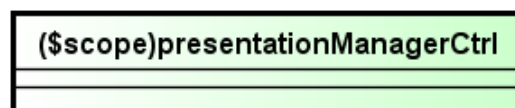


Figura 6: Diagramma della classe premi/client/presentationManager/controllers/presentationManagerCtrl

Descrizione

Questo controller non è al momento provvisto di funzionalità. Si appoggia alla vista associata *presentationManager.ng*, la quale funge da scheletro per le viste necessarie alla gestione delle presentazioni dell'utente.

3.4.9 premi/client/presentationManager/controllers/presentationsCtrl

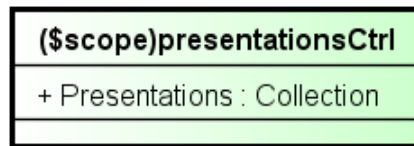


Figura 7: Diagramma della classe premi/client/presentationManager/controllers/presentationsCtrl

Descrizione

Fornisce alla vista associata *presentations.ng* una lista di tutte le presentazioni in possesso dell'utente

Attributi

+ Presentations : Collection

Collezione MongoDB di presentazioni. Viene inizializzata dal controller prelevando le presentazioni pubblicate al client.

3.4.10 premi/client/presentationManager/controllers/removePresentationCtrl

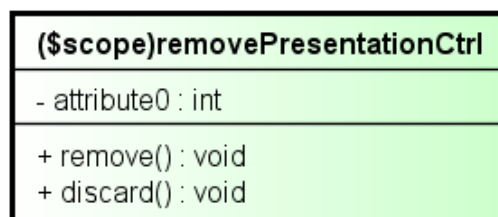


Figura 8: Diagramma della classe premi/client/presentationManager/removePresentationCtrl

Descrizione

Controller della view *removePresentation.ng*. Permette all'utente di eliminare una presentazione da lui creata in precedenza.

Associazioni

- **client/presentation/lib/databaseAPI**: per effettuare la rimozione sul database

Attributi

+ Presentation : Collection

Presentazione che l'utente intende rimuovere. Viene inizializzata dal controller tramite il codice identificativo passato come parametro dal browser (servizio `$stateParams` di AngularJS)

Metodi

+ `remove()` : void

Utilizza il metodo `+ removePresentation(id, title, description)` di `databaseAPI` per la rimozione della presentazione dal database. Rimanda poi alla lista delle presentazioni utilizzando l'oggetto `$state` di `$stateProvider`

+ `discard()` : void

Annulla le modifiche effettuate dall'utente sulla presentazione. Rimanda poi alla lista delle presentazioni utilizzando l'oggetto `$state` di `$stateProvider`

3.5 premi/client/editor

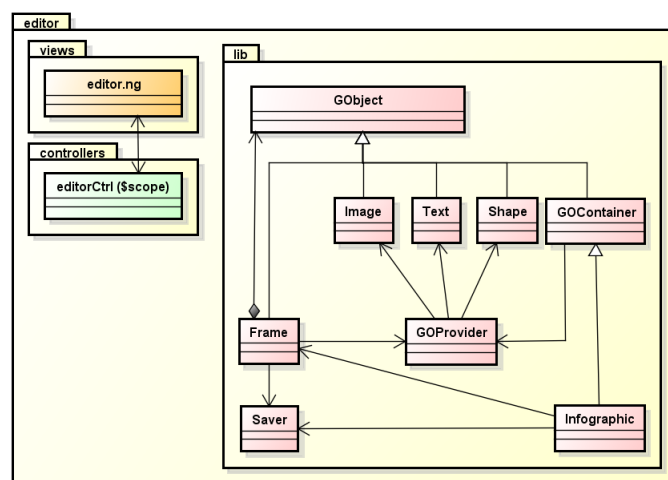


Figura 9: Diagramma della classe premi/client/editor

3.5.1 premi/client/editor/lib/GObject

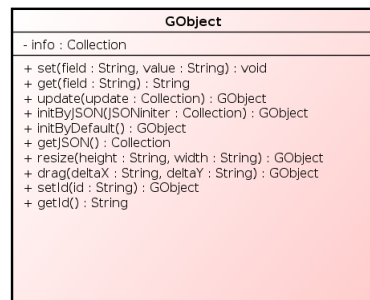


Figura 10: Diagramma della classe premi/client/editor/GObject

Descrizione

GObject è una classe astratta che rappresenta un oggetto generico della presentazione. Contiene i metodi generali che caratterizzano ciascun oggetto grafico che può essere inserito in una presentazione.

Attributi

- info : Collection

info è un oggetto JSON che contiene i seguenti campi:

- *_id*: rappresenta l'id che identifica l'oggetto;
- *dataX*: identifica la posizione orizzontale dell'asse x dell'oggetto;
- *dataY*: identifica la posizione verticale dell'asse y dell'oggetto;
- *dataZ*: identifica il grado di trasparenza dell'oggetto;
- *height*: identifica l'altezza dell'oggetto;
- *width*: identifica la larghezza dell'oggetto;
- *scale*: identifica la scala dell'oggetto;
- *lvl*: identifica il livello dell'oggetto.

Metodi

+ set(field : String, value : String) : void

permette di settare un campo dell'attributo info.

Argomenti

- **field** : **String**
field identifica il campo da settare di info;
- **value** : **String**
value rappresenta il valore del campo da settare su l'attributo info.

+ **get(field : String) : String**

restituisce il valore di un campo dell'attributo info.

Argomenti

- **field** : **String**
field identifica l'attributo info di cui si vuole venga restituito il valore.

+ **update(update : Collection) : GObject**

permette di aggiornare i campi dell'attributo info.

Argomenti

- **update** : **Collection**
update è un oggetto JSON che contiene chiave e valore dei campi che devono essere aggiornati.

+ **initByJSON(JSONniter : Collection) : GObject**

permette di inizializzare l'attributo info tramite un oggetto JSON.

Argomenti

- **JSONniter** : **Collection**
è un oggetto JSON che contiene chiave e valore di inizializzazione dei campi dell'attributo info.

+ **initByDefault() : GObject**

permette di inizializzare i campi dell'attributo info con i parametri di default.

+ **getJSON() : String**

restituisce la collection dell'attributo info.

+ **resize(height : String, width : String) : GObject**

permette di settare l'altezza e la larghezza dell'oggetto.

Argomenti

- **height** : **String**
identifica il valore dell'altezza da settare sul campo height dell'attributo info;
- **width** : **String**
identifica il valore dell'altezza da settare sul campo width dell'attributo info;

+ **drag(deltaX : String, deltaY : String) : GObject**

permette di settare la posizione dell'oggetto sull'asse x e y.

Argomenti

- **deltaX** : **String**
identifica il valore della posizione sull'asse x da settare sul campo deltaX dell'attributo info.
- **deltaY** : **String**
identifica il valore della posizione sull'asse y da settare sul campo deltaY dell'attributo info;

+ **setId(id : String) : GObject**

permette di modificare o di settare l'id dell'oggetto

Argomenti

- **id** : **String**
identifica il valore dell'id da settare sul campo id dell'attributo info.

+ **getId()**

restituisce l'id dell'oggetto.

3.5.2 premi/client/editor/lib/GOProvider

Descrizione

GOProvider è una classe statica contiene i metodi per inizializzare gli oggetti image, text, shape che possono essere inseriti in un frame.

Metodi

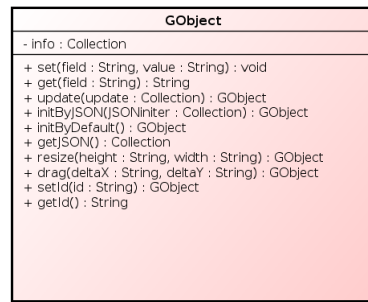


Figura 11: Diagramma della classe premi/client/editor/GObject

+ init(type : String) : GObject

inizializza con i parametri di default un oggetto image, shape o text e ne restituisce il riferimento.

Argomenti

- **type : String**
identifica il tipo di oggetto da inizializzare.

+ initByJSON(GO : Collection) : GObject

inizializza con un oggetto JSON, un oggetto image, text o shape e ne restituisce il riferimento.

Argomenti

- **GO : Collection**
è un oggetto JSON che serve per inizializzare l'oggetto da resituire come riferimento.

3.5.3 premi/client/editor/lib/frame

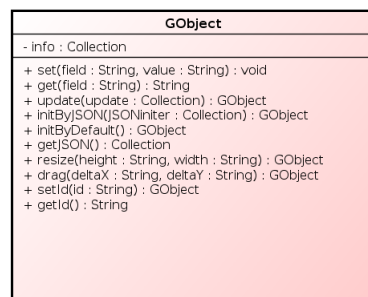


Figura 12: Diagramma della classe premi/client/editor/GObject

Descrizione

frame è una classe che rappresenta un frame di una presentazione. E' un oggetto che può essere rappresentato nella presentazione.

Classi ereditate

- GObject

Attributi

- info : Collection

l'attributo info è un oggetto JSON che estende l'attributo info ereditato da GObject. I campi aggiuntivi sono:

- *backgroundColor*: rappresenta il colore di Background del frame;
- *content*: è un oggetto JSON che contiene gli oggetti che fanno parte del frame;
- *type*: identifica che l'oggetto trattato è un frame.

- selectedGO : GObject

contiene l'oggetto GObject contenuto nel frame corrente selezionato dall'utente.

- saver : Saver

contiene un oggetto saver che permette di interfacciarsi con il database.

Metodi

- findNewId() : String

trova un nuovo id valido per il frame corrente.

+ initByJSON(GO : Collection) : GObject

inizializza con un oggetto JSON, un oggetto image, text o shape e ne restituisce il riferimento.

Argomenti

- G0 : Collection
è un oggetto JSON che serve per inizializzare l'oggetto da resituire come riferimento.

3.6 premi/client/frameEditor

3.7 premi/client/infographicEditor

3.8 premi/client/trailsEditor

3.9 premi/client/userManager

3.10 premi/client/viewer

4 Tracciamento

5 Diagrammi di Sequenza