404NotFound

Premi: better than Prezi.

Definizione di Prodotto

Versione	2.0
Redazione	Gobbo Ismaele Rettore Andrea Vegro Federico Manuto Monica De Lazzari Enrico Cossu Mattia Camborata Marco
Verifica	Vegro Federico Camborata Marco
Responsabile	Camborata Marco
Uso	Esterno
Stato	Formale
Ultima modifica	5 settembre 2015
Lista di distribuzione	404NotFound prof. Tullio Vardanega prof. Riccardo Cardin Zucchetti S.p.a.

Registro delle modifiche

Versione	Autore	Data	Descrizione
1.7	Gobbo Ismaele	31-08-2015	Aggiunte sottosezioni a 3. Collezioni di MongoDB e Gestione degli Account
1.6	Manuto Monica	31-08-2015	Corretto il metodo CurrentUser in currentUser nella sottosezione 3.3.2
1.5	Gobbo Ismaele	31-08-2015	Modificata la descrizione dei metodi di server/methods
1.4	Gobbo Ismaele	31-08-2015	Corretta dimensione di alcune figure
1.3	Gobbo Ismaele	25-08-2015	Descritti, nell'introduzione della sezione 3, i $servizi di AngularJS$_G$ utilizzati dai componenti di Premi
1.2	Rettore Andrea	25-08-2015	Rimossi alcuni view e controller del package premi/client non più utilizzati
1.1	Rettore Andrea	25-08-2015	Aggiunti view e controller del package premi/client/editor
1.0	Cossu Mattia	21-08-2015	Approvazione documento
0.20	De Lazzari Enrico	21-08-2015	verifica classi, corretti metodi di trailsEditor
0.19	Rettore Andrea	19-08-2015	corretti alcuni termini non presenti nel glossario
0.18	De Lazzari Enrico	18-08-2015	Verifica generale delle classi
0.17	Gobbo Ismaele	10-08-2015	Correzioni su trailsEditor
0.16	Cossu Mattia	1-08-2015	aggiunti metodi a trailsEditor, correzioni frameEditor
0.15	Vegro Federico	29-07-2015	Scrittura trailsEditor, aggiunto Trails a presentation/lib
0.14	De Lazzari Enrico	28-07-2015	aggiunti altri controller mancanti, presentationManager
0.13	Manuto Monica	24-07-2015	correzioni su infographicEditor e frameEditor. Aggiunti dei nuovi metodi resi necessari per la modifica del testo
0.12	Camborata Marco	8-07-2015	aggiunte definizioni dei controller di UserManager
0.11	Gobbo Ismaele	8-07-2015	aggiunti interactInit e Observer in editor
0.10	Gobbo Ismaele	6-07-2015	Stesura dei controller principali della classe client
0.9	Manuto Monica	5-07-2015	Aggiunta di metodi alle classi methods e databaseAPI per il salvataggio di trails

0.8	De Lazzari Enrico	27-06-2015	incremento classi frameEditor, infographicEditor
0.7	Cossu Mattia	24-06-2015	definizione classe databaseAPI, correzioni sui metodi di infographicEditor e frameEditor
0.6	Gobbo Ismaele	24-06-2015	Scrittura package infographicEditor
0.5	Cossu Mattia	21-06-2015	scrittura package server
0.4	De Lazzari Enrico	21-06-2015	Scrittura package frameEditor
0.3	Gobbo Ismaele	20-06-2015	Scrittura delle classi di editor/lib
0.2	Vegro Federico	17-06-2015	Scrittura sezione Standard di Progetto
0.1	Gobbo Ismaele	16-06-2015	Stesura scheletro, scrittura introduzione al documento

Tabella 1: Storico versioni del documento.

Indice

1 Introduzione **5**
1.1 Scopo del documento . . . 5
1.2 Scopo del Progetto . . . 5
1.3 Glossario . . . 5
1.4 Riferimenti . . . 5
1.4.1 Normativi . . . 5
1.4.2 Informativi . . . 5

2 Standard di Progetto **6**
2.1 Standard di progettazione architetturale . . . 6
2.2 Standard di documentazione del codice . . . 6
2.3 Standard di denominazione di entità e relazioni . . . 6
2.4 Standard di programmazione . . . 6
2.5 Strumenti di lavoro . . . 6

3 Specifica componenti **7**
3.1 Collezioni di MongoDB . . . 7
3.2 Template . . . 7
3.3 Servizi di AngularJS$_G$ . . . 7
3.4 Gestione degli Account Utente . . . 9
3.5 premi/server . . . 10
3.5.1 premi/server/publish . . . 10
3.5.2 premi/server/methods . . . 13
3.6 premi/client . . . 23
3.6.1 premi/client/views/header.ng . . . 23
3.6.2 premi/client/views/home.ng . . . 23
3.6.3 premi/client/controllers/premi . . . 23
3.6.4 premi/client/lib/toastMessageFactory . . . 23
3.7 premi/client/presentation . . . 24
3.7.1 premi/client/presentation/lib/databaseAPI . . . 24
3.7.2 premi/client/presentations/lib/OrderedGOList . . . 30
3.7.3 premi/client/presentation/lib/Trail . . . 34
3.7.4 premi/client/presentation/lib/signalCtrl . . . 39
3.8 premi/client/presentationManager . . . 42
3.8.1 premi/client/presentationManager/views/editPresentation.ng . . . 42
3.8.2 premi/client/presentationManager/views/newPresentation.ng . . . 42
3.8.3 premi/client/presentationManager/views/presentationManager.ng 43
3.8.4 premi/client/presentationManager/views/presentations.ng . . . 43
3.8.5 premi/client/presentationManager/views/removePresentation.ng 43
3.8.6 premi/client/presentationManager/controllers/editPresentationCtrl 44
3.8.7 premi/client/presentationManager/controllers/newPresentationCtrl 45
3.8.8 premi/client/presentationManager/controllers/presentationManagerCtrl 46
3.8.9 premi/client/presentationManager/controllers/presentationsCtrl 46
3.8.10 premi/client/presentationManager/controllers/removePresentationCtrl 47
3.9 premi/client/editor . . . 49
3.9.1 premi/client/editor/views/editor.ng . . . 49

3.9.2 premi/client/editor/controllers/editorCtrl . . . 49
3.9.3 premi/client/editor/lib/GObject . . . 50
3.9.4 premi/client/editor/lib/GOProvider . . . 52
3.9.5 premi/client/editor/lib/Frame . . . 53
3.9.6 premi/client/editor/lib/GOContainer . . . 58
3.9.7 premi/client/editor/lib/Image . . . 62
3.9.8 premi/client/editor/lib/Infographic . . . 64
3.9.9 premi/client/editor/lib/interactInit . . . 68
3.9.10 premi/client/client/lib/Observer . . . 70
3.9.11 premi/client/editor/lib/saver . . . 72
3.9.12 premi/client/editor/lib/Shape . . . 75
3.9.13 premi/client/editor/lib/Text . . . 77
3.10 premi/client/frameEditor . . . 80
3.10.1 premi/client/frameEditor/views/frame.ng . . . 80
3.10.2 premi/client/frameEditor/controllers/frameEditorCtrl . . . 81
3.11 premi/client/infographicEditor . . . 89
3.11.1 premi/client/infographicEditor/views/infographic.ng . . . 89
3.11.2 premi/client/infographicEditor/controllers/infographicEditorCtrl 90
3.12 premi/client/trailsEditor . . . 97
3.12.1 premi/client/trailsEditor/views/basicToolbar.ng . . . 97
3.12.2 premi/client/trailsEditor/views/editTrail.ng . . . 97
3.12.3 premi/client/trailsEditor/views/listTrail.ng . . . 98
3.12.4 premi/client/trailsEditor/views/modTrail.ng . . . 98
3.12.5 premi/client/trailsEditor/views/newTrail.ng . . . 98
3.12.6 premi/client/trailsEditor/views/removeTrail.ng . . . 99
3.12.7 premi/client/trailsEditor/controllers/basicToolbarCtrl . . . 99
3.12.8 premi/client/trailsEditor/controllers/editTrailCtrl . . . 100
3.12.9 premi/client/trailsEditor/controllers/listTrailCtrl . . . 101
3.12.10 premi/client/trailsEditor/controllers/modTrailCtrl . . . 102
3.12.11 premi/client/trailsEditor/controllers/newTrailCtrl . . . 105
3.12.12 premi/client/trailsEditor/controllers/removeTrailCtrl . . . 106
3.12.13 premi/client/trailsEditor/controllers/trailsEditorCtrl . . . 107
3.13 premi/client/userManager . . . 109
3.13.1 premi/client/userManager/views/changePassword.ng . . . 109
3.13.2 premi/client/userManager/views/signin.ng . . . 109
3.13.3 premi/client/userManager/views/signup.ng . . . 109
3.13.4 premi/client/userManager/views/userManager.ng . . . 110
3.13.5 premi/client/userManager/controllers/changePasswordCtrl . . . 110
3.13.6 premi/client/userManager/controllers/signinCtrl . . . 111
3.13.7 premi/client/userManager/controllers/signoutCtrl . . . 112
3.13.8 premi/client/userManager/controllers/signupCtrl . . . 113
3.14 premi/client/viewer . . . 115
3.14.1 premi/client/viewer/views/trails.ng . . . 115
3.14.2 premi/client/viewer/views/viewer.ng . . . 115
3.14.3 premi/client/viewer/controllers/trailsCtrl . . . 116
3.14.4 premi/client/viewer/controllers/viewerCtrl . . . 117

4 Tracciamento **119**

Elenco delle tabelle

1 Storico versioni del documento. 2

Elenco delle figure

1 Diagramma del package premi/client 10
2 Diagramma della classe premi/server/publish 10
3 Diagramma della classe premi/server/methods 13
4 Diagramma del package premi/client 23
5 Diagramma del package premi/client/presentation 24
6 Diagramma della classe premi/client/presentation/lib/databaseAPI . . 24
7 Diagramma della classe premi/client/presentations/lib/OrderedGOList 30
8 Diagramma della classe premi/client/presentation/lib/Trail 34
9 Diagramma della classe premi/client/presentation/lib/signalCtrl 39
10 Diagramma del package premi/client/presentationManager 42
11 Diagramma della classe premi/client/presentationManager/controllers/editPresentationCtrl 44
12 Diagramma della classe premi/client/presentationManager/controllers/newPresentationCtrl 45
13 Diagramma della classe premi/client/presentationManager/controllers/presentationManagerCtrl 46
14 Diagramma della classe premi/client/presentationManager/controllers/presentationsCtrl 46
15 Diagramma della classe premi/client/presentationManager/controllers/removePresentationCtrl 47
16 Diagramma della classe premi/client/editor 49
17 Diagramma della classe premi/client/editor/controllers/editorCtrl . . . 49
18 Diagramma della classe premi/client/editor/lib/GObject 50
19 Diagramma della classe premi/client/editor/lib/GOProvider 52
20 Diagramma della classe premi/client/editor/lib/Frame 53
21 Diagramma della classe premi/client/editor/lib/GOContainer 58
22 Diagramma della classe premi/client/editor/lib/Image 62
23 Diagramma della classe premi/client/editor/lib/infographic 64
24 Diagramma della classe premi/client/editor/lib/InteractInit 68
25 Diagramma della classe premi/client/editor/lib/Observer 70
26 Diagramma della classe premi/client/editor/lib/Saver 72
27 Diagramma della classe premi/client/editor/lib/Shape 75
28 Diagramma della classe premi/client/editor/lib/Text 77
29 Diagramma del package premi/client/frameEditor 80
30 Diagramma della classe premi/client/frameEditor/controllers/frameEditorCtrl 81
31 Diagramma del package premi/client/infographicEditor 89
32 Diagramma della classe premi/client/infographicEditor/controllers/infographicEditorCtrl 90
33 Diagramma del package premi/client/trailsEditor 97

34 Diagramma della classe premi/client/trailsEditor/controllers/basicToolbarCtrl . . . 99
35 Diagramma della classe premi/client/trailsEditor/controllers/editTrailCtrl . . . 100
36 Diagramma della classe premi/client/trailsEditor/controllers/listTrailCtrl . . . 101
37 Diagramma della classe premi/client/trailsEditor/controllers/modTrailCtrl . . . 102
38 Diagramma della classe premi/client/trailsEditor/controllers/newTrailCtrl . . . 105
39 Diagramma della classe premi/client/trailsEditor/controllers/removeTrailCtrl . . . 106
40 Diagramma della classe premi/client/trailsEditor/controllers/trailsEditorCtrl . . . 107
41 Diagramma del package premi/client/userManager . . . 109
42 Diagramma della classe premi/client/userManager/controllers/changePasswordCtrl . . . 110
43 Diagramma della classe premi/client/userManager/controllers/signinCtrl 111
44 Diagramma della classe premi/client/userManager/controllers/signoutCtrl . . . 112
45 Diagramma della classe premi/client/userManager/controllers/signupCtrl 113
46 Diagramma del package premi/client/viewer . . . 115
47 Diagramma della classe premi/client/viewer/controllers/trailsCtrl . . . 116
48 Diagramma della classe premi/client/viewer/controllers/viewerCtrl . . . 117

1 Introduzione

1.1 Scopo del documento

Questo documento approfondisce la definizione della struttura e dei componenti di Premi già discussa nella *Specifica Tecnica v3.0*. 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 v4.0*.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:** *Norme di Progetto v4.0*;
- **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

2.1 Standard di progettazione architetturale

I diagrammi inseriti in questo documento seguono lo standard UML_G 2.x. Per ulteriori informazioni sugli standard utilizzati si rimanda ai documenti *Specifica Tecnica v3.0* e *Norme di Progetto v4.0*.

2.2 Standard di documentazione del codice

Per gli standard di documentazione del codice consultare la sezione apposita delle *Norme di Progetto v4.0*.

2.3 Standard di denominazione di entità e relazioni

Ogni package, classe, metodo, attributo, template o semplice file di codice deve avere un nome chiaro ed esplicito che rappresenti la funzione che esso svolge all'interno del software.
In particolare:

- i package dei componenti principali del client devono avere le loro classi interne suddivise in tre ulteriori package interni: `views` per i template delle viste, `controllers` per i controllers, e `lib` per le classi che fungono da modello per i dati dell'applicazione
- componenti marcati `template` sono denominati con uno o due termini che indicano la loro funzione, seguiti da `.ng`
- componenti marcati `controller` hanno il nome del template ad essi associato seguito da `Ctrl`

I termini scelti devono essere in lingua inglese, ed è preferibile non utilizzare abbreviazioni se la lunghezza del nome non risulta eccessiva.
Per ulteriori informazioni si rimanda al documento *Norme di Progetto v4.0*.

2.4 Standard di programmazione

Per gli standard di programmazione consultare il documento *Norme di Progetto v4.0* nella sezione apposita.

2.5 Strumenti di lavoro

Gli strumenti di lavoro utilizzati sono descritti in dettaglio nel documento *Norme di Progetto v4.0*.

3 Specifica componenti

Vengono qui descritti i metodi e gli attributi di ogni componente dell'applicazione.

3.1 Collezioni di MongoDB

Le informazioni in Meteor vengono salvate all'interno di collezioni di documenti, i quali non sono altro che collezioni di coppie di chiavi e valori. Sono facilmente manipolabili poiché si convertono naturalmente in oggetti JavaScript$_G$, e documenti della stessa collezione non devono per forza contenere le stesse chiavi (MongoDB possiede uno schema dinamico).
All'avvio di MeteorJS$_G$ vengono create, se non sono già presenti, quattro collezioni di documenti:

- **Presentations** contiene tutte le presentazioni create dagli utenti;
- **Trails** contiene i percorsi di tutte le presentazioni;
- **Frames** contiene le slides di tutte le presentazioni;
- **Infographics** contiene l'infografica di ogni presentazione;

3.2 Template

Componenti marcati come *template* sono speciali pagine HTML che non possiedono metodi e attributi propri, ma utilizzano quelli forniti dal loro controller dedicato attraverso lo *$scope* (l'oggetto di AngularJS$_G$ che funge da ViewModel$_G$); per questo motivo saranno solamente descritti tramite note e raccomandazioni che specificano come essi devano essere costruiti.

3.3 Servizi di AngularJS$_G$

I servizi di AngularJS sono oggetti intercambiabili che vengono collegati tra loro attraverso il pattern Dependency Injection$_G$. Sono quasi sempre dei Singleton$_G$, e vengono istanziati solo quando un componente dipende da loro. I principali servizi utilizzati per lo sviluppo di questa applicazione sono:

- **$scope** è il collegamento tra i controller e le view dell'applicazione. Il controller modella lo $scope con attributi e metodi pubblici, che la view utilizza per mostrare e modificare il database in tempo reale. Ogni applicazione che utilizza AngularJS ha un solo $scope principale chiamato $rootScope, ma può avere svariati $scope figli aggiunti in una struttura ad albero che rispecchia esattamente la struttura a stati delle view del sito.
  Nei diagrammi dei package di questo documento lo $scope è rappresentato simbolicamente come classe esterna posta tra la view e il controller, mentre a livello implementativo viene invece definito all'interno del controller associato. Nella definizione dei controller quindi:

  - **tutti i gli attributi e i metodi pubblici dei controller vanno inseriti nello $scope**;
  - **tutti i gli attributi e i metodi privati dei controller appartengono al controller**.

Questo esempio mostra la dichiarazione di un metodo (privato) in un controller che accede ad attributi pubblici dello $scope:

```
var var_init = function () {
  $scope.emailState = "";
  $scope.passwordState = "";
};
```

quest'altro esempio invece mostra la dichiarazione di un metodo (pubblico) dello $scope all'interno del controller che accede al metodo privato del controller mostrato prima:

```
$scope.initVariables = function () {
  var_init();
};
```

Lo $scope ha accesso completo al controller, mentre la view può solo utilizzare quello che è stato definito nello $scope;

- **$rootScope** è lo $scope radice da cui discendono tutti gli altri $scope dell'applicazione;
- **$meteor** è un servizio creato da Urigo: Angular-Meteor$_G$ per permettere di accedere alle funzionalità di Meteor all'interno dei moduli di AngularJS;
- **$state** rappresenta lo stato, o la posizione, in cui l'utente si trova all'interno dell'applicazione. Viene fornito dal package esterno AngularUI Router per AngularJS, che vede il re-indirizzamento dell'utente all'interno dell'applicazione come lo spostamento attraverso una macchina a stati;
- **$stateProvider** è il servizio dentro cui vengono registrati gli stati dell'applicazione;
- **$stateParams** è una lista di parametri passati dallo stato precedente a quello corrente;
- **$location** rende l'URL nella barra degli indirizzi del browser disponibile all'applicazione: modifiche all'URL nella barra degl indirizzi si riflettono dentro $location e viceversa;
- **$locationProvider** permette la configurazione del servizio $locationProvider.

3.4 Gestione degli Account Utente

Il Framework MeteorJS$_G$ si occupa della gestione dell'account degli utenti che utilizzano l'applicazione. Le informazioni dell'utente sono consultabili attraverso la variabile `currentUser` in $rootScope, mentre all'interno dei metodi del server che pubblicano le informazioni all'utente è necessario usare `this.userid`.

3.5 premi/server

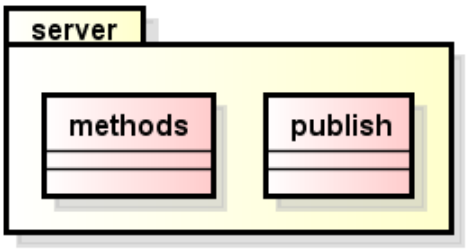

Figura 1: Diagramma del package premi/client

3.5.1 premi/server/publish

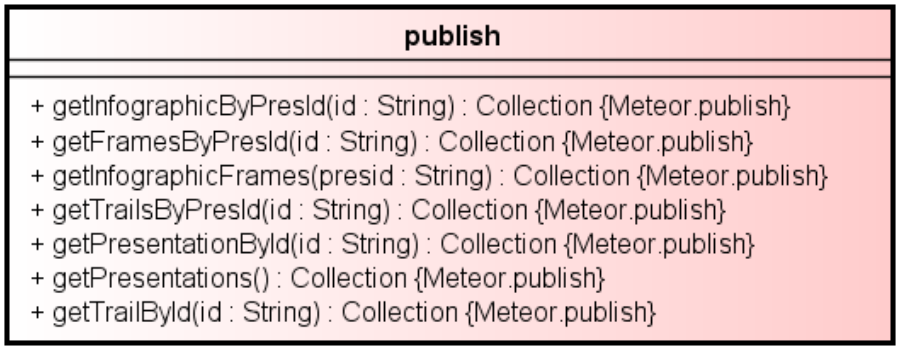

Figura 2: Diagramma della classe premi/server/publish

Descrizione

Questi metodi trasferiscono informazioni dal database MongoDB al database minimongo, prelevando solamente le informazioni di cui l'utente ha bisogno e a cui ha accesso.
Vengono pubblicati all'interno di Meteor e sono sempre acessibili in ogni parte dell'applicazione. Il seguente esempio mostra come vengono dichiarati:

```
Meteor.publish('nomeMetodo', function publishFunction(parametro1,
    parametro2,...) {
  /* ... */
    return Collezione.find(...);
});
```

Le informazioni vengono prelevate dalle Collezioni di Meteor, che sono il modo con cui le informazioni persistenti vengono archiviate (vedi 3.1).
Questi metodi vengono solitamente chiamati all'interno dei file routes.js contenuti in ognuno dei package del client.

Metodi

+ **getInfographicByPresId(id : String) : Collection**
  Restituisce una collezione di infografiche associate ad una presentazione, cercandola nella collezione Infographics creata automaticamente al primo avvio del database MongoDB.

  Argomenti

  - **id : 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 sta effettuando la richiesta (il codice identificativo dell'utente viene individuato tramite `this.userId`)

+ **getFramesByPresId(id : String) : Collection**
  Restituisce una collezione di frames associati ad una presentazione, cercandola nella collezione Frames creata automaticamente al primo avvio del database MongoDB.

  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 ne sta effettuando la richiesta (il codice identificativo dell'utente viene individuato tramite `this.userId`)

+ **getInfographicFrames(presid : String) : Collection**
  Restituisce una collezione di tutti i Frame contenuti nell'infografica di una presentazione, cercando prima l'Infografica associata al codice identificativo ricevuto, e successivamente cercando i Frame. Consulta le collezioni Infographics e Frames create automaticamente al primo avvio del database MongoDB.

  Argomenti

  - **presid : String** Il codice identificativo della presentazione associata alle infografiche

  Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish
- La presentazione in cui è contenuta l'infografica deve appartenere all'utente che ne sta effettuando la richiesta (il codice identificativo dell'utente viene individuato tramite `this.userId`)

+ getTrailsByPresId(id : String) : Collection

Restituisce una collezione di Trails associati ad una presentazione, cercandola nella collezione Trails creata automaticamente al primo avvio del database MongoDB.

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 sta effettuando la richiesta (il codice identificativo dell'utente viene individuato tramite `this.userId`)

+ getPresentationsById(id : String) : Collection

Restituisce la presentazione associata al codice identificativo ricevuto, cercandola nella collezione Presentations creata automaticamente al primo avvio del database MongoDB.

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 sta effettuando la richiesta (il codice identificativo dell'utente viene individuato tramite `this.userId`)

+ getPresentations() : Collection

Restituisce una collezione di tutte le presentazioni create dall'utente, cercandole nella collezione Presentations creata automaticamente al primo avvio del database MongoDB.

Note

- Dev'essere inserito come funzione **publishfunction** in Meteor.publish
- Le presentazioni devono appartenere all'utente che ne sta effettuando la richiesta (il codice identificativo dell'utente viene individuato tramite `this.userId`)

+ getTrailsById(id : String) : Collection

Restituisce il Trail associato al codice identificativo ricevuto, cercandolo nella collezione Trails creata automaticamente al primo avvio del database MongoDB.

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 (il codice identificativo dell'utente viene individuato tramite `this.userId`)

3.5.2 premi/server/methods

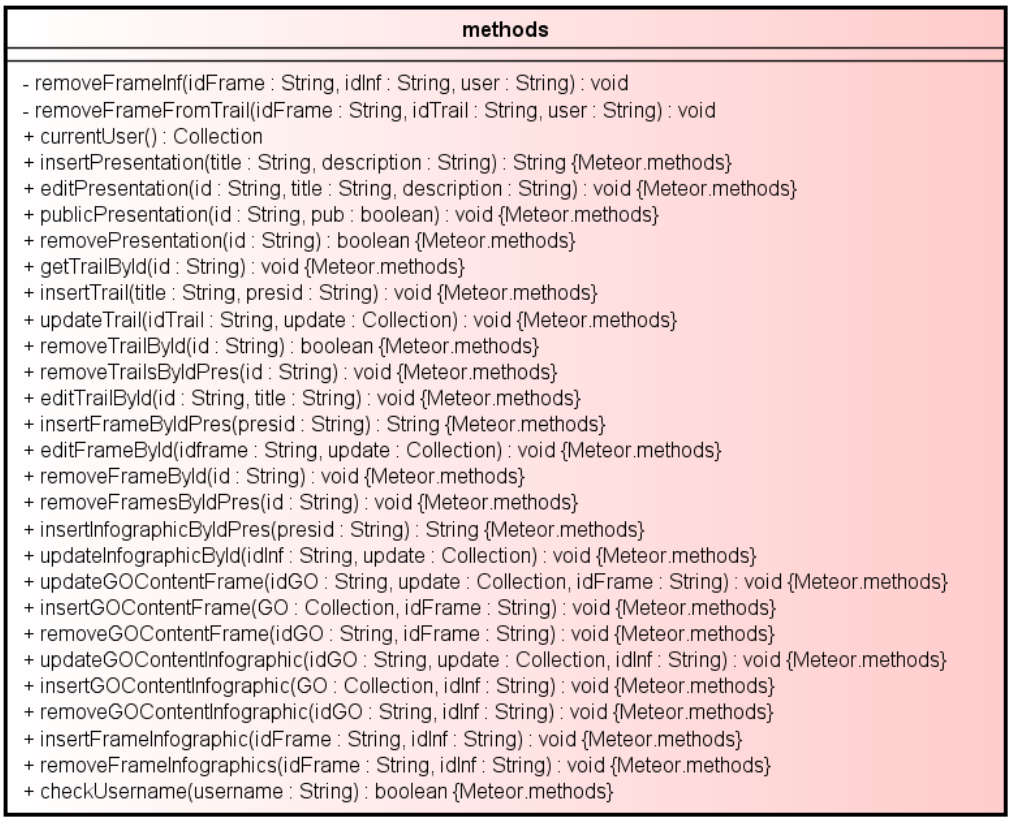

Figura 3: Diagramma della classe premi/server/methods

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$_G$. Il codice identificativo dell'utente viene individuato tramite `this.userId`.

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 ha effettuato la chiamata al metodo, tramite `Meteor.user()`

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 effetuato 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

3.6 premi/client

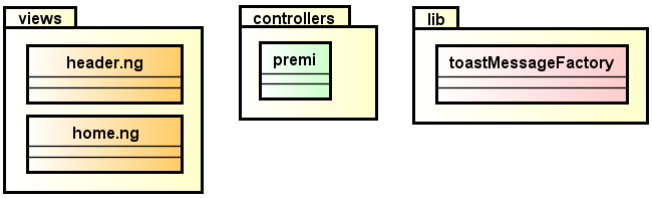

Figura 4: Diagramma del package premi/client

3.6.1 premi/client/views/header.ng

Descrizione

Template che genera un header per la pagina principale dell'applicazione

Note

- Deve mostrare il nome dell'utente, se loggato, oppure un pulsante per loggarsi
- Deve mostrare un pulsante per accedere alla pagina di modifica della password, se l'utente è loggato

3.6.2 premi/client/views/home.ng

Descrizione

Template della pagina principale dell'applicazione

Note

- Deve mostrare il logo dell'applicazione
- Deve mostrare un pulsante per registarsi, se l'utente non è loggato

3.6.3 premi/client/controllers/premi

Descrizione

Controller della pagina principale dell'applicazione. Rende *currentUser* reattivo, ossia utilizza il metodo getReactively di urigo:angular-meteor$_G$ per fare in modo che venga inviato un segnale ogni volta che currentUser cambia

3.6.4 premi/client/lib/toastMessageFactory

Descrizione

Semplice factory che restituisce una funzione per l'invio di notifiche o messaggi di errore all'utente

3.7 premi/client/presentation

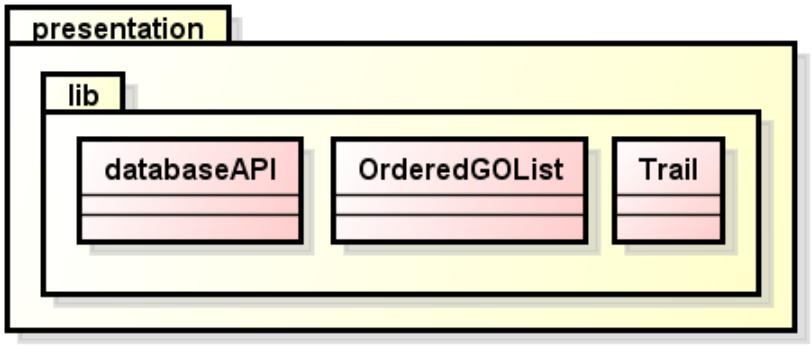

Figura 5: Diagramma del package premi/client/presentation

3.7.1 premi/client/presentation/lib/databaseAPI

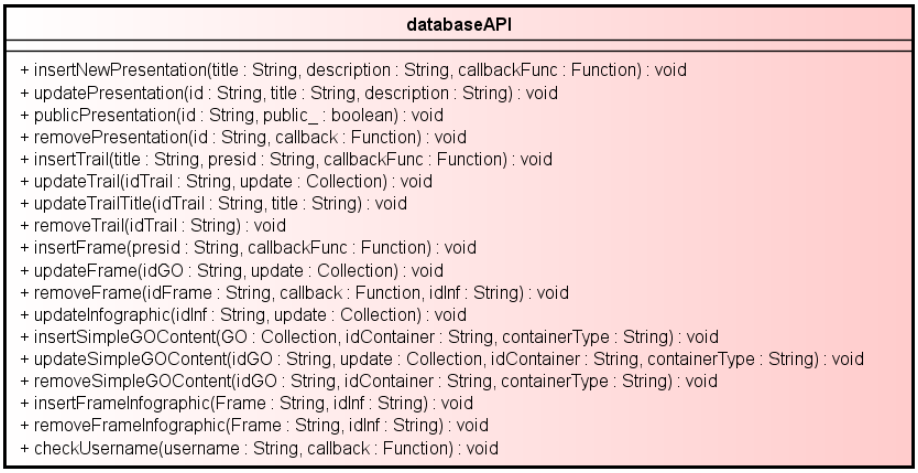

Figura 6: Diagramma della classe premi/client/presentation/lib/databaseAPI

Descrizione

Classe di metodi statici che permette al client di interfacciarsi ai metodi corrispondenti del server

Metodi

+ insertNewPresentation(title : String, description : String, callbackFunc : Function) : void

Permette l'inserimento di una nuova presentazione all'interno del database

Argomenti

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

- **description : String** Descrizione della nuova presentazione
- **callbackFunc : Function** Funzione di $callback_G$ per la restituzione dell'id della presentazione

Note

- Chiama il metodo *insertPresentation* pubblicato in *$meteor*

+ updatePresentation(id : String, title : String, description : String) : void

Permette l'aggiornamento del titolo e della descrizione di una presentazione dell'utente

Argomenti

- **id : String** Codice identificativo della presentazione da aggiornare
- **title : String** Nuovo titolo della presentazione
- **description : String** Nuova descrizione della presentazione

Note

- Chiama il metodo *editPresentation* pubblicato in *$meteor*

+ publicPresentation(id : String, public_ : boolean) : void

Rende una presentazione pubblica o privata

Argomenti

- **id : String** Codice identificativo della presentazione da aggiornare
- **public_ : boolean** Variabile booleana: se *true* la presentazione verrà resa pubblica, se *false* verrà resa privata

Note

- Chiama il metodo *publicPresentation* pubblicato in *$meteor*

+ removePresentation(id : String, callback : Function) : void

Permette la rimozione di una presentazione dell'utente dal database

Argomenti

- **id : String** Codice identificativo della presentazione da rimuovere
- **callbackFunc : Function** Funzione di $callback_G$ per la restituzione di una conferma dell'avvenuta rimozione della presentazione

Note

- Chiama il metodo *removePresentation* pubblicato in *$meteor*

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

Permette l'inserimento di un trail all'interno del database

Argomenti

- **title : String** Titolo del trail da inserire
- **presid : String** Codice identificativo della presentazione associata al trail
- **callbackFunc : Function** Funzione di callback_G per la restituzione del codice identificativo del nuovo trail creato

Note

- Chiama il metodo *insertTrail* pubblicato in *$meteor*

+ updateTrail(idTrail : String, update : Collection) : void

Permette l'aggiornamento di un trail

Argomenti

- **idTrail : String** Codice identificativo del trail da aggiornare
- **update : Collection** Collezione di MongoDB degli attributi aggiornati del trail

Note

- Chiama il metodo *updateTrail* pubblicato in *$meteor*

+ updateTrailTitle(idTrail : String, title : String) : void

Permette la modifica del titolo di un trail

Argomenti

- **idTrail : String** Codice identificativo del trail da aggiornare
- **title : String** Nuovo titolo del trail

Note

- Chiama il metodo *updateTrailById* pubblicato in *$meteor*

+ removeTrail(idTrail : String) : void

Permette la rimozione di un trail dal database

Argomenti

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

Note

- Chiama il metodo *removeTrailById* pubblicato in *$meteor*

+ insertFrame(presid : String, callbackFunc : Function) : void

Permette l'inserimento di un frame all'interno del database

Argomenti

- **presid : String** Codice identificativo della presentazione associata al nuovo frame
- **callbackFunc : Function** Funzione di callback_G per la restituzione del codice identificativo del nuovo frame

Note

- Chiama il metodo *insertFrameByIdPres* pubblicato in *$meteor*

+ updateFrame(idGO : String, update : Collection) : void

Permette l'aggiornamento di un frame all'interno del database

Argomenti

- **idGO : String** Codice identificativo del frame da aggiornare
- **update : Collection** Collezione di MongoDB di attributi aggiornati

Note

- Chiama il metodo *editFrameById* pubblicato in *$meteor*

+ removeFrame(idFrame : String, callback : Function, idInf : String) : void

Permette la rimozione di un frame dal database

Argomenti

- **idFrame : String** Codice identificativo del frame da rimuovere
- **callback : Function** Funzione di callback_G per la restituzione del codice identificativo del frame

Note

- Chiama il metodo *removeFrameInfographic* pubblicato in *$meteor* per rimuovere le associazioni del frame con l'infografica della presentazione
- Chiama il metodo *removeFrameById* pubblicato in *$meteor* per rimuovere il frame dal database

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

Permette l'aggiornamento di un'infografica

Argomenti

- **idInf : String** Codice identificativo dell'infografica da aggiornare
- **update : Collection** Collezione di MongoDB di attributi aggiornati

Note

- Chiama il metodo *updateInfographicById* pubblicato in *$meteor*

+ insertSimpleGOContent(GO : Collection, idContainer : String, containerType : String) : void

Inserisce un oggetto grafico all'interno del database

Argomenti

- **GO : Collection** Collezione degli attributi dell'oggetto grafico
- **idContainer : String** Codice identificativo del contenitore in cui inserire l'oggetto grafico
- **ContainerType : String** Tipo del contenitore: puo' essere un frame(*frame*)) o un'infografica (*infographic*)

Note

- Chiama il metodo *insertGOContentFrame* pubblicato in *$meteor* se il contenitore è un frame, mentre chiama il metodo *insertGOContentInfographic* se il contenitore è un'infografica

+ updateSimpleGOContent(idGO : String, update : Collection, idContainer : String, containerType : String) : void

Aggiorna un oggetto grafico con gli attributi modificati dall'utente

Argomenti

- **idGO : String** Codice identificativo dell'oggetto grafico da aggiornare
- **update : Collection** Collezione di MongoDB di attributi aggiornati
- **idContainer : String** Codice identificativo del contenitore in cui l'oggetto grafico è stato inserito
- **ContainerType : String** Tipo del contenitore: puo' essere un frame(*frame*)) o un'infografica (*infographic*)

Note

- Chiama il metodo *updateGOContentFrame* pubblicato in *$meteor* se il contenitore è un frame, mentre chiama il metodo *updateGOContentInfographic* se il contenitore è un'infografica

+ removeSimpleGOContent(idGO : String, update : Collection, idContainer : String, containerType : String) : void

Aggiorna un oggetto grafico con gli attributi modificati dall'utente

Argomenti

- **idGO : String** Codice identificativo dell'oggetto grafico da aggiornare
- **update : Collection** Collezione di MongoDB di attributi aggiornati
- **idContainer : String** Codice identificativo del contenitore in cui l'oggetto grafico è stato inserito
- **ContainerType : String** Tipo del contenitore: puo' essere un frame(*frame*)) o un'infografica (*infographic*)

Note

- Chiama il metodo *updateGOContentFrame* pubblicato in *$meteor* se il contenitore è un frame, mentre chiama il metodo *updateGOContentInfographic* se il contenitore è un'infografica

+ removeSimpleGOContent(idGO : String, idContainer : String, containerType : String) : void

Rimuove un oggetto grafico da un frame o da un'infografica

Argomenti

- **idGO : String** Codice identificativo dell'oggetto grafico da rimuovere
- **idContainer : String** Codice identificativo del contenitore in cui l'oggetto grafico è stato inserito
- **ContainerType : String** Tipo del contenitore: puo' essere un frame(*frame*)) o un'infografica (*infographic*)

Note

- Chiama il metodo *removeGOContentFrame* pubblicato in *$meteor* se il contenitore è un frame, mentre chiama il metodo *removeGOContentInfographic* se il contenitore è un'infografica

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

Inserisce un frame all'interno di un'infografica

Argomenti

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

Note

- Chiama il metodo *insertFrameInfographic* pubblicato in *$meteor*

+ removeFrameInfographic(Frame : String, idInf : String) : void

Rimuove un frame all'interno di un'infografica

Argomenti

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

Note

- Chiama il metodo *removeFrameInfographic* pubblicato in *$meteor*

+ checkUsername(username : String, callback : Function) : void

Verifica la presenza di uno username nel database

Argomenti

- **username : String** Nome utente da cercare nel database
- **callback : Function** Funzione callback$_G$ per confermare la presenza o l'assenza dello username nel database

Note

- Chiama il metodo *checkUsername* pubblicato in *$meteor*. Restituisce una variabile booleana che andrà passata alla funzione callback$_G$

3.7.2 premi/client/presentations/lib/OrderedGOList

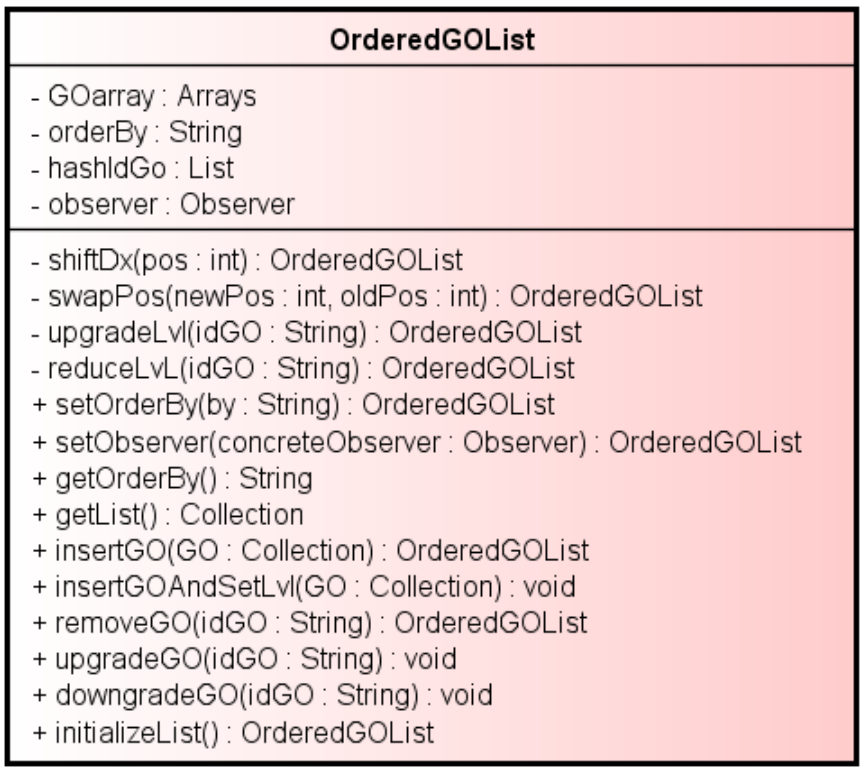

Figura 7: Diagramma della classe premi/client/presentations/lib/OrderedGOList

Descrizione

Questa classe gestisce una lista ordinata di oggetti grafici da poter essere utilizzata per i frame e l'infografica di una presentazione.

Dipendenze

- **premi/client/editor/lib/Observer**: per l'invio di segnali che avvertono gli altri componenti delle modifiche apportate agli oggetti grafici

Attributi

- GOarray : Array
Array di oggetti grafici che compongono la lista ordinata

- orderBy : String
Indica il nome dell'attributo che è stato scelto per ordinare gli oggetti grafici

- hashIdGo : List
Oggetto Hash_G Javascript_G che contiene una lista degli id degli oggetti grafici presenti nell'array, associati alla loro posizione all'interno dell'array

- observer : Observer
Contiene una lista di segnali, ognuno dei quali è associato ad uno specifico oggetto grafico.

Metodi

- shiftDx(pos : int) : OrderedGOList
Sposta l'oggetto grafico presente nella posizione ricevuta nella posizione successiva dell'array. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

Argomenti

- **pos : int** Posizione dell'oggetto da spostare

- swapPos(newPos : int, oldPos : int) : OrderedGOList
Scambia di posizione un oggetto grafico. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

Argomenti

- **newPos : int** La nuova posizione in cui spostare l'oggetto grafico
- **oldPos : int** La posizione in cui si trova l'oggetto grafico prima dell'esecuzione del metodo

- upgradeLvl(idGO : String) : OrderedGOList
Incrementa il livello di visibilità dell'oggetto grafico associato al codice identificativo ricevuto. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

Argomenti

- **idGO : String** Il codice identificativo dell'oggetto grafico da modificare

- reduceLvl(idGO : String) : OrderedGOList

Riduce il livello di visibilità dell'oggetto grafico associato al codice identificativo ricevuto. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

Argomenti

- **idGO : String** Il codice identificativo dell'oggetto grafico da modificare

+ setOrderBy(by : String) : OrderedGOList

Cambia l'attributo con il quale la classe ordina gli oggetti grafici. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

Argomenti

- **by : String** Il nome dell'attributo con il quale si intende stabilire l'ordine degli oggetti grafici

+ setObserver(concreteObserver : Observer) : OrderedGOList

Imposta l'Observer della classe. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

Argomenti

- **concreteObserver : Observer** L'oggetto Observer da associare alla classe

+ getOrderBy() : String

Restituisce il nome dell'attributo col quale si sta effettuando l'ordinamento degli oggetti grafici

+ getList() : Collection

Restituisce l'array degli oggetti grafici inseriti finora all'interno della lista.

+ insertGO(GO : Collection) : OrderedGOList

Inserisce un oggetto grafico convertito precedentemente in formato JSON_G all'interno della lista. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

Argomenti

- **GO : Collection** L'oggetto grafico da inserire nella lista, già convertito in formato JSON

+ insertGOAndSetLvl(GO : Collection) : void

Inserisce un oggetto grafico convertito precedentemente in formato $JSON_G$ alla fine della lista, e tramite l'Observer invia un segnale di cambio livello dell'oggetto

Argomenti

- **GO : Collection** L'oggetto grafico da inserire nella lista, già convertito in formato JSON

+ removeGO(idGO : String) : OrderedGOList

Rimuove un oggetto grafico dalla lista. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

Argomenti

- **idGO : String** Il codice identificativo dell'oggetto grafico da rimuovere dalla lista

+ upgradeGO(idGO : String) : void

Incrementa di posizione un oggetto grafico nella lista

Argomenti

- **idGO : String** Il codice identificativo dell'oggetto grafico da incrementare di posizione

Note

- Sfrutta il metodo privato *swapPos* per li spostamento dell'oggetto grafico

+ downgradeGO(idGO : String) : void

Decrementa di posizione un oggetto grafico nella lista

Argomenti

- **idGO : String** Il codice identificativo dell'oggetto grafico da decrementare di posizione

Note

- Sfrutta il metodo privato *swapPos* per li spostamento dell'oggetto grafico

+ initializeList() : OrderedGOList

Inizializza la lista svuotando *GOarray* e *hashIdGO*. Restituisce un riferimento al *this* per chiamate multiple ai metodi della classe

3.7.3 premi/client/presentation/lib/Trail

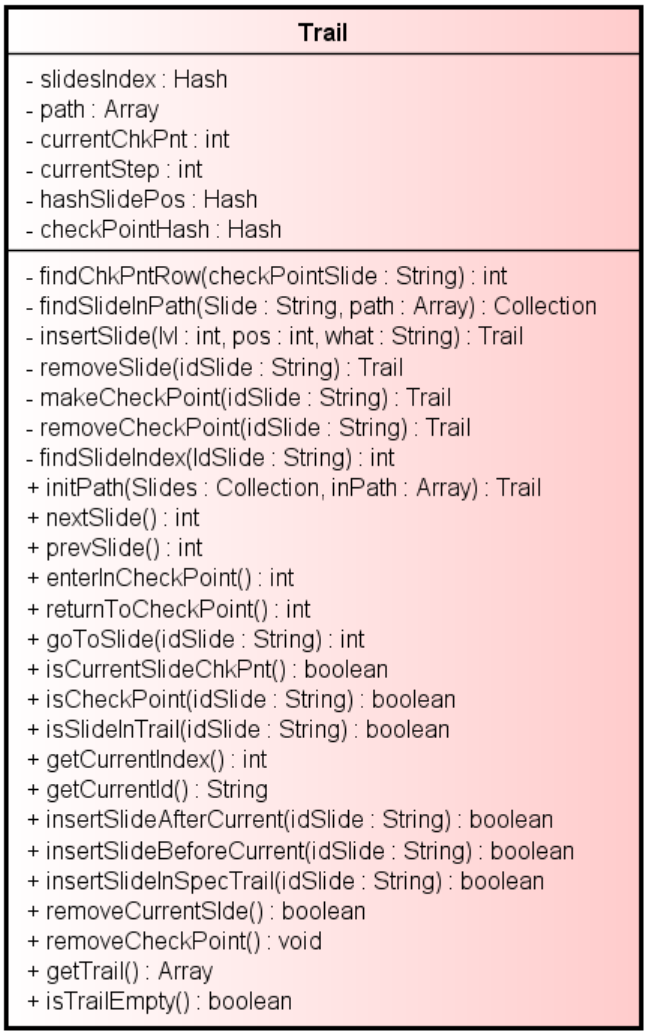

Figura 8: Diagramma della classe premi/client/presentation/lib/Trail

Descrizione

Trail modella un percorso di presentazione, attraverso una lista delle slide contenute al suo interno, e una matrice che rappresenta la struttura del percorso e segnala la presenza di percorsi di specializzazione.

Attributi

- slidesIndex : Hash

Lista di tutte le slide contenute all'interno della presentazione create dall'utente. Contiene quindi anche quelle non aggiunte al percorso. È una lista di valori { `id_slide` : `posizione nell'array di slide` }

- path : Array

Matrice rettangolare che rappresenta il percorso di specializzazione. Contiene i codici identificativi delle slide presenti nel percorso; se non esistono

percorsi di specializzazione la matrice sarà composta da una sola riga e tante colonne quante sono le slide presenti al suo interno, la prima collocata nella posizione [0][0]. Ogni riga aggiuntiva indicherà quindi un percorso di specializzazione, che parte dalla slide il cui codice identificativo è inserito nella posizione [riga][0]

- currentChkPnt : int
Il checkpoint nel quale l'utente sta lavorando

- currentStep : int
Il punto del percorso di specializzazione nel quale l'utente sta lavorando

- hashSlidePos : Hash
Lista di codici identificativi delle slide di cui è composto il percorso di presentazione. Sono associati a due integer: *row* indica la riga in cui si trova la slide all'interno della matrice path, mentre *col* indica la colonna

- checkPointHash : Hash
Lista delle slide che fungono da checkpoint. È una lista di valori { `id_slide` : `riga nella matrice del percorso` }

Metodi

- findChkPntRow(checkPointSlide : String) : int
Trova la posizione corrispondente della slide che funge da checkpoint all'interno della matrice di presentazione.

Argomenti

- **checkPointSlide : String** Codice identificativo della slide

Note

- Utilizza checkPointHash per restituire il valore. Non è quindi necessario consultare la matrice path
- Se la slide non è stata trovata, e non è quindi un checkpoint, restituisce `-1`

- findSlideInPath(Slide : String, path : Array) : Collection
Trova le coordinate della slide all'interno di una matrice di codici identificativi. Restituisce un hash composto da: `row`: riga in cui si trova la slide, `col`: colonna in cui si trova la slide, `chkRow`: riga in cui si trova la slide se funge anche da checkpoint

Argomenti

- **Slide : String** Codice identificativo della slide
- **path : Array** La matrice in cui cercare la slide

Note

- row, col, chkRow vanno inizializzati a `-1` e restituiti con questo valore se la slide non viene trovata

- insertSlide(lvl : int, pos : int, what : String) : Trail

Inserisce il codice identificativo della slide in path, nella posizione indicata dalle coordinate inviate. Restituisce un riferimento al this per consentire chiamate consecutive ai metodi della classe

Argomenti

- **lvl : int** Indica la riga in cui si intende inserire la slide
- **pos : int** Indica la colonna in cui si intende inserire la slide
- **what : String** È il codice identificativo della slide

- removeSlide(idSlide : String) : Trail

Rimuove una slide dal percorso di presentazione. Restituisce un riferimento al this per consentire chiamate consecutive ai metodi della classe

Argomenti

- **idSlide : String** È il codice identificativo della slide

Note

- se la slide da rimuovere è quella nella posizione [0][0], allora l'intero percorso andrà rimosso e hashSlidePos e checkPointHash azzerati
- la slide va rimossa da hashSlidePos, e da checkPointHash se fungeva da checkpoint (come vanno rimosse anche le slide incluse nel suo percorso di specializzazione, e se anche qualcuna di queste slide fungeva da checkpoint andranno rimosse ricorsivamente anche le altre slide che dipendevano da essa, etc)

- makeCheckPoint(idSlide : String) : Trail

Imposta la slide come checkpoint. Restituisce un riferimento al this per consentire chiamate consecutive ai metodi della classe

Argomenti

- **idSlide : String** È il codice identificativo della slide

- removeCheckPoint(idSlide : String) : Trail

Rimuove il percorso di specializzazione associato alla slide. Restituisce un riferimento al this per consentire chiamate consecutive ai metodi della classe

Argomenti

- **idSlide : String** È il codice identificativo della slide

- findSlideIndex(IdSlide : String) : int

Restituisce la posizione della slide all'interno della lista di tutte le slide create dall'utente nella presentazione in cui sta lavorando.

Argomenti

- **idSlide : String** È il codice identificativo della slide

+ initPath(Slides : Collection, inPath : Array) : Trail

Data una lista di slides ed una matrice rettangolare, inizializza i tre hash slidesIndex, hashSlidePos e checkPointHash. Restituisce un riferimento al this per consentire chiamate consecutive ai metodi della classe

Argomenti

- **Slides : Collection** È una collezione di tutte le slides create dall'utente nella presentazione in cui sta lavorando
- **inPath : Array** È la matrice rettangolare che rappresenta il percorso della presentazione

+ nextSlide() : int

Restituisce la posizione (all'interno della lista di tutte le slides create dall'utente nella presentazione corrente) della prossima slide rispetto alla posizione in cui l'utente si trova nel percorso che sta visualizzando.

+ prevSlide() : int

Restituisce la posizione (all'interno della lista di tutte le slides create dall'utente nella presentazione corrente) della slide precedente rispetto alla posizione in cui l'utente si trova nel percorso che sta visualizzando.

+ enterInCheckPoint() : int

Restituisce la riga in cui si trova il percorso di specializzazione della slide che l'utente sta visualizzando, solo se la slide funge da checkpoint

Note

- restituisce -1 se la slide non è un checkpoint o se è la slide nella posizione [0][0]

+ returnToCheckPoint() : int

Ritorna al percorso in cui l'utente si trovava prima di di accedere ad un percorso di specializzazione

Note

- restituisce -1 se l'utente non si trova in un percorso di specializzazione

+ goToSlide(idSlide : String) : int

Sposta la visualizzazione sulla slide ricevuta, se presente all'interno del percorso, e restituisce la posizione della slide all'interno di slidesIndex

Argomenti

- **idSlide : String** È il codice identificativo della slide

+ isCurrentSlideChkPnt() : boolean

Restituisce *true* se la slide in cui l'utente si trova è un checkpoint, *false* altrimenti

+ isCheckPoint(idSlide : String) : boolean

Restituisce *true* se la slide corrispondente al codice identificativo rievuto è un checkpoint, *false* altrimenti

Argomenti

- **idSlide : String** È il codice identificativo della slide

+ isSlideInTrail(idSlide : String) : boolean

Restituisce *true* se la slide corrispondente al codice identificativo rievuto è presente all'interno del percorso, *false* altrimenti

Argomenti

- **idSlide : String** È il codice identificativo della slide

+ getCurrentIndex() : int

Restituisce la posizione in slidesIndex della slide attualmente selezionata

+ getCurrentId() : String

Restituisce il codice identificativo della slide attualmente selezionata

+ insertSlideAfterCurrent(idSlide : String) : boolean

Inserisce la slide ricevuta nella posizione successiva a quella attualmente selezionata. Restituisce *true* se l'operazione ha avuto successo, *false* altrimenti

Argomenti

- **idSlide : String** È il codice identificativo della slide

+ insertSlideAfterCurrent(idSlide : String) : boolean

Inserisce la slide ricevuta nella posizione precedente a quella attualmente selezionata. Restituisce *true* se l'operazione ha avuto successo, *false* altrimenti

Argomenti

- **idSlide : String** È il codice identificativo della slide

+ insertSlideAfterCurrent(idSlide : String) : boolean

Inserisce la slide ricevuta nel percorso di specializzazione correlato alla slide attualmente selezionata. Se la slide non era un checkpoint viene impostata come tale. Restituisce *true* se l'operazione ha avuto successo, *false* altrimenti

Argomenti

- **idSlide : String** È il codice identificativo della slide

+ removeCurrentSlde() : boolean
Rimuove la slide attualmente selezionata. Restituisce *true* se l'operazione ha avuto successo, *false* altrimenti

+ removeCheckPoint() : void
Toglie il percorso di specializzazione dalla slide attualmente selezionata, che non sarà più un checkpoint

+ getTrail() : Array
Restituisce l'array path

+ isTrailEmpty() : Boolean
Restituisce *true* se il percorso è vuoto, *false* altrimenti

3.7.4 premi/client/presentation/lib/signalCtrl

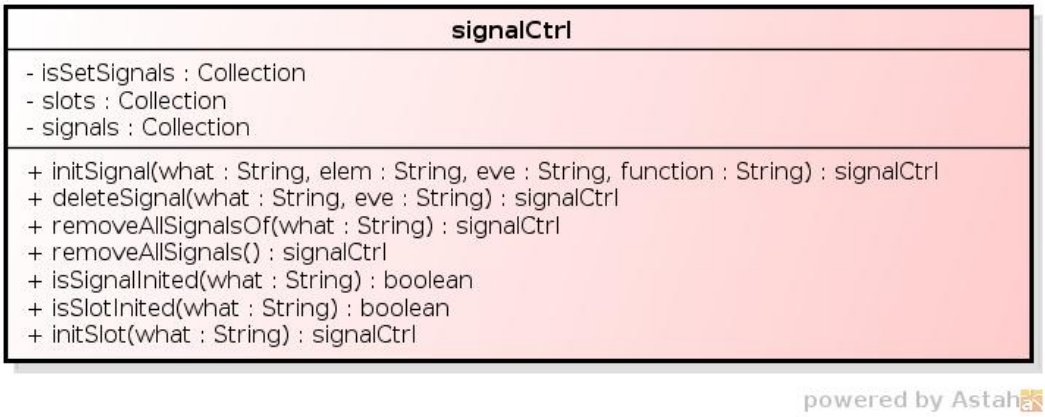

Figura 9: Diagramma della classe premi/client/presentation/lib/signalCtrl

Descrizione

Questa classe ha lo scopo di registrare i signal quando un certo slot viene creato in modo tale che facendo un controllo sul relativo controller si eviti di aggiungerne in più. Inoltre potrà registrare i listner associati a qualche elemento del DOM (di solito il document stesso) in modo tale da rimuovere/aggiungere dinamicamente i listner giusti a seconda dello stato in cui si trova l'applicazione.

Attributi

- isSetSignals : Collection
Oggetto JSON che indica per ogni campo che contiene se è presente almeno un Signal. I campi che contiene sono:

- *infographic:* identifica l'oggetto infographic;
- *trails:* identifica gli oggetti trail;
- *viewer:* identifica gli oggetti viewer.

- slots : Collection

Oggetto JSON che indica per ogni campo che contiene se è stato creato un certo slots. I campi che contiene sono:

- *infographic:* identifica l'oggetto infographic;
- *trails:* identifica gli oggetti trail;
- *viewer:* identifica gli oggetti viewer.

- signals : Collection

Oggetto JSON che indica per ogni campo quali sono i signal presenti:

- *infographic:* identifica i signal dell'oggetto infographic;
- *trails:* identifica i signal degli oggetti trail;
- *viewer:* identifica i signal degli oggetti viewer.

Metodi

+ initSignal(what : String, elem : String, eve : String, func : String) : signalCtrl

inizializza i signal di un oggetto.

Argomenti

- **what : String** identifica l'oggetto (infographic, trails, viewer);
- **elem : String** identifica l'elemento su cui fa riferimento un signal;
- **eve : String** rappresenta l'evento;
- **func : String** rappresenta la funzione.

+ deleteSignal(what : String, eve : String) : signalCtrl

rimuove i signal che si riferiscono ad un certo evento.

Argomenti

- **what : String** identifica l'oggetto (infographic, trails, viewer) su cui eliminare i signal;
- **eve : String** identifica l'evento.

+ removeAllSignalOf(what : String) : signalCtrl

rimuove tutti i signal di un determinato oggetto.

Argomenti

- **what : String** identifica l'oggetto (infographic, trails, viewer);

+ removeAllSignals() : signalCtrl
rimuove tutti i signal di tutti gli oggetti.

+ isSignalInited(what : String) : boolean
restituisce true se è impostato almeno un signal di un determinato oggetto.

Argomenti

- **what : String** identifica l'oggetto (infographic, trails, viewer) su cui controllare i signal;

+ initSlot(what : String) : signalCtrl
inizializza gli slot di un determinato oggetto.

Argomenti

- **what : String** identifica l'oggetto (infographic, trails, viewer);

+ isSlotInited(what : String) : boolean
restituisce true se è stato inizializzato almeno uno slot su un determinato oggetto.

Argomenti

- **what : String** identifica l'oggetto (infographic, trails, viewer) su cui controllare gli slot.

+ initSlot(what : String) : signalCtrl
inizializza uno slot su un determinato oggetto.

Argomenti

- **what : String** identifica l'oggetto (infographic, trails, viewer);

3.8 premi/client/presentationManager

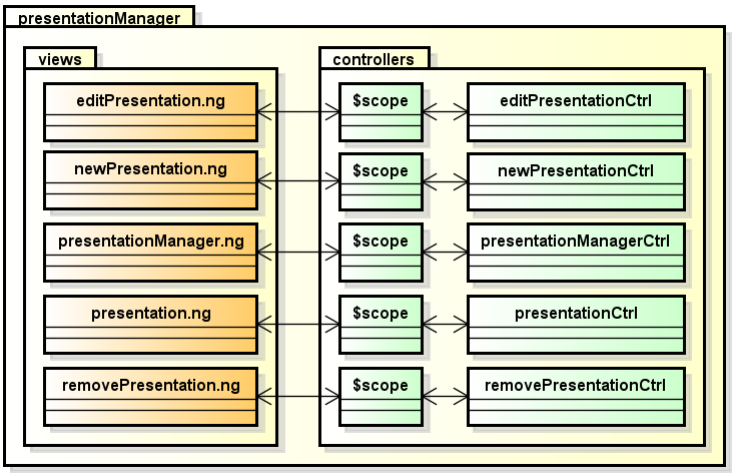

Figura 10: Diagramma del package premi/client/presentationManager

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

3.8.2 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

3.8.3 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

3.8.4 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 sua la modifica e rimozione

3.8.5 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

3.8.6 premi/client/presentationManager/controllers/editPresentationCtrl

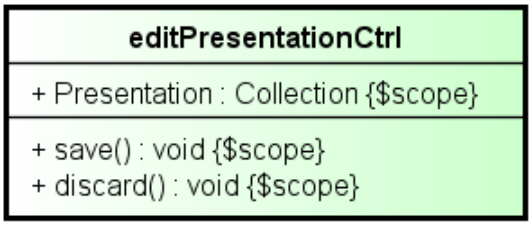

Figura 11: Diagramma della classe premi/client/presentationManager/controllers/editPresentationCtrl

Descrizione

Controller della view *editPresentation.ng*. Permette all'utente di modificare titolo e descrizione di una presentazione.
La dicitura {$scope} nel diagramma UML$_G$ indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

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*

3.8.7 premi/client/presentationManager/controllers/newPresentationCtrl

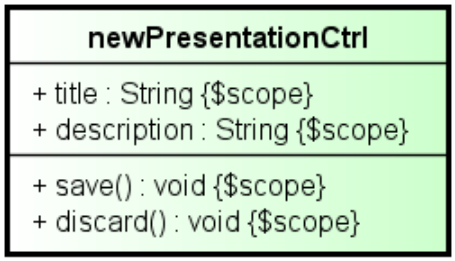

Figura 12: Diagramma della classe premi/client/presentationManager/controllers/-newPresentationCtrl

Descrizione

Controller della view *newPresentation.ng*. Permette all'utente di creare una nuova presentazione e di salvarla nel database.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

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*

3.8.8 premi/client/presentationManager/controllers/presentationManagerCtrl

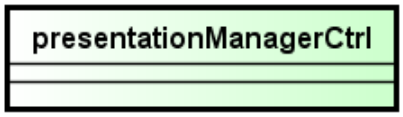

Figura 13: Diagramma della classe premi/client/presentationManager/controllers/-presentationManagerCtrl

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.

3.8.9 premi/client/presentationManager/controllers/presentationsCtrl

Figura 14: 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
La dicitura {$scope} nel diagramma UML$_G$ indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Attributi

+ Presentations : Collection
Collezione MongoDB di presentazioni. Viene inizializzata dal controller prelevando le presentazioni pubblicate al client.

3.8.10 premi/client/presentationManager/controllers/removePresentationCtrl

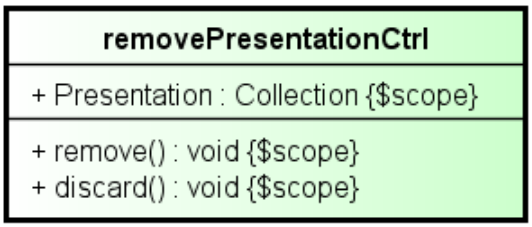

Figura 15: Diagramma della classe premi/client/presentationManager/controllers/removePresentationCtrl

Descrizione
Controller della view *removePresentation.ng*. Permette all'utente di eliminare una presentazione da lui creata in precedenza.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

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*

3.9 premi/client/editor

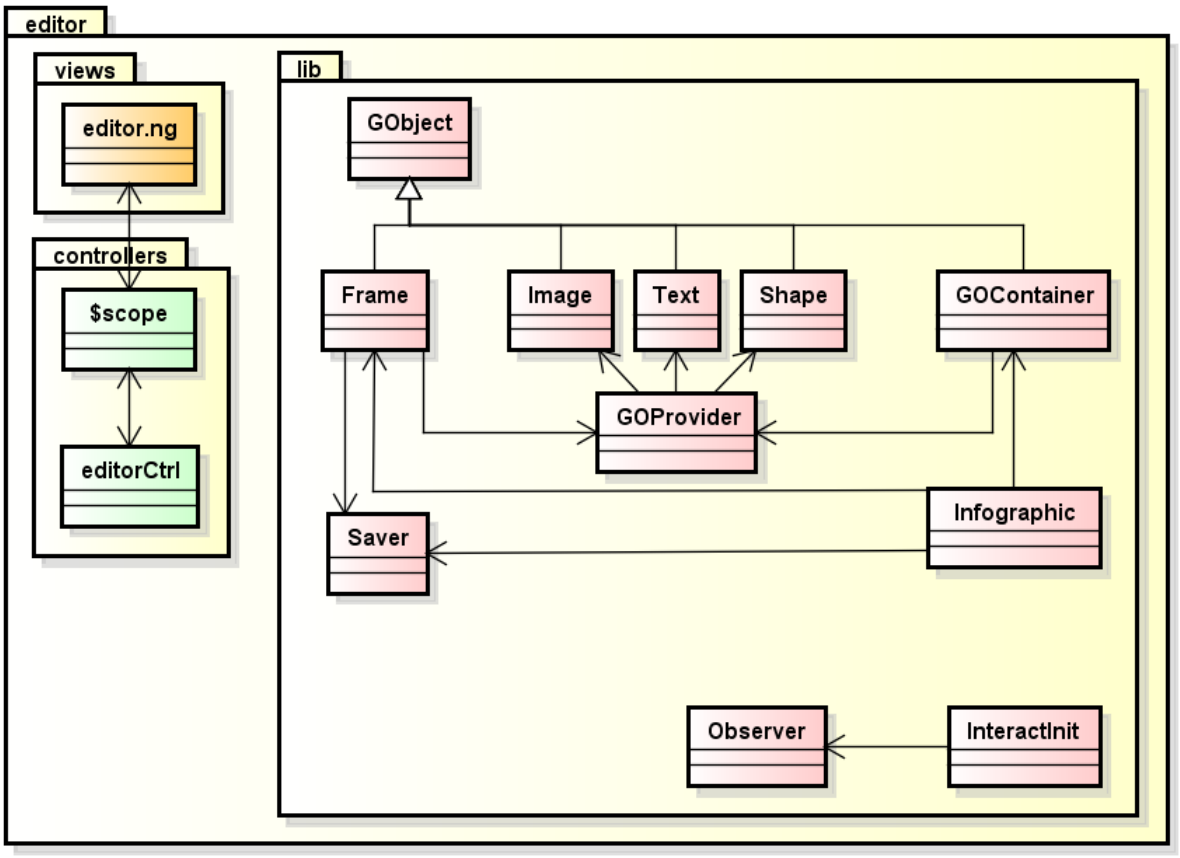

Figura 16: Diagramma della classe premi/client/editor

3.9.1 premi/client/editor/views/editor.ng

Descrizione

Template della vista associata allo *$scope* di *editorCtrl*. Fornisce uno scheletro per le altre viste dedicate alla gestione dell'editor.

3.9.2 premi/client/editor/controllers/editorCtrl

Figura 17: Diagramma della classe premi/client/editor/controllers/editorCtrl

Descrizione

Questo controller non è al momento provvisto di funzionalità. Si appoggia alla vista associata *editor.ng*, la quale funge da scheletro per le viste necessarie alla gestione dei vari editor a disposizione dell'utente.

3.9.3 premi/client/editor/lib/GObject

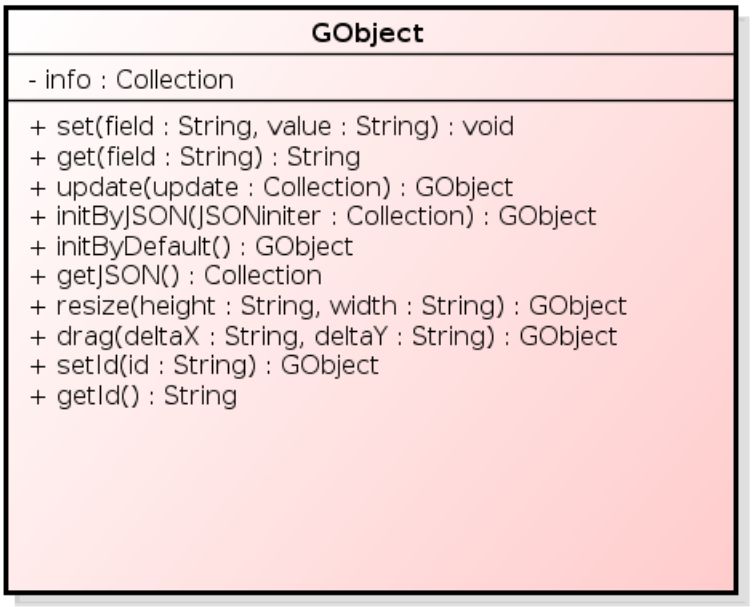

Figura 18: Diagramma della classe premi/client/editor/lib/GObject

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) : GObject

permette di settare un campo dell'attributo info. Restituisce un riferimento di GObject.

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. Restituisce un valore del campo info richiesto.

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. Restituisce un riferimento di GObject.

Argomenti

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

+ initByJSON(JSONiniter : Collection) : Gobject

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

Argomenti

- **JSONiniter : 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. Restituisce un riferimento di GObject.

+ getJSON() : String

restituisce la collection dell'attributo info.

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

permette di settare l'altezza e la larghezza del'oggetto. Restituisce un riferimento di GObject.

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. Restituisce un riferimento di GObject.

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. Restituisce un riferimento di GObject.

Argomenti

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

+ getId() : String

restituisce l'id dell'oggetto.

3.9.4 premi/client/editor/lib/GOProvider

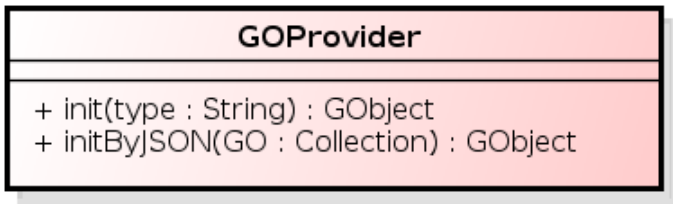

Figura 19: Diagramma della classe premi/client/editor/lib/GOProvider

Descrizione

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

Dipendenze

- **Image**: per inizializzare le immagini;
- **Shape**: per inizializzare gli shape;
- **Text**: per inizializzare i testi.

Metodi

+ init(type : String) : Collection
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) : Collection
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.

3.9.5 premi/client/editor/lib/Frame

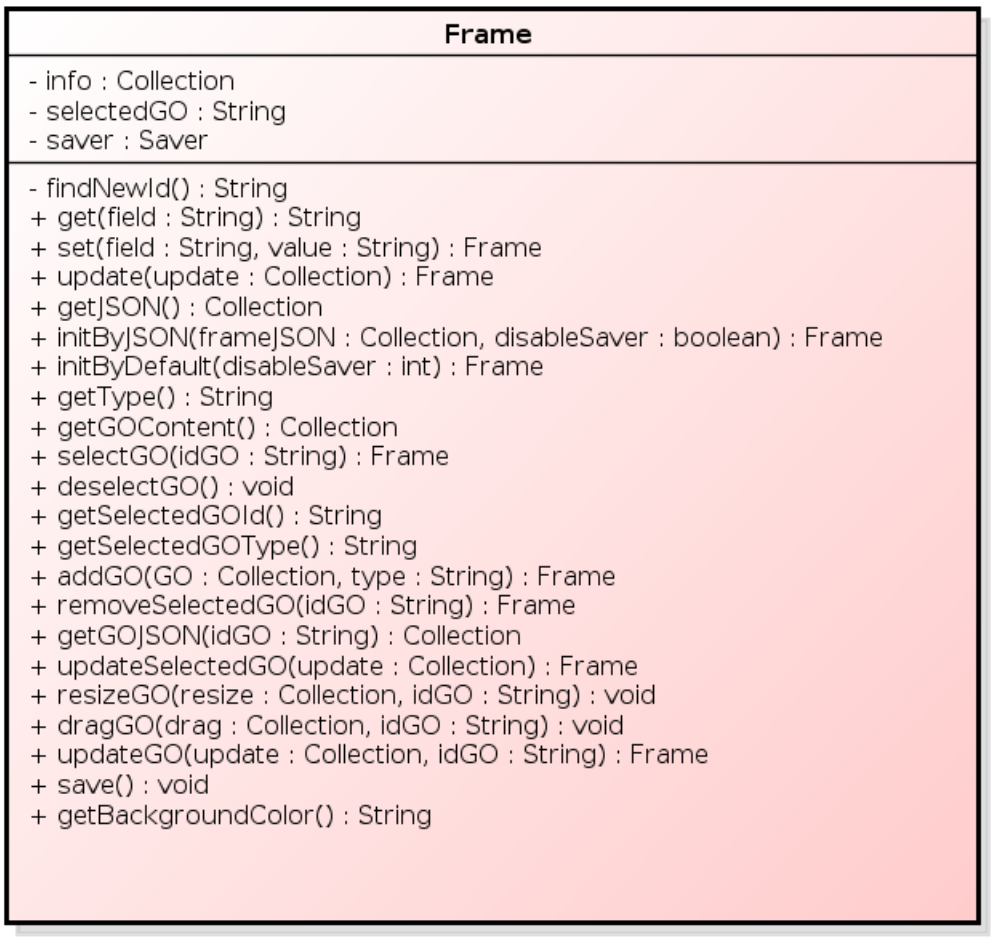

Figura 20: Diagramma della classe premi/client/editor/lib/Frame

Descrizione

frame è una classe che rappresenta un frame di una presentazione. E' un oggetto che può essere rappresentato nella presentazione. Per inizializzare gli oggetti image, shape, text si utilizzano i metodi di GOProvider *initByJSON(GO)* e *init(type)*

Classi ereditate

- GObject

Dipendenze

- **GOProvider**: per inizializzare gli oggetti image, shape e text;
- **Saver**: per effettuare le modifiche del frame nel database.

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 : Collection

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.

+ 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) : Frame

permette di aggiornare i campi dell'attributo info. Restituisce un riferimento di Frame.

Argomenti

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

+ initByJSON(frameJSON : Collection, disableSaver: boolean) : Frame

permette di inizializzare l'attributo info tramite un oggetto JSON. Se disableSaver è uguale a false viene utilizzato il metodo di Saver *setContainer(id,type)* e *init()* per impostare il frame come contenitore per il salvataggio dei dati sul database. Restituisce un riferimento di Frame.

Argomenti

- **frameJSON : Collection** è un oggetto JSON che contiene chiave e valore di inizializzazione dei campi dell'attributo info.
- **disableSaver : boolean** se è uguale a false il contenuto di frame non viene salvato.

+ initByDefault(disableSaver) : Frame

permette di inizializzare i campi dell'attributo info con i parametri di default. Se disableSaver è uguale a false viene utilizzato il metodo di Saver *setContainer(id,type)* e *init()* per impostare il frame come contenitore per il salvataggio dei dati sul database. Restituisce un riferimento di Frame.

Argomenti

- **disableSaver : boolean** se è uguale a false il contenuto di frame non viene salvato.

+ getJSON() : String

restituisce la collection dell'attributo info.

+ getType() : String

restituisce la stringa frame per indicare che il tipo dell'oggetto è frame.

+ getGOContent() : Collection

restituisce il contenuto della collezione content che è un insieme di oggetti JSON che fanno parte del frame.

+ selectGO(idGO) : Collection
restituisce l'oggetto grafico con id = idGO contenuto all'interno del frame.

Argomenti

- **idGO : String** valore dell'id dell'oggetto grafico da restituire.

+ deselectGO() : void
deseleziona l'oggetto grafico portando a null l'attributo selectedGo.

+ getSelectedGOId() : String
restituisce l'id dell'oggetto grafico selezionato.

+ getSelectedGOType() : String
restituisce il tipo dell'oggetto grafico selezionato.

+ getSelectedGO() : String
restituisce l'oggetto grafico selezionato.

+ addGO(GO : Collection, type : String) : Frame
aggiunge un oggetto di tipo type al frame e restituisce il riferimento del frame. Per salvare sul database viene usato il metodo di Saver *insert(GOJSON)* che permette di appendere un operazione di inserimento nell'oggetto Saver. Restituisce un riferimento di Frame.

Argomenti

- **GO : Collection** un oggetto JSON che serve per inizializzare l'oggetto da aggiungere al frame;
- **type : String** contiene il tipo dell'oggetto da inserire nel frame.

+ removeSelectedGO(idGO : String) : Frame
se l'oggetto con id = idGO è selezionato lo elimina. Viene utilizzato il metodo *remove(idGO,type)* che permette di appendere un operazione di rimozione nell'oggetto Saver. Restituisce un riferimento di Frame.

Argomenti

- **idGO : Collection** rappresenta l'id dell'oggetto da eliminare.

+ getGOJSON(idGO : String) : Collection
restituisce l'oggetto JSON con id= idGO appartenente al frame.

Argomenti

- **idGO : String** contiene l'id dell'oggetto che dev'essere restituito.

+ updateSelectedGO(update : Collection) : Collection
aggiorna i campi dell'oggetto selezionato.

Argomenti

- **update : Collection** è un oggetto JSON che contiene chiave e valore dei campi da aggiornare dell'oggetto selezionato. Viene utilizzato il metodo *update(idGO,type,update)* per appendere un operazione di modifica nell'oggetto Saver.

+ resizeGO(resize : Collection, idGO : String) : void

aggiorna i campi dell'oggetto con un determinato id, per permettere il ridimensionamento dell'oggetto. Viene utilizzato il metodo *update(idGO,type,update)* per appendere le operazioni di modifica nell'oggetto Saver.

Argomenti

- **resize : Collection** è un oggetto JSON che contiene chiave e valore dei campi height, width, dataX e dataY per permettere di ridimensionare l'oggetto appartenente al frame;
- **idGO : String** contiene l'id dell'oggetto da ridimensionare.

+ dragGO(drag : Collection, idGO : String) : void

aggiorna i campi dell'oggetto con un determinato id, per permettere lo spostamento dell'oggetto nel template grafico. Viene utilizzato il metodo *update(idGO,type,update)* per appendere le operazioni di modifica nell'oggetto Saver.

Argomenti

- **drag : Collection** è un oggetto JSON che contiene chiave e valore dei campi dataX e dataY per permettere lo spostamento dell'oggetto appartenente al frame;
- **idGO : String** contiene l'id dell'oggetto da spostare.

+ updateGO(update : Collection, idGO : String) : void

aggiorna i campi dell'oggetto con un determinato id appartenente al frame. Viene utilizzato il metodo *update(idGO,type,update)* per appendere un operazione di modifica nell'oggetto Saver.

Argomenti

- **update : Collection** è un oggetto JSON che contiene chiave e valore dei campi da aggiornare dell'oggetto con un determinato id;
- **idGO : String** contiene l'id dell'oggetto da aggiornare.

+ save() : void

salva le operazioni pendenti nel database. Viene utilizzato il metodo *save()* che si occupa di inserire le operazioni di inserimento, modifica, rimozione presenti nell'oggetto Saver nel database.

+ getBackgroundColor() : String

restituisce il colore in formato esadecimale dello sfondo del frame.

3.9.6 premi/client/editor/lib/GOContainer

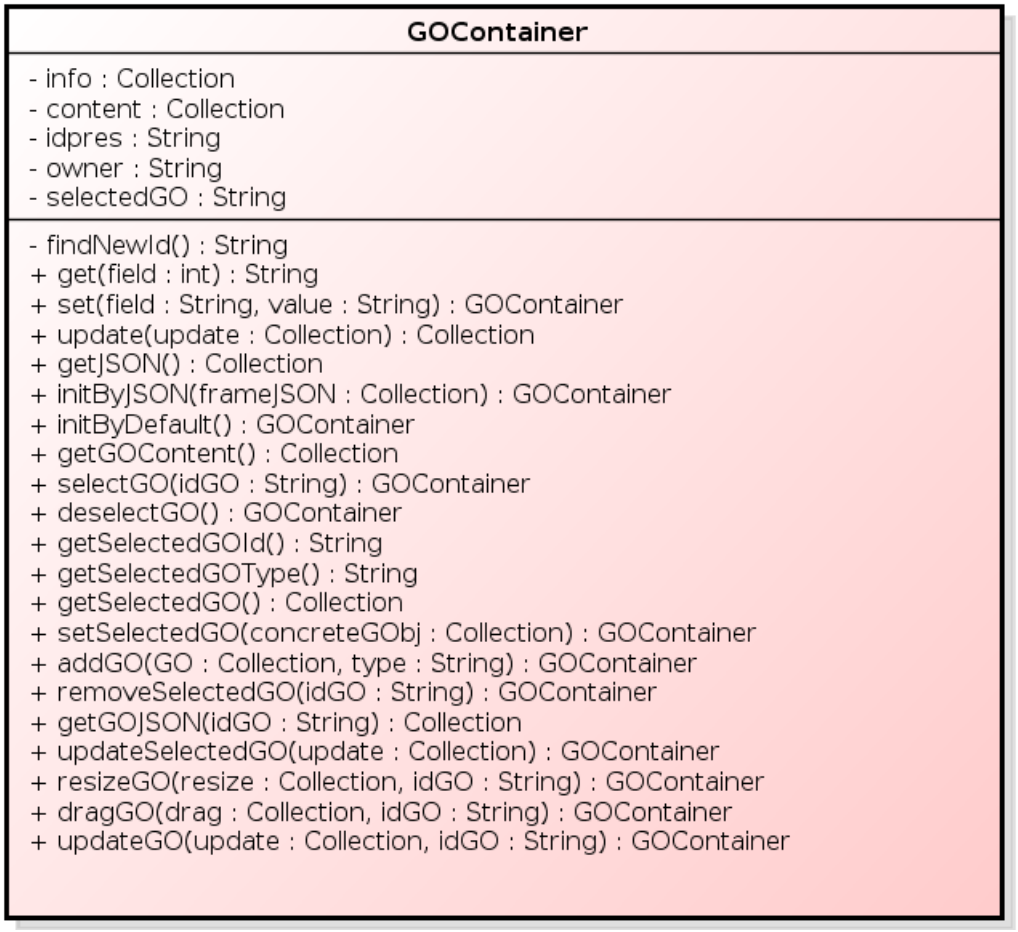

Figura 21: Diagramma della classe premi/client/editor/lib/GOContainer

Descrizione

è una classe che rappresenta il contenitore degli oggetti che possono essere inseriti in un frame. Per inizializzare gli oggetti image, shape, text si utilizzano i metodi di GOProvider *initByJSON(GO)* e *init(type)*.

Classi ereditate

- GObject

Dipendenze

- **GOProvider**: per inizializzare gli oggetti image, shape e text.

Attributi

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

- *background:* è un oggetto JSON che contiene i seguenti campi:
  - *image:* definisce il percorso dell'immagine di background;
  - *size:* definisce la grandezza dell'immagine di background;
  - *color:* definisce il colore di background;
  - *repeat:* definisce il metodo di ripetizione dello sfondo di background;
  - *type:* definisce il nome del tipo dell'oggetto.
- *content:* è un oggetto JSON che contiene gli oggetti che fanno parte del frame;
- *idpres:* identifica l'id della presentazione a cui si riferisce;
- *owner:* identifica l'id dell'utente che ha creato la presentazione.

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

Metodi

- findNewId() : String
trova un nuovo id valido per il frame corrente.

+ 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) : GOContainer
permette di aggiornare i campi dell'attributo info. Restituisce un riferimento di GOContainer.

Argomenti

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

+ initByJSON(frameJSON : Collection, disableSaver: boolean) : frame

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

Argomenti

- **frameJSON : Collection** è un oggetto JSON che contiene chiave e valore di inizializzazione dei campi dell'attributo info.
- **disableSaver : boolean** se è uguale a false il contenuto di frame non viene salvato.

+ initByDefault(disableSaver) : GOContainer

permette di inizializzare i campi dell'attributo info con i parametri di default. Restituisce un riferimento di GOContainer.

Argomenti

- **disableSaver : boolean** se è uguale a false il contenuto di frame non viene salvato.

+ getGOContent() : Collection

restituisce il contenuto della collezione content che è un insieme di oggetti JSON che fanno parte del frame.

+ selectGO(idGO) : GObject

restituisce l'oggetto grafico con id = idGO contenuto all'interno del GOContainer.

Argomenti

- **idGO : String** valore dell'id dell'oggetto grafico da restituire.

+ deselectGO() : void

deseleziona l'oggetto grafico portando a null l'attributo selectedGo.

+ getSelectedGOId() : String

restituisce l'id dell'oggetto grafico selezionato.

+ getSelectedGOType() : String

restituisce il tipo dell'oggetto grafico selezionato.

+ getSelectedGO() : String

restituisce l'oggetto grafico selezionato.

+ setSelectedGO(concreteGObj : Collection) : frame

setta l'oggetto concreteGObj come oggetto selezionato.

Argomenti

- **concreteGObj : Collection** oggetto GObject.

+ addGO(GO : Collection, type : String) : GOContainer

aggiunge un oggetto di tipo type al GOContainer e restituisce il riferimento del GOContainer.

Argomenti

- **GO : Collection** un oggetto JSON che serve per inizializzare l'oggetto da aggiungere al GOContainer;
- **type : String** contiene il tipo dell'oggetto da inserire nel GOContainer.

+ removeSelectedGO(idGO : String) : GOContainer

se l'oggetto con id = idGO è selezionato lo elimina. Restituisce un riferimento di GOContainer.

Argomenti

- **idGO : Collection** rappresenta l'id dell'oggetto da eliminare.

+ getGOJSON(idGO : String) : Collection

restituisce l'oggetto JSON con id= idGO appartenente al GOContainer.

Argomenti

- **idGO : String** contiene l'id dell'oggetto che dev'essere restituito.

+ updateSelectedGO(update : Collection) : Collection

aggiorna i campi dell'oggetto selezionato.

Argomenti

- **update : Collection** è un oggetto JSON che contiene chiave e valore dei campi da aggiornare dell'oggetto selezionato.

+ resizeGO(resize : Collection, idGO : String) : void

aggiorna i campi dell'oggetto con un determinato id, per permettere il ridimensionamento dell'oggetto.

Argomenti

- **resize : Collection** è un oggetto JSON che contiene chiave e valore dei campi height, width, dataX e dataY per permettere di ridimensionare l'oggetto appartenente al GOContainer;
- **idGO : String** contiene l'id dell'oggetto da ridimensionare.

+ dragGO(drag : Collection, idGO : String) : void

aggiorna i campi dell'oggetto con un determinato id, per permettere lo spostamento dell'oggetto nel template grafico.

Argomenti

- **drag : Collection** è un oggetto JSON che contiene chiave e valore dei campi dataX e dataY per permettere lo spostamento dell'oggetto appartenente al GOContainer;
- **idGO : String** contiene l'id dell'oggetto da spostare.

+ updateGO(update : Collection, idGO : String) : void

aggiorna i campi dell'oggetto con un determinato id appartenente al GOContainer.

Argomenti

- **update : Collection** è un oggetto JSON che contiene chiave e valore dei campi da aggiornare dell'oggetto con un determinato id;
- **idGO : String** contiene l'id dell'oggetto da aggiornare.

3.9.7 premi/client/editor/lib/Image

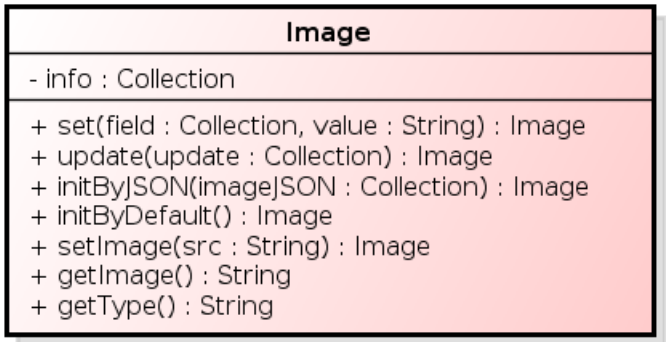

Figura 22: Diagramma della classe premi/client/editor/lib/Image

Descrizione

è una classe che rappresenta un oggetto immagine. Contiene i metodi per gestire un immagine.

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:

- *src:* definisce il percorso dell'immagine;
- *type:* identifica il tipo di oggetto.

- selectedGO : Collection
contiene l'oggetto GObject contenuto nel frame corrente selezionato dall'utente.

Metodi

+ set(field : String, value : String) : void
permette di settare un campo dell'attributo info.

Argomenti

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

+ update(update : Collection) : Image
permette di aggiornare i campi dell'attributo info. Restituisce un riferimento di Image.

Argomenti

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

+ initByJSON(imageJSON : Collection) : Image
permette di inizializzare l'attributo info tramite un oggetto JSON. Restituisce un riferimento di Image.

Argomenti

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

+ initByDefault() : Image
permette di inizializzare i campi dell'attributo info con i parametri di default. Restituisce un riferimento di Image.

+ setImage(src : String) : Image
setta l'url(percorso) dell'immagine. Restituisce un riferimento di Image.

Argomenti

- **src : String** identifica l'url(percorso) dell'immagine.

+ getImage() : String
restituisce l'url(percorso) dell'immagine.

+ getType() : String
restituisce la stringa image per identificare che il tipo dell'oggetto è image.

3.9.8 premi/client/editor/lib/Infographic

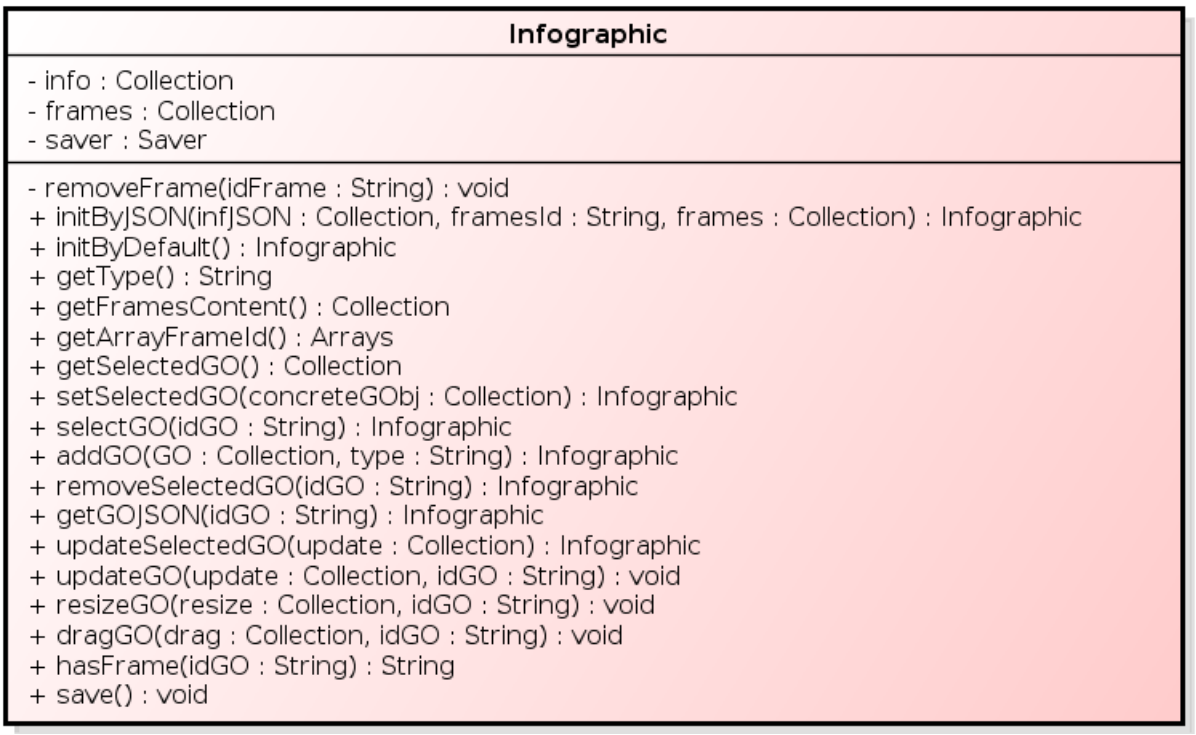

Figura 23: Diagramma della classe premi/client/editor/lib/infographic

Descrizione

è una classe che rappresenta un oggetto infografica. Un infografica contiene i frame e gli oggetti che si vogliono visualizzare nella presentazione.

Classi ereditate

- GOContainer

Dipendenze

- **Frame**: per gestire i frame nell'infografica;
- **Saver**: per apportare le modifiche sul database.

Attributi

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

- *framesId:* array che contiene gli id dei frame appartenenti all'infografica;
- *type:* identifica il tipo di oggetto ovvero infografica.

- frames : Collection
contiene degli oggetti JSON che rappresentano i frame appartenenti all'infografica.

- saver : Saver
oggetto Saver che si occupa delle modifiche e dei salvataggi su database.

Metodi

- removeFrame(idFrame : String) : void
rimuove un frame dall'infografica

Argomenti

- **idFrame : String** Identifica l'id del frame da rimuovere dall'infografica.

+ initByJSON(infJSON : Collection,framesId : String[],frames : Collection) : Infographic
permette di inizializzare l'oggetto infografica. Viene utilizzato il metodo *setContainer(id,type)* per selezionare l'oggetto infographic come contenitore dell'oggetto Saver. Restituisce un riferimento di Infographic.

Argomenti

- **infJSON : Collection** è un oggetto JSON che contiene chiave e valore di inizializzazione dei campi dell'attributo info;
- **framesId : String[]** array di identificativi che rappresenta gli id dei frame da aggiungere all'infografica;
- **frames : Collection** contiene oggetti JSON che identificano i frame appartenenti all'infografica.

+ initByDefault() : Infographic
permette di inizializzare i campi dell'attributo info con i parametri di default e di lasciare vuoti l'attributo framesId e frames. Viene utilizzato il metodo *setContainer(id,type)* per selezionare l'oggetto infographic come contenitore dell'oggetto Saver. Restituisce un riferimento di Infographic.

+ getType() : String
restituisce la stringa infographic perchè il tipo di oggetto è un infografica.

+ getFramesContent() : Collection
restituisce gli oggetti JSON che rappresentano i frame appartenenti all'infografica.

+ getArrayFrameId() : Arrays
restituisce un array in cui ciascun elemento identifica un frame appartenente all'infografica.

+ getSelectedGO() : String
restituisce l'oggetto selezionato nell'infografica.

+ setSelectedGO(concreteGObj : Collection) : Infographic
imposta su selezionato un oggetto dell'infografica. Restituisce un riferimento di Infographic.

Argomenti

- **concreteGObj : Collection** rappresenta l'oggetto JSON da selezionare nell'infografica.

+ selectedGO(idGO : String) : Infographic
seleziona un oggetto dell'infografica. Restituisce un riferimento di Infographic.

Argomenti

- **idGO : String** identifica l'id dell'oggetto da selezionare.

+ addGO(GO : Collection, type : String) : Infographic
aggiunge un oggetto di tipo type all'infografica e restituisce il riferimento dell'infografica. Viene utilizzato il metodo *insert(GOJSON)* di Saver per appendere un operazione di inserimento nell'oggetto saver. Restituisce un riferimento di Infographic.

Argomenti

- **GO : Collection** un oggetto JSON che serve per inizializzare l'oggetto da aggiungere all'infografica;
- **type : String** contiene il tipo dell'oggetto da inserire nell'infografica.

+ removeSelectedGO(idGO : String) : Infographic
se l'oggetto con id = idGO è selezionato lo elimina. Viene utilizzato il metodo *remove(idGO,type)* di Saver per appendere un operazione di rimozione nell'oggetto saver. Restituisce un riferimento di Infographic.

Argomenti

- **idGO : Collection** rappresenta l'id dell'oggetto da eliminare.

+ getGOJSON(idGO : String) : Collection

restituisce l'oggetto JSON con id= idGO appartenente all'infografica.

Argomenti

- **idGO : String** contiene l'id dell'oggetto che dev'essere restituito.

+ updateSelectedGO(update : Collection) : Infographic

aggiorna i campi dell'oggetto selezionato. Viene utilizzato il metodo *update(idGO,type,update)* di Saver per appendere un operazione di aggiornamento nell'oggetto saver. Restituisce un riferimento di Infographic.

Argomenti

- **update : Collection** è un oggetto JSON che contiene chiave e valore dei campi da aggiornare dell'oggetto selezionato.

+ updateGO(update : Collection, idGO : String) : Infographic

aggiorna i campi di un oggetto dell'infografica. Viene utilizzato il metodo *update(idGO,type,update)* di Saver per appendere un operazione di aggiornamento nell'oggetto saver. Restituisce un riferimento di Infographic.

Argomenti

- **update : Collection** è un oggetto JSON che contiene chiave e valore dei campi da modificare dell'oggetto da aggiornare;
- **idGO : Collection** identifica l'id dell'oggetto da aggiornare.

+ resizeGO(resize : Collection, idGO : String) : void

aggiorna i campi dell'oggetto con un determinato id, per permettere il ridimensionamento dell'oggetto. Viene utilizzato il metodo *update(idGO,type,update)* di Saver per appendere un operazione di aggiornamento nell'oggetto saver.

Argomenti

- **resize : Collection** è un oggetto JSON che contiene chiave e valore dei campi height, width, dataX e dataY per permettere di ridimensionare l'oggetto appartenente al frame;
- **idGO : String** contiene l'id dell'oggetto da ridimensionare.

+ dragGO(drag : Collection, idGO : String) : void

aggiorna i campi dell'oggetto con un determinato id, per permettere lo spostamento dell'oggetto nel template grafico. Viene utilizzato il metodo *update(idGO,type,update)* di Saver per appendere un operazione di aggiornamento nell'oggetto saver.

Argomenti

- **drag : Collection** è un oggetto JSON che contiene chiave e valore dei campi dataX e dataY per permettere lo spostamento dell'oggetto appartenente al frame;

- **idGO : String** contiene l'id dell'oggetto da spostare.

+ hasFrame(idGO : String) : Collection

restituisce le proprietà in JSON di un oggetto dell'infografica.

Argomenti

- **idGO : String** rappresenta l'id dell'oggetto dell'infografica che si vuole restituire.

+ save() : void

esegue le operazioni pendenti di inserimento, modifica e rimozione sul database. Restituisce un riferimento dell'oggetto Saver.

3.9.9 premi/client/editor/lib/interactInit

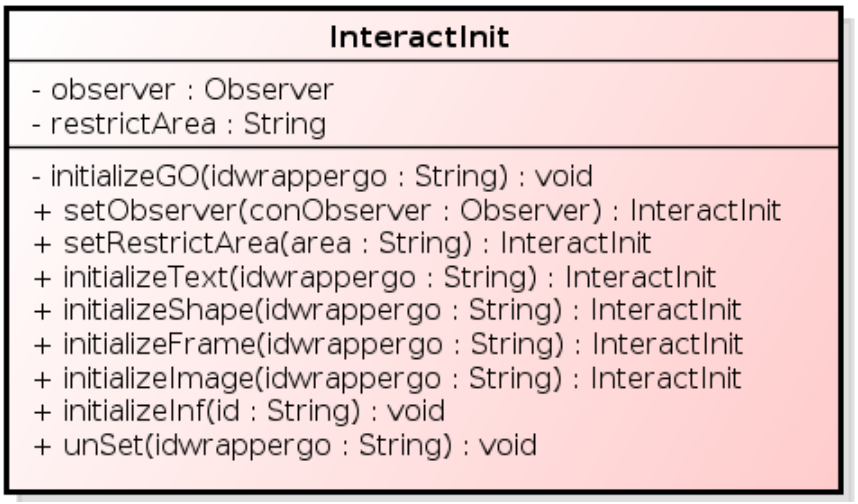

Figura 24: Diagramma della classe premi/client/editor/lib/InteractInit

Descrizione

classe che contiene i metodi per la gestione del ridimensionamento e dello spostamento degli oggetti.

Dipendenze

- **Observer**: per osservare l'oggetto inizializzato con interactjs. In molti metodi vengono usati le funzioni *on(signal,func)* e *emit(signal,param1,param2,param3,param4)* di Observer per emettere dei segnali.

Attributi

- observer : Observer

oggetto Observer che viene utilizzato per osservare un oggetto grafico;

- restrictArea : String
identifica l'area in cui un oggetto può essere spostato.

Metodi

- initializeGO(idwrapperGO : String) : void
inizializza un oggetto grafico impostando l'area di restrizione e impostando l'observer sull'oggetto. Si appoggia alla libreria interact e sulla classe observer.

Argomenti

- **idwrapperGO : String** Identifica l'id dell'oggetto da inizializzare.

+ setObserver(conObserver : Observer) : interactInit
imposta un oggetto Observer sull'oggetto da controllare. restituisce un riferimento dell'oggetto interactInit.

Argomenti

- **conObserver : Observer** Identifica l'observer da impostare sull'oggetto.

+ setRestrictArea(area : String) : interactInit
imposta l'area di restrizione entro cui un oggetto può essere spostato. Restituisce un riferimento dell'oggetto interactInit.

Argomenti

- **area : String** Identifica l'area di restrizione.

+ initializeText(idwrappergo : String) : interactInit
inizializza un determinato oggetto di tipo text. Restituisce un riferimento dell'oggetto interactInit.

Argomenti

- **idwrappergo : String** Identifica l'id dell'oggetto text da inizializzare.

+ initializeShape(idwrappergo : String) : interactInit
inizializza un determinato oggetto di tipo shape impostando il comportamento per lo spostamento e ridimensionamento dell'oggetto. Restituisce un riferimento dell'oggetto interactInit.

Argomenti

- **idwrappergo : String** Identifica l'id dell'oggetto shape da inizializzare.

+ initializeFrame(idwrappergo : String) : interactInit
inizializza un determinato oggetto di tipo frame. Restituisce un riferimento dell'oggetto interactInit.

Argomenti

- **idwrappergo : String** Identifica l'id dell'oggetto frame da inizializzare.

+ initializeImage(idwrappergo : String) : interactInit
inizializza un determinato oggetto di tipo image impostando i comportamenti per lo spostamento e il ridimensionamento dell'oggetto. Restituisce un riferimento dell'oggetto interactInit.

Argomenti

- **idwrappergo : String** Identifica l'id dell'oggetto image da inizializzare.

+ initializeInf(id : String) : interactInit
inizializza un determinato oggetto di tipo infografica. Restituisce un riferimento dell'oggetto interactInit.

Argomenti

- **idwrappergo : String** Identifica l'id dell'oggetto infografica da inizializzare.

+ unSet(idwrappergo : String) : interactInit
disabilita interactjs per un determinato oggetto togliendo la possibilità di ridimensionamento e spostamento dell'oggetto.

Argomenti

- **idwrappergo : String** Identifica l'id dell'oggetto su cui disabilitare interactjs.

3.9.10 premi/client/client/lib/Observer

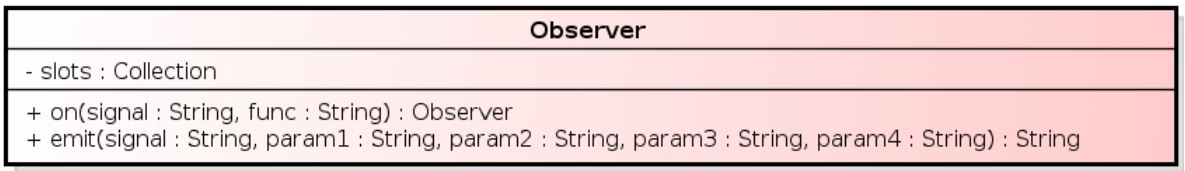

Figura 25: Diagramma della classe premi/client/editor/lib/Observer

Descrizione
è la classe che si occupa di osservare degli oggetti grafici impostando e inviando dei segnali.

Attributi

- slots : Collection
array di oggetti JSON in cui la chiave rappresenta un segnale e il valore l'azione da intraprendere.

Metodi

+ on(signal : String, func : String) : Observer
assegna la funzione func al segnale signal. Restituisce un riferimento di Observer.

Argomenti

- **signal : String** identifica il nome del segnale;
- **func : String** identifica la funzione da eseguire al verificarsi del segnale;

+ emit(signal : String, param1 : String, param2 : String, param3 : String, param4 : String) : String
Restituisce lo slot con un determinato segnale e con determinati parametri.

Argomenti

- **signal : String** identifica il nome del segnale;
- **param1,param2,param3,param4 : String** identificano i parametri del segnale signal;

3.9.11 premi/client/editor/lib/saver

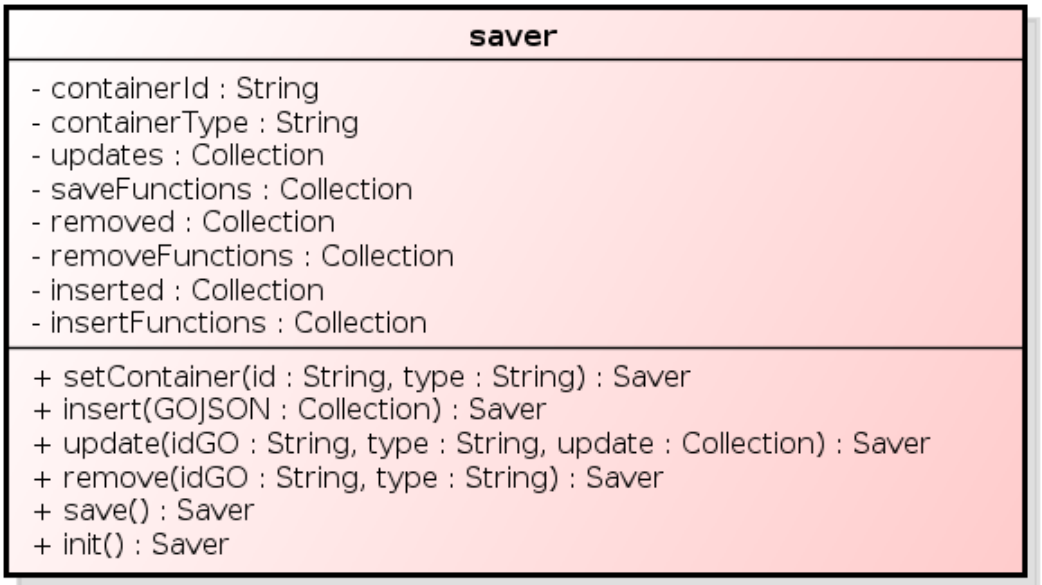

Figura 26: Diagramma della classe premi/client/editor/lib/Saver

Descrizione

rappresenta un oggetto che permette ad un contenitore di oggetti di interfacciarsi con il database ed effettuare le modifiche. L'oggetto saver riceve dal contenitore una serie di operazioni da eseguire sul database e le esegue rispettando i suoi tempi di risposta.

Attributi

- containerId : String
identifica l'id dell'oggetto da interfacciare con il database;

- containerType : String
identifica il tipo dell'oggetto da interfacciare con il database;

- updates : Collection
contiene le operazioni di aggiornamento da eseguire nel db. E' un oggetto JSON che contiene i seguenti campi:

- *image:* oggetto JSON che contiene le modifiche sugli oggetti image;
- *shape:* oggetto JSON che contiene le modifiche sugli oggetti shape;
- *text:* oggetto JSON che contiene le modifiche sugli oggetti text;
- *frame:* oggetto JSON che contiene le modifiche sugli oggetti frame;
- *infographic:* oggetto JSON che contiene le modifiche sugli oggetti infographic.

- saveFunctions : Collection
contiene i nomi delle funzioni che si occupano di salvare degli oggetti nel database. I campi che contiene sono:

- *image:* contiene la funzione che si occupa di salvare le image;
- *shape:* contiene la funzione che si occupa di salvare gli shape;
- *text:* contiene la funzione che si occupa di salvare i text;
- *frame:* contiene la funzione che si occupa di salvare i frame;
- *infographic:* contiene la funzione che si occupa di salvare le infographic.

- removed : Collection

contiene le operazioni di rimozione da eseguire nel db. E' un oggetto JSON che contiene i seguenti campi:

- *image:* oggetto JSON che contiene le operazioni di rimozione degli oggetti image;
- *shape:* oggetto JSON che contiene le operazioni di rimozione degli oggetti shape;
- *text:* oggetto JSON che contiene le operazioni di rimozione degli oggetti text;
- *frame:* oggetto JSON che contiene le operazioni di rimozione degli oggetti frame;
- *infographic:* oggetto JSON che contiene le operazioni di rimozione degli oggetti infographic.

- removeFunctions : Collection

contiene i nomi delle funzioni che si occupano di rimuovere degli oggetti dal database. I campi che contiene sono:

- *image:* contiene la funzione che si occupa di rimuovere le image;
- *shape:* contiene la funzione che si occupa di rimuovere gli shape;
- *text:* contiene la funzione che si occupa di rimuovere i text;
- *frame:* contiene la funzione che si occupa di rimuovere i frame;
- *infographic:* contiene la funzione che si occupa di rimuovere l'infographic.

- inserted : Collection

contiene le operazioni di inserimento da eseguire nel db. E' un oggetto JSON che contiene i seguenti campi:

- *image:* oggetto JSON che contiene le operazioni di inserimento degli oggetti image;
- *shape:* oggetto JSON che contiene le operazioni di inserimento degli oggetti shape;
- *text:* oggetto JSON che contiene le operazioni di inserimento degli oggetti text;
- *frame:* oggetto JSON che contiene le operazioni di inserimento degli oggetti frame;
- *infographic:* oggetto JSON che contiene le operazioni di inserimento degli oggetti infographic.

- insertFunctions : Collection

contiene i nomi delle funzioni che si occupano di inserire degli oggetti sul database. I campi che contiene sono:

- *image:* contiene la funzione che si occupa di inserire le image;
- *shape:* contiene la funzione che si occupa di inserire gli shape;
- *text:* contiene la funzione che si occupa di inserire i text;
- *frame:* contiene la funzione che si occupa di inserire i frame;
- *infographic:* contiene la funzione che si occupa di inserire l'infographic.

Metodi

+ setContainer(id : String, type : String) : Saver

imposta il container da interfacciare con il Saver. Restituisce un riferimento dell'oggetto Saver.

Argomenti

- **id : String** identificativo del container;
- **type : String** definisce il tipo del container.

+ insert(GOJSON : Collection) : Saver

inserisce un operazione di inserimento da eseguire nel campo inserted. Restituisce un riferimento dell'oggetto Saver.

Argomenti

- **GOJSON : Collection** oggetto JSON che dev'essere inserito nel database.

+ update(idGO : String, type : String, update : Collection) : Saver

inserisce un operazione di modifica, nel campo updates. Restituisce un riferimento dell'oggetto Saver.

Argomenti

- **idGO : String** identifica l'id dell'oggetto da modificare;
- **type : String** identifica il tipo dell'oggetto da modificare;
- **update : Collection** oggetto JSON che contiene le modifiche da apportare sul db.

+ remove(idGO : String, type : String) : Saver

inserisce un operazione di rimozione di un oggetto, nel campo removed. Restituisce un riferimento dell'oggetto Saver.

Argomenti

- **idGO : String** identifica l'id dell'oggetto da rimuovere;
- **type : String** identifica il tipo dell'oggetto da rimuovere;

+ save() : Saver

esegue le operazioni pendenti di inserimento, modifica e rimozione sul database. Restituisce un riferimento dell'oggetto Saver.

+ init() : Saver
inizializza i campi dell'oggetto Saver. Restituisce un riferimento dell'oggetto Saver.

3.9.12 premi/client/editor/lib/Shape

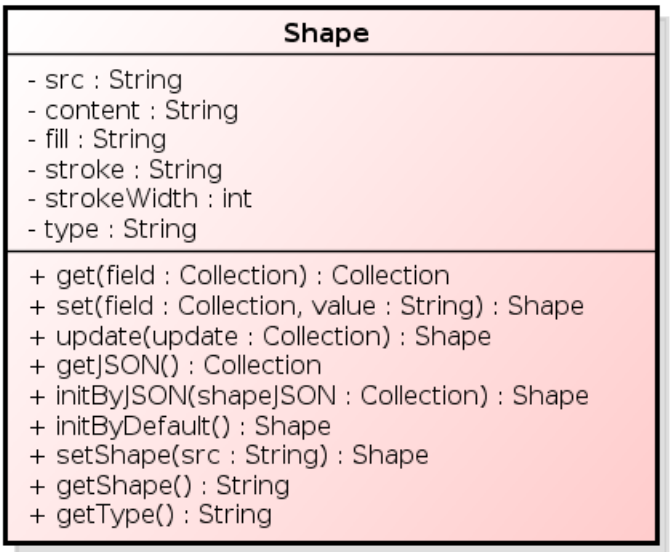

Figura 27: Diagramma della classe premi/client/editor/lib/Shape

Descrizione
è una classe che rappresenta un oggetto shape. Contiene i metodi per gestire uno shape.

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:

- *src:* definisce il percorso dello shape da visualizzare. Ogni shape è rappresentato da un immagine svg;
- *content:* definisce il contenuto dello shape;
- *fill:* definisce il colore della parte interna dell'oggetto grafico;
- *stroke:* definisce il colore del bordo dell'oggetto grafico;
- *strokeWidth:* definisce la dimensione del bordo dell'oggetto grafico;
- *type:* identifica il tipo di oggetto.

Metodi

+ **get(field : String) : String**
  restituisce una proprietà dell'oggetto shape.

  Argomenti

  - **field : String** identifica la proprietà da restituire;

+ **set(field : String, value : String) : Shape**
  permette di settare un campo dell'attributo info. Restituisce un riferimento dell'oggetto Shape.

  Argomenti

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

+ **update(update : Collection) : Shape**
  permette di aggiornare i campi dell'attributo info. Restituisce un riferimento dell'oggetto Shape.

  Argomenti

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

+ **initByJSON(shapeJSON : Collection) : Shape**
  permette di inizializzare l'attributo info tramite un oggetto JSON. Restituisce un riferimento dell'oggetto Shape.

  Argomenti

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

+ **initByDefault() : Shape**
  permette di inizializzare i campi dell'attributo info con i parametri di default. Restituisce un riferimento dell'oggetto Shape.

+ **setShape(src : String) : Shape**
  setta l'url(percorso) dello shape. Restituisce un riferimento dell'oggetto Shape.

  Argomenti

  - **src : String** identifica l'url(percorso) dello shape.

+ getShape() : String
restituisce l'url(percorso) dello shape.

+ getType() : String
restituisce la stringa shape per identificare che il tipo dell'oggetto è image.

3.9.13 premi/client/editor/lib/Text

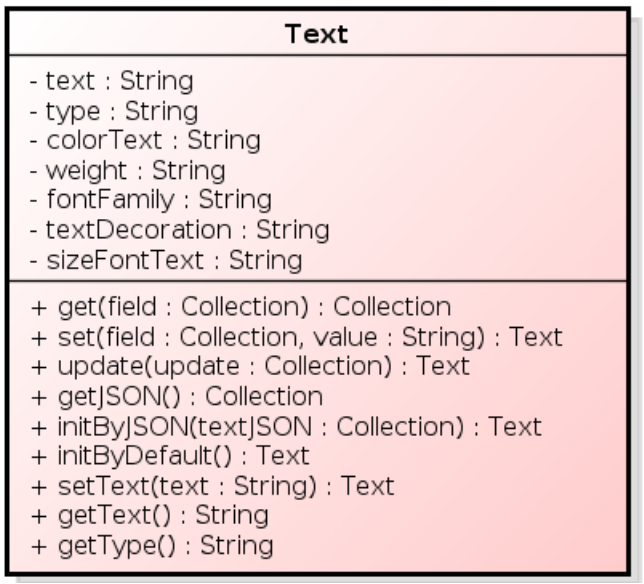

Figura 28: Diagramma della classe premi/client/editor/lib/Text

Descrizione
è una classe che rappresenta un oggetto text. Contiene i metodi per gestire uno text.

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:

- *text:* definisce il testo da visualizzare sull'oggetto;
- *colorText:* definisce il colore del testo;
- *weight:* definisce la larghezza del testo;
- *fontFamily:* definisce il font del testo;
- *textDecoration:* definisce particolari decorazioni da attribuire al testo;

- *sizeFontText:* definisce la grandezza del testo;
- *type:* identifica il tipo di oggetto.

Metodi

+ get(field : String) : String
restituisce una proprietà dell'oggetto text.

Argomenti

- **field : String** identifica la proprietà da restituire;

+ set(field : String, value : String) : Text
permette di settare un campo dell'attributo info. Restituisce un riferimento dell'oggetto Text.

Argomenti

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

+ update(update : Collection) : Text
permette di aggiornare i campi dell'attributo info. Restituisce un riferimento dell'oggetto Text.

Argomenti

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

+ initByJSON(textJSON : Collection) : Text
permette di inizializzare l'attributo info tramite un oggetto JSON. Restituisce un riferimento dell'oggetto Text.

Argomenti

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

+ initByDefault() : Text
permette di inizializzare i campi dell'attributo info con i parametri di default. Restituisce un riferimento dell'oggetto Text.

+ setText(text : String) : Collection
setta il testo del text.

Argomenti

- **text : String** identifica il testo da inserire.

+ getText() : String
restituisce il testo dell'oggetto.

+ getType() : String
restituisce la stringa shape per identificare che il tipo dell'oggetto è image.

3.10 premi/client/frameEditor

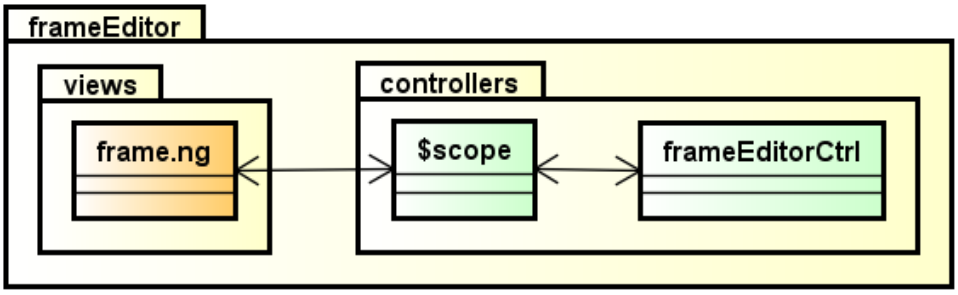

Figura 29: Diagramma del package premi/client/frameEditor

3.10.1 premi/client/frameEditor/views/frame.ng

Descrizione

Template della vista associata allo *$scope* di *frameEditorCtrl*. Fornisce tutti gli strumenti necessari alla creazione di un frame, tra cui:

- creazione, modifica e rimozione di frame
- inserimento, modifica e rimozione di immagini, shape, e testo
- spostamento di immagini, shape e testo all'interno del frame

3.10.2 premi/client/frameEditor/controllers/frameEditorCtrl

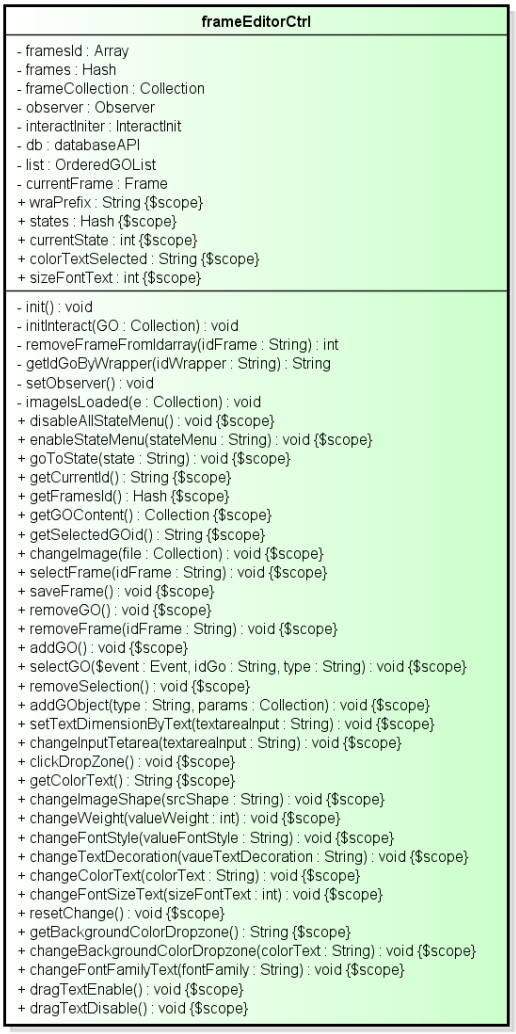

Figura 30: Diagramma della classe premi/client/frameEditor/controllers/frameEditorCtrl

Descrizione

Questo controller crea lo $scope associato alla vista generata da **frame.ng**, fornendo i dati e i metodi necessari per consentire all'utente di creare e modellare un frame, inserendo o rimuovendo oggetti grafici al suo interno.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Dipendenze

- **premi/client/presentation/lib/OrderedGoList**: per la gestione degli oggetti grafici contenuti nei frame
- **premi/client/presentation/lib/databaseAPI**: per il salvataggio dei frame nel database
- **premi/client/editor/lib/InteractInit**: per accedere alla libreria Interact.JS e offrire una rappresentazione grafica dei frame all'utente
- **premi/client/editor/lib/Frame**: per la creazione e la modifica dei frame
- **premi/client/editor/lib/Observer**: per dare un Observer agli oggetti che ne fanno uso

Attributi

- framesId : Array
Array dei codici identificativi dei frame creati dall'utente nella presentazione che sta modificando

- frames : Hash
Hash dei frames creati dall'utente in questa presentazione, strutturato come *id_frame* : *Collezione JSON dei suoi attributi*

- frameCollection : Collection
Collezione di MongoDB dei frames contenuti nella presentazione che l'utente sta modificando

- observer : Observer
Observer della classe, che andrà associato agli oggetti con cui l'utente lavora

- interactIniter : InteractInit
interactIniter è utilizzato per l'inizializzazione della libreria Interact.JS

- db : databaseAPI
Contiene una lista di metodi coi quali la classe può salvare i dati dell'utente nel database

- list : OrderedGOList
È la lista di oggetti grafici contenuti nel frame che l'utente sta modificando

- currentFrame : Frame
È il frame che l'utente sta modificando

+ wraPrefix : String
Stringa che indica quale prefisso si sta utilizzando per gli oggetti wrapper che compongono l'interfaccia grafica di questa parte di editor

+ states : Hash
Oggetto contenente una lista di stati che l'editor può assumere durante il suo utilizzo da parte dell'utente. Una volta inizializzato non può più essere modificato. Inizializzarlo con i seguenti campi:

- noSelection : 1
- imageEditing : 2
- shapeEditing : 3

- textEditing : 4
- addingGo : 5
- framesList : 6

+ **currentState : int**
  Lo stato che l'editor sta assumendo. Il suo valore può essere solo uno tra quelli rappresentati da states

+ **currentImage : String**
  Il codice identificativo dell'immagine che l'utente sta utilizzando

+ **colorTextSelected : String**
  Il colore selezionato dall'utente nell'editor

+ **sizeFontText : int**
  La dimensione del testo selezionata dall'utente nell'editor

Metodi

- init() : void
Inizializza gli attributi della classe.

Note

- chiama la classe setObserver()
- imposta l'Observer in interactIniter
- inizializza framesId e frames, prelevando le informazioni da framesCollection
- carica il primo frame, inizializzando list e currentFrame, e preparando la visualizzazione attraverso initInteract

- initInteract(GO : Collection) : void
Rappresenta in modo visivo all'utente l'oggetto grafico ricevuto, attraverso la libreria esterna Interact.JS. Utilizza i metodi initializeImage, initializeShape o initializeText di interactIniter, in base al tipo dell'oggetto grafico ricevuto.

Argomenti

- **GO : Collection** Collezione di attributi dell'oggetto grafico da rappresentare

- removeFrameFromIdarray(idFrame : String) : int
Rimuove il frame rappresentato dal codice identificativo ricevuto da framesId. Restituisce la posizione in cui si trovava il frame

Argomenti

- **idFrame : String** Il codice identificativo del frame da rimuovere

- getIdGoByWrapper() : String

Restituisce il codice identificativo dell'oggetto grafico estraendolo dal codice identificativo del wrapper (che sarà composto dall'unione tra wraPrefix e il codice identificativo dell'oggetto)

Argomenti

- **idWrapper : String** Il codice identificativo del wrapper

- setObserver() : void

Imposta l'observer per reagire agli eventi select, resize, drag, update e changeLvl inviati dalla vista, per l'aggiornamento degli attributi dell'oggetto grafico che sta venendo modificato dall'utente

- imageIsLoaded(e : Collection) : void

Aggiorna il frame appena modificato con nuovi attributi

Argomenti

- **e : Collection** Collezione di attributi modificati dall'utente

+ disableAllStateMenu() : void

Imposta currentState a states.noSelection

+ enableStateMenu(stateMenu : String) : void

Cambia currentState in base allo stato ricevuto

Argomenti

- **stateMenu : String** Lo stato in cui si trova l'editor

+ goToState(state : String) : void

Ridireziona il browser in base allo stato ricevuto

Argomenti

- **state : String** Il nuovo stato. In questo caso rappresenta una posizione, o pagina, all'interno dell'applicazione

+ getCurrentId() : String

Restituisce il codice identificativo del frame contenuto in currentFrame

+ getFramesId() : Hash

Restituisce i codici identificativi di tutti i frames creati dall'utente per la presentazione (restituisce l'attributo frames)

+ getGOContent() : Collection

Restituisce la lista degli oggetti grafici presenti nel frame che l'utente sta modificando, sfruttando il metodo getList() di list

+ getSelectedGOid() : String

Restituisce il codice identificativo dell'oggetto grafico attualmente selezionato, sfruttando il metodo getSelectedGOId() di currentFrame

+ changeImage(file : Collection) : void

Riceve una collezione di files, preleva da essa l'immagine inviata dall'utente e la interpreta attraverso il metodo FileReader() di JavaScript$_G$

Argomenti

- **file : Collection** Collezione rappresentante il file inviato dall'utente, che si troverà nell'attributo files[0]

+ selectFrame(idFrame : String) : void

Seleziona il frame associato al codice identificativo ricevuto

Argomenti

- **idFrame : String** Codice identificativo del frame da selezionare

Note

- il frame viene cercato in frames e caricato in currentFrames. Quest'ultimo va salvato prima di essere sovrascritto
- list va inizializzato caricando gli oggetti grafici contenuti nel frame
- inizializza framesId e frames, prelevando le informazioni da framesCollection
- prepara la visualizzazione di ogni oggetto grafico attraverso initInteract

+ saveFrame() : void

Richiama il metodo save() di currentFrame per aggiornare il database con le modifiche apportate dall'utente

+ removeGO() : void

Rimuove l'oggetto grafico attualmente selezionato dal frame che l'utente sta modificando

+ removeFrame(idFrame : String) : void

Rimuove il frame rappresentato dal codice identificativo ricevuto. Se il codice identificativo è vuoto rimuove il frame contenuto in currentFrame

Argomenti

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

+ addGO() : void

Imposta currentState come states.addingGO, per avvertire la vista della scelta effettuata dall'utente

+ selectGO($event : Event, idGo : String, type : String) : void

Prepara l'editor alla modifica dell'oggetto grafico in base al suo tipo

Argomenti

- **$event : Event** l'evento che ha portato alla chiamata del metodo. I suoi segnali devono essere interrotti attraverso il metodo stopPropagation()
- **idGO : String** Il codice identificativo dell'oggetto grafico che l'utente ha scelto di modificare
- **type : String** Il tipo dell'oggetto grafico da rappresentare

+ removeSelection() : void

Annulla la selezione del frame, azzerando tutti gli attributi interessati alla selezione

+ addGObject(type : String, params : Collection) : void

Aggiunge un nuovo frame, oppure un nuovo oggetto grafico al frame selezionato, e prepara l'editor alla sua modifica

Argomenti

- **type : String** Il tipo di oggetto grafico che si sta aggiungendo
- **params : Collection** Collezione di attributi del nuovo oggetto grafico

+ setTextDimensionByText(textareaInput : String) : void

Cambia la dimensione dell'input di testo in base al numero di righe inserite finora

Argomenti

- **textareaInput : String** Il testo inserito dall'utente all'interno dell'input

+ changeInputTetarea(textareaInput : String) : void

Cambia la dimensione dell'input di testo in base al numero di righe inserite finora, richiamando il metodo setTextDimensionByText, e aggiorna l'oggetto grafico associato all'input attraverso il metodo updateSelectedGO di currentFrame

Argomenti

- **textareaInput : String** Il testo inserito dall'utente all'interno dell'input

+ clickDropZone() : void

Deseleziona il frame attualmente selezionato (utilizzato quando si preme un'area vuota dell'editor)

+ **getColorText() : String**
  Restituisce il colore attualmente impostato per l'inserimento del testo (restituisce colorTextSelected)

+ **changeImageShape(srcShape : String) : void**
  Cambia lo shape attualmente selezionato, sostituendo la sua immagine con quella rappresentata dalla stringa ricevuta

  Argomenti

  - **srcShape : String** Il nome della nuova immagine scelta per lo shape

+ **changeWeight(valueWeight : int) : void**
  Cambia la grossezza del testo attualmente selezionato dall'utente

  Argomenti

  - **valueWeight : int** Il nuovo valore di grossezza del testo

+ **changeFontStyle(valueFontStyle : String) : void**
  Cambia lo stile del testo attualmente selezionato

  Argomenti

  - **valueFontStyle : String** Il nuovo stile del testo

+ **changeTextDecoration(vaueTextDecoration : String) : void**
  Cambia l'aspetto del testo attualmente selezionato

  Argomenti

  - **valueTextDecoration : String** Il nuovo aspetto del testo

+ **changeColorText(colorText : String) : void**
  Cambia il colore del testo attualmente selezionato

  Argomenti

  - **colorText : String** Il nuovo colore del testo

+ **changeFontSizeText(sizeFontText : int) : void**
  Cambia la dimensione del testo attualmente selezionato

  Argomenti

  - **sizeFontText : int** La nuova dimensione del testo

+ **resetChange() : void**
  Annulla le modifiche apportate all'oggetto grafico attualmente selezionando, azzerando i suoi attributi

+ **getBackgroundColorDropzone() : String**
  Restituisce il colore di background del frame attualmente selezionato

+ **changeBackgroundColorDropzone(colorText : String) : void**
  Cambia il colore di background del frame attualmente selezionato

  Argomenti

  - **colorText : String** Il nuovo colore dello sfondo, in formato esadecimale

+ **changeBackgroundColorDropzone(colorText : String) : void**
  Cambia il colore di background del frame attualmente selezionato

  Argomenti

  - **colorText : String** Il nuovo colore dello sfondo, in formato esadecimale

+ **changeFontFamilyText(fontFamily : String) : void**
  Cambia il carattere del testo attualmente selezionato

  Argomenti

  - **fontFamily : String** Il nuovo tipo di carattere del testo

+ **dragTextEnable() : void**
  Consente di visualizzare il testo spostato nell'oggetto grafico selezionato. Utilizza il metodo initializeText di interactIniter

+ **dragTextEnable() : void**
  Annulla la visualizzazione del testo spostato nell'oggetto grafico selezionato. Utilizza il metodo unset di interactIniter

3.11 premi/client/infographicEditor

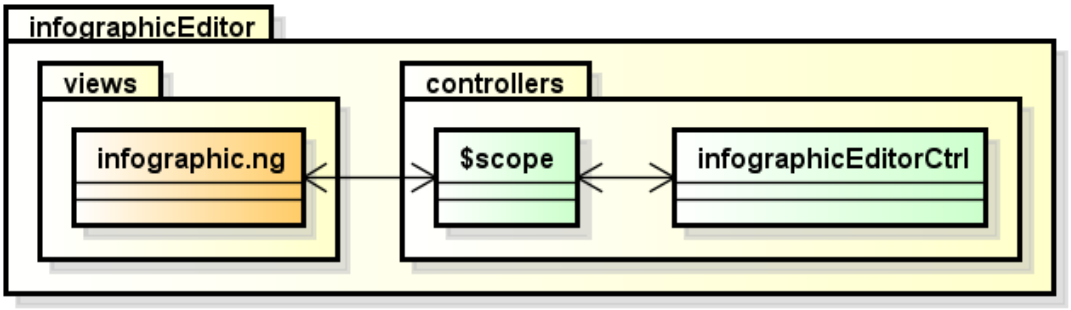

Figura 31: Diagramma del package premi/client/infographicEditor

3.11.1 premi/client/infographicEditor/views/infographic.ng

Descrizione

Template della vista associata allo *$scope* di *infographicEditorCtrl*. Fornisce tutti gli strumenti necessari alla modifica dell'infografica della presentazione, tra cui:

- aggiunta e rimozione dei frame creati finora dall'utente
- inserimento, modifica e rimozione di immagini, shape, e testo
- spostamento e ridimensionamento di frame, immagini, shape e testo all'interno dell'infografica

3.11.2 premi/client/infographicEditor/controllers/infographicEditorCtrl

Figura 32: Diagramma della classe premi/client/infographicEditor/controllers/infographicEditorCtrl

Descrizione

Questo controller crea lo $scope associato alla vista generata da **infographic.ng**, fornendo i dati e i metodi necessari per consentire all'utente di creare e modellare l'infografica di una presentazione, inserendo o rimuovendo frame e altri oggetti grafici al suo interno.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Dipendenze

- **premi/client/presentation/lib/databaseAPI**: per l'accesso ai metodi che interagiscono con il database
- **premi/client/editor/lib/InteractInit**: per collegarsi alla libreria esterna Interact.js e visualizzare oggetti grafici e frame
- **premi/client/editor/lib/Observer**: fornisce agli oggetti che compongono l'infografica un observer per l'aggiornamento in tempo reale delle modifiche apportate ad essi
- **premi/client/editor/lib/Infographic**: per la creazione e modifica dell'infografica
- **premi/client/presentation/lib/OrderedGOList**: per la gestione degli oggetti grafici contenuti all'interno dell'infografica

Attributi

- frameCollection : Collection
Collezione di MongoDB di tutti i frame creati dall'utente per la presentazione che sta modificando

- infographicCollection : Collection
Collezione di MOngoDB. Contiene gli attributi dell'infografica associata alla presentazione che l'utente intende modificare

- infFrames : Collection
Collezione di MongoDB dei frames finora inseriti all'interno dell'infografica

- observer : Observer
Oggetto Observer da collegare agli oggetti che compongono l'infografica

- interactIniter : InteractInit
Serve a inizializzare la libreria Interact.js per la visualizzazione degli oggetti grafici

- db : databaseAPI
Oggetto creato per l'utilizzo dei metodi statici contenuti al suo interno, che permettono al controller di accedere al server per il salvataggio delle modifiche apportate

- frames : Hash
Lista di frames inseriti finora nell'infografica, composta da *frame_id* : *collezione di attributi*

- hashFrameIndex : Hash
Lista di codici identificativi dei frame inseriti finora nell'infografica, associati alla loro posizione all'iterno di frameCollection

- infContents : OrderedGOList
Lista ordinata degli oggetti grafici inseriti finora all'interno dell'infografica

+ wraPrefix : String
Prefisso del codice identificativo dei contenitori degli oggetti grafici nella vista. Va inizializzato a *wrapper-*

+ **states : Hash**
  Oggetto contenente una lista di stati che l'editor può assumere durante il suo utilizzo da parte dell'utente. Una volta inizializzato non può più essere modificato. Inizializzarlo con i seguenti campi:
  - noSelection : 1
  - imageEditing : 2
  - shapeEditing : 3
  - textEditing : 4
  - addingGo : 5
  - addingFrame : 6
  - addingImage : 7
  - addingShape : 8
  - addingText : 9
  - frameEditing : 10
  - framesList : 11
  - goList : 12

+ **currentState : int**
  Lo stato che l'editor sta assumendo. Il suo valore può essere solo uno tra quelli rappresentati da states

+ **standard_height : int**
  L'altezza di default dell'infografica

+ **standard_width : int**
  La lunghezza di default dell'infografica

+**infTransOn : boolean**
  Se impostato a *true* rende l'infografica trasparente

Metodi

- init() : void
Inizializza gli attributi del controller e dello *$scope*

Note

- inizializza frames dalla collezione infFrames
- inizializza hashFrameIndex da frameCollection
- inizializza infographic utilizzando il suo metodo initByJSON
- inizializza l'observer collegandolo, tramite il metodo on, ai segnali *select, drag, resize, selectInf, dragInf, dragInfStart, dragInfEnd, changeLvl*

- initInteract(GO : Collection) : void
Visualizza l'oggetto grafico ricevuto attraverso la liberia Interact.JS

Argomenti

- **Go : Collection** Lista di attributi di un oggetto grafico qualsiasi

- getIdGOByWrapper(idWrapperGO : String) : String

Estrae il codice identificativo di un oggetto grafico dal suo wrapper (rimuovendo il prefisso del wrapper)

Argomenti

- **idWrapperGO : String** Codice identificativo del wrapper dell'oggetto grafico

- insertInLvlArray() : void

Inserisce frames e oggetti grafici dentro infContents, estraendoli da infographic tramite i metodi getFramesContent() e getGOContent()

- imageIsLoaded(e : Event) : void

Riceve l'immagine caricata dall'utente, aggiorna l'oggetto grafico corrente e mostra il cambiamento nella vista

Argomenti

- **e : Evento** Evento JavaScript collegato alla ricezione dell'immagine

- setTextDimensionByText(textareaInput : String) : void

Imposta la dimensione dell'area di input in base alle righe di testo inserite finora al suo interno

Argomenti

- **textareaInput : String** Il testo finora inserito all'interno dell'area di testo

+ disableAllStateMenu() : void

Azzera currentState impostandolo a states.noSelection

+ enableStateMenu(stateMenu : String) : void

Imposta currentState in base allo stato ricevuto

Argomenti

- **stateMenu : String** Lo stato associato ai possibili valori di currentState

+ goToState(state : String) : void

Mostra all'utente la vista associata allo stato ricevuto, reindirizzando il browser ad un'altra parte di applicazione

Argomenti

- **state : String** Lo stato associato alla parte di applicazione che l'utente ha scelto di visitare

+ incrementLvl($event : Event, idGO : String) : void

Interrompe la propagazione dell'evento ricevuto, e aumenta di livello l'oggetto grafico associato al codice identificativo ricevuto attraverso il metodo upgradeGO di infContents

Argomenti

- **$event : Event** L'evento associato all'azione dell'utente
- **idGO : String** Il codice identificativo dell'oggetto grafico da aumentare di livello

+ decrementLvl($event : Event, idGO : String) : void

Interrompe la propagazione dell'evento ricevuto, e diminuisce di livello l'oggetto grafico associato al codice identificativo ricevuto attraverso il metodo downgradeGO di infContents

Argomenti

- **$event : Event** L'evento associato all'azione dell'utente
- **idGO : String** Il codice identificativo dell'oggetto grafico da diminuire di livello

+ getGOContent() : Collection

Restituisce la lista degli oggetti grafici inseriti finora nell'infografica attraverso il metodo getList di infContents

+ hasInfFrame(idFrame : String) : boolean

Restituisce *true* se il frame associato al codice identificativo ricevuto è presente all'interno dell'infografica, oppure *false* altrimenti. Utilizza il metodo hasFrame di infographic

Argomenti

- **idFrame : String** Il codice identificativo del frame da cercare

+ getSelectedGOId() : String

Restituisce il codice identificativo dell'oggetto grafico attualmente selezionato dall'utente. Sfrutta il metodo getSelectedGOId di infographic

+ changeImage(file : Collection) : void

Riceve una collezione di files, preleva da essa l'immagine inviata dall'utente e la interpreta attraverso il metodo FileReader() di JavaScript$_G$

Argomenti

- **file : Collection** Collezione rappresentante il file inviato dall'utente, che si troverà nell'attributo files[0]

+ selectGO($event : Event, idGO : String, type : String) : void

Prepara l'editor alla modifica dell'oggetto grafico in base al suo tipo

Argomenti

- **$event : Event** l'evento che ha portato alla chiamata del metodo. I suoi segnali devono essere interrotti attraverso il metodo stopPropagation()
- **idGO : String** Il codice identificativo dell'oggetto grafico che l'utente ha scelto di modificare
- **type : String** Il tipo dell'oggetto grafico da rappresentare

+ addGO() : void

Imposta currentState come states.addingGO, per avvertire la vista della scelta effettuata dall'utente

+ clickDropZone() : void

Deseleziona l'oggetto grafico attualmente selezionato (utilizzato quando si preme un'area vuota dell'editor)

+ addGObject(type : String, params : Collection) : void

Aggiunge un nuovo oggetto grafico all'infografica, e prepara l'editor alla sua modifica

Argomenti

- **type : String** Il tipo di oggetto grafico che si sta aggiungendo
- **params : Collection** Collezione di attributi del nuovo oggetto grafico

+ addFrame(idFrame : String) : void

Aggiunge un frame all'infografica

Argomenti

- **idFrame : String** Il codice identificativo del frame da aggiungere

+ getPresFrames() : Collection

Restituisce la collezione di tutti i frame creati finora dall'utente (restituisce frameCollection)

+ removeSelection() : void

Deseleziona l'oggetto grafico attualmente selezionato chiamando il metodo deselectGO di infographic, e imposta currentState a states.noSelection

+ removeGO() : void

Rimuove l'oggetto grafico attualmente selezionato da parte dell'utente

+ saveInfographic() : void

Salva le modifiche apportate all'infografica nel database, attraverso il metodo save() di infographic

3.12 premi/client/trailsEditor

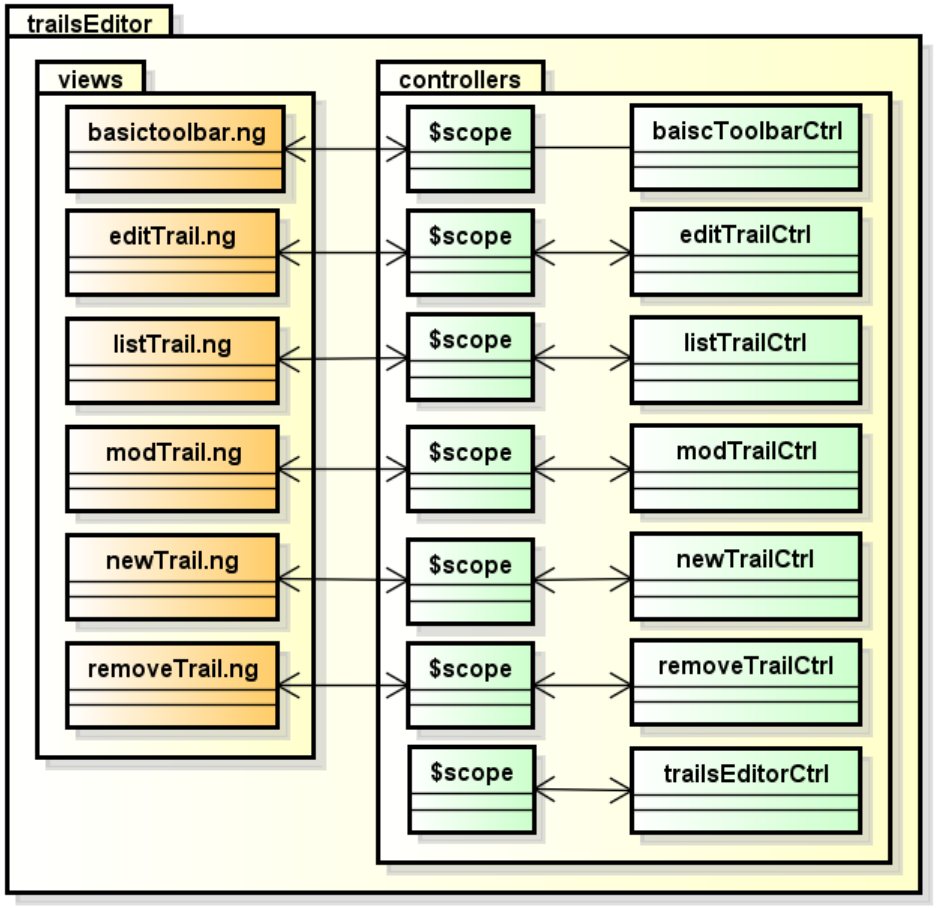

Figura 33: Diagramma del package premi/client/trailsEditor

3.12.1 premi/client/trailsEditor/views/basicToolbar.ng

Descrizione

Template della vista associata allo *$scope* di *basicToolbarCtrl*. Fornisce una toolbar per la navigazione tra le varie fasi della modifica della presentazione

Note

- Deve possedere un bottone per ogni fase dell'editor (Gestione Frame, Gestione Infografica e Gestione Trail)
- Deve possedere un bottone per l'uscita dall'editor

3.12.2 premi/client/trailsEditor/views/editTrail.ng

Descrizione

Template della vista associata allo *$scope* di *editTrailCtrl*. Permette la modifica del titolo del trail selezionato dall'utente

Note

- Mostra il titolo del trail in un input HTML$_G$, modificabile, attraverso l'attributo dello scope *Trail.title*
- Possiede un bottone associato al metodo *save()* dello *$scope* per salvare le modifiche apportate al Trail
- Possiede un bottone associato al metodo *discard()* dello *$scope* per annullare le modifiche effettuate sul trail

3.12.3 premi/client/trailsEditor/views/listTrail.ng

Descrizione

Template della vista associata allo *$scope* di *listTrailCtrl*. Mostra la lista dei percorsi creati finora dall'utente associati alla presentazione

Note

- mostra una lista di tutti i percorsi attraverso l'attributo dello scope *Trails*
- mostra una lista di tutti i frame inseribili attraverso il metodo dello scope *getFramesId()*

3.12.4 premi/client/trailsEditor/views/modTrail.ng

Descrizione

Template della vista associata allo *$scope* di *modTrailCtrl*. Deve consentire la modifica di un percorso in tutti i suoi attributi:

- inserimento di un frame in qualsiasi punto del percorso
- inserimento dello stesso frame più volte nel percorso
- spostamento di un frame da un punto all'altro del percorso
- eliminazione di un frame dal percorso
- trasformazione del frame in checkpoint
- elimiazione di un checkpoint

3.12.5 premi/client/trailsEditor/views/newTrail.ng

Descrizione

Template della vista associata allo *$scope* di *newTrailCtrl*. Fornisce all'utente i comandi per l'inserimento di un novo trail associato alla presentazione nel database

Note

- Mostra un input HTML$_G$ associato all'attributo dello scope *title* per l'inserimento del titolo del trail

- Possiede un bottone associato al metodo *save()* dello *$scope* per salvare il nuovo Trail nel database
- Possiede un bottone associato al metodo *discard()* dello *$scope* per annullare il processo di creazione del trail

3.12.6 premi/client/trailsEditor/views/removeTrail.ng

Descrizione

Template della vista associata allo *$scope* di *removeTrailCtrl*. Fornisce all'utente i comandi per la rimozione di un trail associato alla presentazione dal database

Note

- Mostra un messaggio di conferma eliminazione del trail
- Possiede un bottone associato al metodo *remove()* dello *$scope* confermare la rimozione
- Possiede un bottone associato al metodo *discard()* dello *$scope* per annullare il processo di rimozione del trail

3.12.7 premi/client/trailsEditor/controllers/basicToolbarCtrl

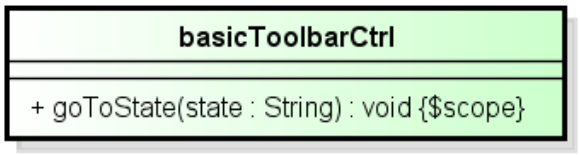

Figura 34: Diagramma della classe premi/client/trailsEditor/controllers/basicToolbarCtrl

Descrizione

Controller della view *basicToolbar.ng*. Fornisce, tramite lo *$scope* un metodo per passaggio da un editor all'altro
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Metodi

+ goToState(state : String) : void

Tramite il metodo *$state.go* cambia stato dell'editor, passando da una fase di creazione della presentazione all'altra.

Argomenti

- **state : String** Il nuovo stato dell'editor. Gli stati dell'editor al momento sono:
  - premi.editor.frame
  - premi.editor.infographic
  - premi.editor.trails

3.12.8 premi/client/trailsEditor/controllers/editTrailCtrl

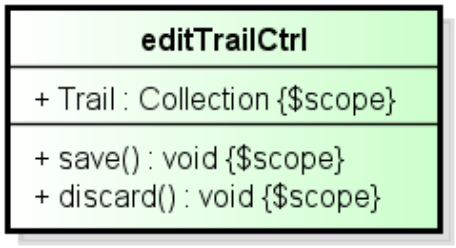

Figura 35: Diagramma della classe premi/client/trailsEditor/controllers/editTrailCtrl

Descrizione

Controller della view *editTrail.ng*. Fornisce, tramite lo *$scope*, metodi e attributi necessari alla modifica del titolo di un trail.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Dipendenze

- **client/presentation/lib/databaseAPI**: per salvare il trail modificato

Attributi

- Trail:Collection
Collezione di MongoDB degli attributi di un trail

Metodi

+ save() : void
Utilizza il metodo *+ updateTrailTitle(idTrail, title)* di databaseAPI per l'aggiornamento del Trail nel database. Aggiorna poi la pagina con il cambiamento apportato

+ discard() : void

Annulla le modifiche effettuate dall'utente sul titolo del trail. Aggiorna poi la pagina riportandola allo stato precedente alla modifica

3.12.9 premi/client/trailsEditor/controllers/listTrailCtrl

Figura 36: Diagramma della classe premi/client/trailsEditor/controllers/listTrailCtrl

Descrizione

Controller della view *listTrail.ng*. Fornisce, tramite lo *$scope*, la lista dei trails associati alla presentazione che l'utente sta modificando

Attributi

- Trails:Collection

Collezione di MongoDB di tutti i trails associati alla presentazione (vengono pubblicati dal pattern publish-subscribe$_G$ al caricamento della pagina)

3.12.10 premi/client/trailsEditor/controllers/modTrailCtrl

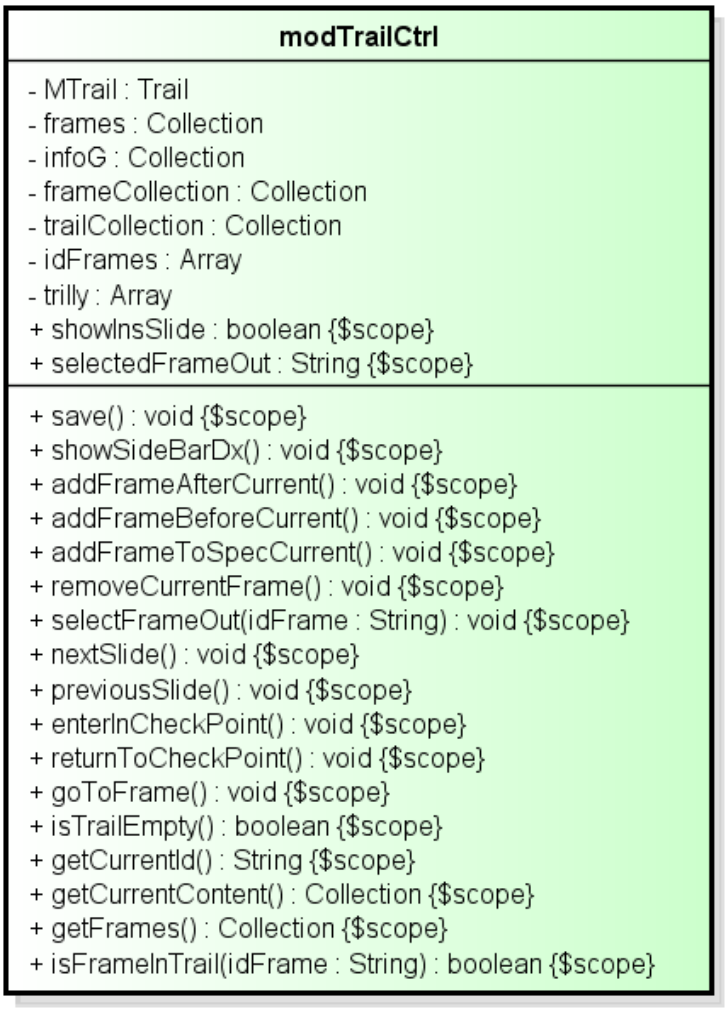

Figura 37: Diagramma della classe premi/client/trailsEditor/controllers/modTrailCtrl

Descrizione

Controller della view *modTrail.ng*. Permette, tramite lo *$scope*, di modificare un trail in ogni suo aspetto, aggiungendo o togliendo frame, o creando percorsi di specializzazione. Fornisce all'utente la possibilità di scorrere il percorso con i quattro tasti freccia della tastiera.
La dicitura {$scope} nel diagramma UML$_G$ indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Dipendenze

- **premi/client/presentation/Trail**: per la gestione del trail

Attributi

- MTrail : Trail
Oggetto Trail da modificare. Inizialmente vuoto, dev'essere inizializzato con gli attributi di trailCollection

- frames : Collection
Collezione di frames, nella forma *id_frame* : frame_data

- infoG : Collection
Collezione degli attributi dell'infografica della presentazione

- frameCollection : Collection
Collezione di MongoDB_G di tutti i frames della presentazione che sono stati inseriti nell'infografica

- trailCollection : Collection
Collezione di MongoDB_G di attributi del trail da modificare.

- idFrames : Array
Array di codici identificativi dei frames presenti in FrameCollection

- trilly : Array
Matrice vuota, utilizzata come variabile d'appoggio per l'utilizzo del metodo di MTrail initPath per la sua inizializzazione

+ showInSide : boolean
Indica se la barra di destra dell'editor dev'essere visualizzata (*true*) o meno (*false*)

+ selectedFrameOut : String
È il frame che l'utente sta selezionando nella lista di tutti i frame della presentazione

Metodi

+ save() : void
utilizza il metodo updateTrail di $meteor per il salvataggio della slide

+ showSideBarDx() : void
Attiva o disattiva la barra laterale destra dell'editor impostando a *true* showInSide se era impostato a *false* e viceversa

+ addFrameAfterCurrent()
utilizza il metodo insertSlideAfterCurrent di MTrail inserire la slide attualmente selezionata(selectedFrameOut) nella lista dopo quella selezionata nel percorso

+ addFrameAfterCurrent()
utilizza il metodo insertSlideAfterCurrent di MTrail per inserire la slide attualmente selezionata(selectedFrameOut) nella lista prima di quella selezionata nel percorso

+ **addFrameBeforeCurrent()**
  utilizza il metodo insertSlideBeforeCurrent di MTrail per inserire la slide attualmente selezionata(selectedFrameOut) nella lista prima di quella selezionata nel percorso

+ **addFrameToSpecCurrent()**
  utilizza il metodo insertSlideInSpecTrail di MTrail per inserire la slide attualmente selezionata(selectedFrameOut) nella lista nel percorso di specializzazione di quella selezionata nel percorso

+ **removeCurrentFrame()**
  utilizza il metodo removeCurrentSlide di MTrail per rimuovere la slide selezionata nel percorso

+ **selectFrameOut(idFrame : String) : void**
  Copia il codice identificativo del frame ricevuto dentro selectedFrameOut, rendendolo in questo modo selezionato.

  Argomenti

  - **idFrame : String** Il codice identificativo del frame da selezionare

+ **nextSlide()**
  utilizza il metodo nextSlide di MTrail per avanzare di un passo nel trail

+ **previousSlide()**
  utilizza il metodo nextSlide di MTrail per retrocedere di un passo nel trail

+ **enterInCheckPoint()**
  utilizza il metodo enterInCheckPoint di MTrail per entrare nel percorso di specializzazione associato al frame attualmente selezionato nel percorso, se il frame funge da checkpoint

+ **returnToCheckPoint()**
  utilizza il metodo returnToCheckPoint di MTrail per uscire dal percorso di specializzazione e tornare al percorso dove risiede il checkpoint

+ **goToFrame(idSlide : String) : void**
  Rende un frame selezionato nel percorso

  Argomenti

  - **idSlide : String** Il codice identificativo del frame da selezionare

+ **isTrailEmpty() : bool**
  utilizza il metodo isTrailEmpty di MTrail per restituire *true* se il percorso è vuoto, *false* altrimenti

+ getCurrentId() : String
utilizza il metodo getCurrentId di MTrail per restituire il codice identificativo del frame attualmente selezionato nel percorso

+ getCurrentContent() : Collection
utilizza il metodo getCurrentId di MTrail per ricevere il codice identificativo del frame attualmente selezionato nel percorso, per poi utilizzarlo per restituire gli attributi del frame, reperibili dalla collezione frames

+ getFrames() : Collection
restituisce l'intera collezione frames

+ isFrameInTrail(idFrame : String) : bool
utilizza il metodo isSlideInTrail di MTrail per restituire *true* se il frame è nel percorso, *false* altrimenti

Argomenti

- **idFrame : String** Il codice identificativo del frame da cercare

3.12.11 premi/client/trailsEditor/controllers/newTrailCtrl

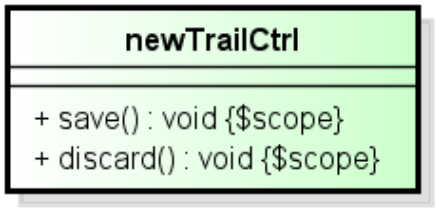

Figura 38: Diagramma della classe premi/client/trailsEditor/controllers/newTrailCtrl

Descrizione
Controller della view *newTrail.ng*. Fornisce, tramite lo *$scope*, metodi e attributi necessari alla creazione di un nuovo trail

Dipendenze

- **premi/client/presentation/lib/databaseAPI**: per i metodi necessari al salvataggio del nuovo trail nel database

Metodi

+ save() : void
Utilizza il metodo *+ insertTrail* di databaseAPI per il salvataggio del nuovo Trail nel database. Aggiorna poi la pagina con il cambiamento apportato al database

+ discard() : void
Annulla il processo di creazione del trail. Aggiorna poi la pagina riportandola allo stato precedente

3.12.12 premi/client/trailsEditor/controllers/removeTrailCtrl

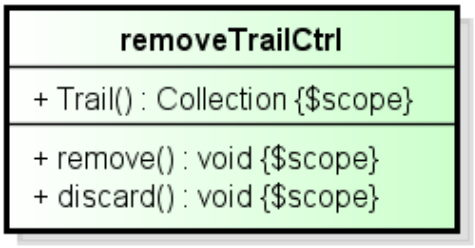

Figura 39: Diagramma della classe premi/client/trailsEditor/controllers/removeTrailCtrl

Descrizione
Controller della view *removeTrail.ng*. Fornisce, tramite lo *$scope*, metodi e attributi necessari all'eliminazione di un trail

Dipendenze

- **premi/client/presentation/lib/databaseAPI**: per i metodi necessari all'eliminazione del trail dal database

Metodi

+ remove() : void
Utilizza il metodo *+ removeTrail* di databaseAPI per la rimozione del trail dal database. Aggiorna poi la pagina con il cambiamento apportato al database

+ discard() : void
Annulla il processo di eliminazione del trail. Aggiorna poi la pagina riportandola allo stato precedente

3.12.13 premi/client/trailsEditor/controllers/trailsEditorCtrl

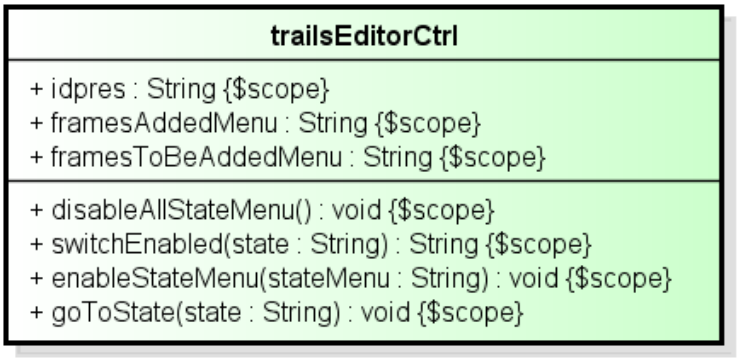

Figura 40: Diagramma della classe premi/client/trailsEditor/controllers/trailsEditorCtrl

Descrizione

Controller generale dell'editor dei trail. Prepara lo *$scope* con attributi e metodi utili alle viste e ai controller interni a questo editor.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Attributi

+ **idPres : String**
  Codice identificativo della presentazione che l'utente sta modificando, è prelevato da *$stateParams*
+ **framesAddedMenu : String**
  Indica se il menu dei frames aggiunti finora al trail è visualizzato (*enabled*) o meno (*disabled*
+ **framesToBeAddedMenu : String**
  Indica se il menu di tutti i frames della presentazione è visualizzato (*enabled*) o meno (*disabled*

Metodi

+ **disableAllStateMenu() : void**
  Disabilita i menu dei frame e aggiorna la pagina

+ **switchEnabled(state : String) : void**
  Se lo stato ricevuto è '*enabled*' lo rende '*disabled*', e viceversa

Argomenti

- **state : String** Lo stato da abilitare/disabilitare

+ enableStateMenu(stateMenu : String) : void

Abilita il menu rappresentato da stateMenu

Argomenti

- **stateMenu : String** Il menu da abilitare, può essere:
  - frameAddedMenu
  - framesToBeAddedMenu

+ goToState(state : String) : void

Rimanda l'utente alla vista rappresentata dallo stato ricevuto

Argomenti

- **state : String** Lo stato che rappresenta la vista da mostrare all'utente

3.13 premi/client/userManager

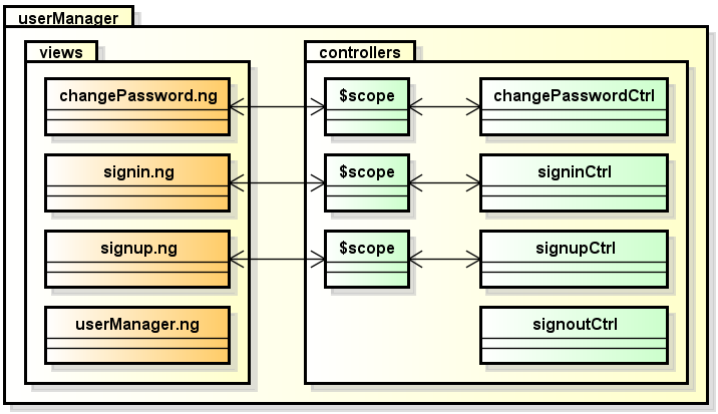

Figura 41: Diagramma del package premi/client/userManager

3.13.1 premi/client/userManager/views/changePassword.ng

Descrizione

Template della vista associata allo *$scope* di *changePasswordCtrl*. Permette il cambio della password dell'utente

Note

- prevede input per l'inserimento dell'email, della vecchia password, della nuova password, e della conferma della nuova password
- prevede due pulsanti, uno per confermare il cambio password, l'altro per annullare il cambiamento e tornare alla pagina principale dell'applicazione

3.13.2 premi/client/userManager/views/signin.ng

Descrizione

Template della vista associata allo *$scope* di *signinCtrl*. Permette all'utente di effettuare il login e di entrare nell'applicazione

Note

- prevede due input, per l'inserimento dell'email e della password
- prevede due pulsanti, uno per effettuare il login, l'altro per passare alla pagina di registrazione nel caso in cui l'utente non sia ancora registrato

3.13.3 premi/client/userManager/views/signup.ng

Descrizione

Template della vista associata allo *$scope* di *signupCtrl*. Permette all'utente di registrarsi nel database, inserendo email e password

Note

- prevede tre input, per l'inserimento dell'email, della password e della conferma della password
- prevede due pulsanti, uno per registrarsi, l'altro per effettuare il login nel caso in cui l'utente sia già registrato
- se l'utente è già registrato segnala la presenza dell'email all'interno del database

3.13.4 premi/client/userManager/views/userManager.ng

Descrizione

Genera una vista per indirizzare l'utente alle varie opzioni di gestione dell'account (signin, signout, cambio password)

3.13.5 premi/client/userManager/controllers/changePasswordCtrl

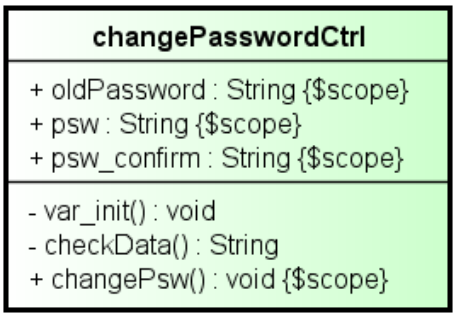

Figura 42: Diagramma della classe premi/client/userManager/controllers/changePasswordCtrl

Descrizione

Controller della view *changePassword.ng*. Permette all'utente di cambiare la password del suo account.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Dipendenze

- **premi/client/lib/toastMessageFactory**: per l'invio di notifiche o messaggi di errore all'utente

Attributi

+ **oldPassword : String**
  La vecchia password dell'account, inserita dall'utente nell'apposito input
+ **psw : String**
  La nuova password dell'account, inserita dall'utente nell'apposito input
+ **psw_confirm : String**
  Conferma della nuova password dell'account, inserita dall'utente nell'apposito input

Metodi

- **var_init() : void**
  Inizializza a stringhe vuote oldPassword, psw e psw_confirm

- **checkData() : String**
  Controlla che le informazioni scritte dall'utente siano corrette. Restituisce una stringa vuota se corrette, o una descrizione sull'errore rilevato se incorrette

+ **changePsw() : void**
  Effettua il cambio di password sfruttando il metodo changePassword di $meteor Se sono presenti errori invia un messaggio tramite toastMessageFactory

3.13.6 premi/client/userManager/controllers/signinCtrl

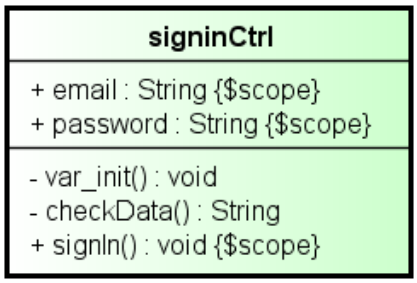

Figura 43: Diagramma della classe premi/client/userManager/controllers/signinCtrl

Descrizione

Controller della view *signin.ng*. Permette all'utente di effettuare il login.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;

- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Dipendenze

- **premi/client/lib/toastMessageFactory**: per l'invio di notifiche o messaggi di errore all'utente

Attributi

+ **email : String**
  L'email con cui l'utente si è registrato nel database dell'applicazione

+ **password : String**
  La password dell'account dell'utente

Metodi

- **var_init() : void**
  Inizializza a stringhe vuote email e password

- **checkData() : String**
  Controlla che le informazioni scritte dall'utente siano corrette. Restituisce una stringa vuota se corrette, o una descrizione sull'errore rilevato se incorrette

+ **signIn() : void**
  Effettua il login sfruttando il metodo loginWithPassword di $meteor Se sono presenti errori invia un messaggio tramite toastMessageFactory, altrimenti manda l'utente alla lista delle presentazioni

3.13.7 premi/client/userManager/controllers/signoutCtrl

Figura 44: Diagramma della classe premi/client/userManager/controllers/signoutCtrl

Descrizione

Controller per il logout dell'utente. Chiama la funzione logout di $meteor e rimanda l'utente alla pagina principale.

3.13.8 premi/client/userManager/controllers/signupCtrl

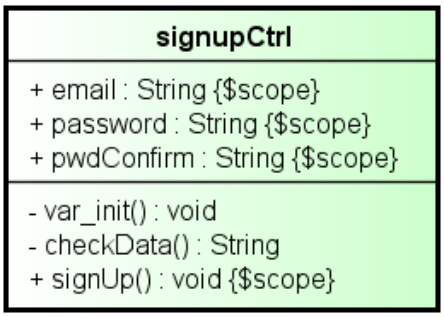

Figura 45: Diagramma della classe premi/client/userManager/controllers/signupCtrl

Descrizione

Controller della view *signup.ng*. Permette all'utente di registrarsi all'interno del database.
La dicitura {$scope} nel diagramma UML$_G$ indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Dipendenze

- **premi/client/lib/toastMessageFactory**: per l'invio di notifiche o messaggi di errore all'utente

Attributi

+ **email : String**
  L'email con cui l'utente intende registrarsi

+ **password : String**
  La password con cui l'utente intende registrarsi

+ **pwdConfirm : String**
  La conferma della password con cui l'utente intende registrarsi

Metodi

- **var_init() : void**
  Inizializza a stringhe vuote email, password e pwdConfirm

- checkData() : String

Controlla che le informazioni scritte dall'utente siano corrette. Restituisce una stringa vuota se corrette, o una descrizione sull'errore rilevato se incorrette

+ signUp() : void

Effettua la registrazione sfruttando il metodo createUser di $meteor Se sono presenti errori invia un messaggio tramite toastMessageFactory, altrimenti manda l'utente alla lista delle presentazioni

3.14 premi/client/viewer

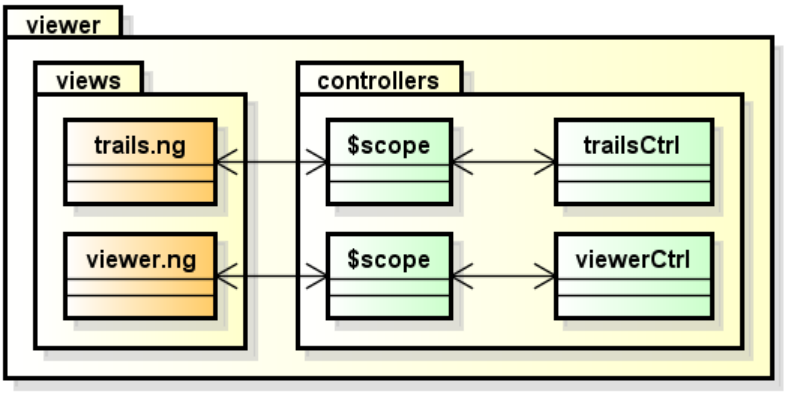

Figura 46: Diagramma del package premi/client/viewer

3.14.1 premi/client/viewer/views/trails.ng

Descrizione

Template della vista associata allo *$scope* di *trailsCtrl*. Fornisce la lista dei trail associati alla presentazione che l'utente intende visualizzare

Note

- per ogni trail fornisce un pulsante per avviare la presentazione

3.14.2 premi/client/viewer/views/viewer.ng

Descrizione

Template della vista associata allo *$scope* di *viewerCtrl*. Visualizza la presentazione secondo il trail, o percorso, scelto dall'utente. Sfrutta la libreria esterna impress.js per lo scorrimento dei frame

Note

- fornisce i comandi necessari allo scorrimento del trail:
  - avanti
  - indietro
  - entra in checkpoint
  - esci da percorso di specializzazione
- fornisce un pulsante per tornare indietro al menu dei trail

3.14.3 premi/client/viewer/controllers/trailsCtrl

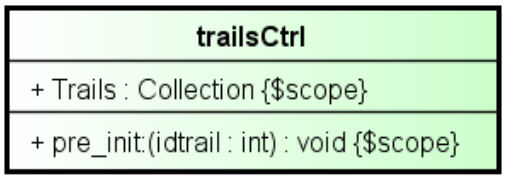

Figura 47: Diagramma della classe premi/client/viewer/controllers/trailsCtrl

Descrizione

Controller della vista generata dal template *trails.ng*. Fornisce, tramite lo *$scope*, la lista di tutti i trail creati dall'utente per la presentazione selezionata.
La dicitura {$scope} nel diagramma UML$_G$ indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Attributi

- Trails : Collection

Collezione di MongoDB dei trail creati dall'utente per la presentazione selezionata. I trail vengono pubblicati prima del caricamento del controller, tramite il pattern publish-subscribe$_G$

Metodi

+ pre_init(idtrail)

Manda l'utente alla lista dei trail

3.14.4 premi/client/viewer/controllers/viewerCtrl

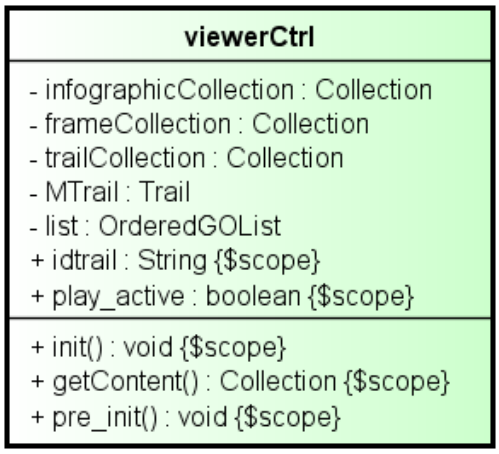

Figura 48: Diagramma della classe premi/client/viewer/controllers/viewerCtrl

Descrizione

Controller della vista generata dal template *viewer.ng*. Fornisce, tramite lo *$scope*, gli attributi ed i metodi necessari alla visualizzazione a allo scorrimento della presentazione tramite la libreria impress.js.
La dicitura {$scope} nel diagramma UML_G indica che:

- tutti gli attributi e i metodi pubblici del controller vanno inseriti nello $scope;
- tutti gli attributi e i metodi privati del controller appartengono al controller.

Vedere la sezione 3.3 per approfondimenti sull'oggetto $scope.

Dipendenze

- **premi/client/presentation/lib/Trail**: per la gestione del percorso della presentazione
- **premi/client/editor/lib/Infographic**: per rappresentare l'infografica come sfondo della presentazione
- **premi/client/presentation/lib/OrderedGOList**: per la gestione della lista degli oggetti grafici dell'infografica

Attributi

- infographicCollection : Collection
Contiene l'infografica della presentazione

- frameCollection : Collection
Collezione dei frame inseriti all'interno dell'infografica

- trailCollection : Collection
Contiene il trail che l'utente intende visualizzare

- MTrail : Trail
Oggetto Trail per la gestione del percorso di presentazione

- list : OrderedGOList
Oggetto OrderedGOList per il caricamento degli oggetti grafici dei frame e dell'infografica

+ idtrail : String
Codice identificativo del trail che l'utente intende visualizzare

+ play_active : boolean
Indica se la riproduzione della presentazione è attiva (*true*) o meno (*false*)

Metodi

+ init() : void
Imposta play_active a *true* e invia un segnale a impress.js per avviare la presentazione

+ getContent() : Collection
Restituisce la lista degli oggetti grafici caricati finora (restituisce *list.getList()*)

+ pre_init() : void
Prepara gli attributi privati dello *$scope* per la visualizzazione della presentazione

Note

- preleva l'infografica tramite l'id della presentazione passato come parametro, e la inserisce in infographicCollection
- preleva la lista dei frame che sono presenti dentro l'infografica, e li inserisce in frameCOllection
- preleva il trail tramite il codice identificativo passato come parametro, e lo inserisce in trailCollection
- inserisce i frame e gli oggetti grafici dell'infografica all'interno di list tramite il suo metodo insertGO
- imposta lo *$scope* per reagire agli eventi *nextslide, previousSlide, enterInCheckPoint, returnToCheckPoint* generati dalla vista per lo scorrimento della presentazione

4 Tracciamento

Il tracciamento requisiti-componenti viene riportato nella sezione apposita della *Specifica Tecnica v3.0*.