

20-08-2015



## Definizione del Prodotto

## Informazioni sul documento

<b>Nome Documento</b>	Definizione del Prodotto
<b>Versione</b>	1.0.0
<b>Stato</b>	<i>Formale</i>
<b>Uso</b>	<i>Esterno</i>
<b>Data Creazione</b>	09-06-2015
<b>Data Ultima Modifica</b>	20-08-2015
<b>Redazione</b>	Fossa Manuel, Venturelli Giovanni, Tollot Pietro, Busetto Matteo
<b>Approvazione</b>	Fossa Manuel
<b>Verifica</b>	Petrucci Mauro
<b>Lista distribuzione</b>	<i>LateButSafe</i> Prof. Tullio Vardanega Prof. Riccardo Cardin Proponente Zucchetti S.p.a.



# Sommario

Il presente documento riporta la Definizione del Prodotto effettuata per il capitolato Premi.

## Registro delle modifiche

Tab 1: Versionamento del documento

Versione	Autore	Data	Descrizione
1.0.0	Fossa Manuel	20-08-2015	Approvazione Documento
0.8.5	Petrucci Mauro	18-08-2015	Modifica del nome del file in "Definizione di Prodotto"
0.8.4	Petrucci Mauro	18-08-2015	Inserimento sezioni relative al tracciamento; aggiornati test; aggiunta consuntivo per la fase di codifica
0.8.3	Busetto Matteo	10-08-2015	Aggiornato Premi::Controller, Premi:: View
0.8.2	Busetto Matteo	07-08-2015	Aggiornato Premi::Controller, Premi:: View
0.8.1	Fossa Manuel	05-08-2015	Aggiornato diagramma Loader
0.8.0	Busetto Matteo	04-08-2015	Aggiornato Controller::goRegistrazione, goHome, AccessController
0.7.11	Fossa Manuel	03-08-2015	Corretto output di NodeServer::PresentationMeta: GET /private/api/presentations
0.7.10	Fossa Manuel	02-08-2015	Corretto serverRelation::MongoRelation, schema di NodeServer, definizione di NodeServer
0.7.9	Busetto Matteo, Petrucci Mauro	31-07-2015	Aggiornata specifica classi del front-end, Premi::App
0.7.8	Petrucci Mauro	28-07-2015	Inserimento services di Angular in Premi::App
0.7.7	Petrucci Mauro	27-07-2015	Aggiunta di contenuti. Inserimento di Premi::Services, modifica di Premi::Controller
0.7.6	Venturelli Giovanni	19-07-2015	Aggiornamento capitolo Premi::Model, metodi di inserimento oggetti della presentazione.

Versione	Autore	Data	Descrizione
0.7.5	Tollot Pietro	14-07-2015	Aggiornamento metodi di serverRelation::Registration, MongoRelation, Loader, FileServerRelation, Authentication
0.7.4	Tollot Pietro	02-07-2015	Aggiornamento node_server::Premi_Server
0.7.3	Tollot Pietro	30-06-2015	Aggiornamento schema del backEndProgettazione
0.7.2	Busetto Matteo	28-06-2015	Aggiornamento Controller::InsertFrame
0.7.1	Tollot Pietro	27-06-2015	Aggiornamento Model::ServerRelations, Model::MongoRelations
0.7.0	Busetto Matteo	24-06-2015	Aggiunta di contenuti. Inserimento del capitolo Package::Premi::Controller
0.5.0	Venturelli Giovanni	20-06-2015	Aggiunta di contenuti. Inserimento del capitolo Package::Premi::Model
0.4.0	Fossa Manuel	15-06-2015	Aggiunta di contenuti. Inserimento del capitolo Package::Premi::View
0.3.0	Tollot Pietro	12-06-2015	Aggiunta di contenuti. Inserimento del capitolo Standard di Progetto
0.2.0	Gabelli Pietro	09-06-2015	Aggiunta di contenuti. Inserimento del capitolo Introduzione e Descrizione generale
0.1.0	Gabelli Pietro	08-06-2015	Stesura dello scheletro del documento

## Storico

$$RP \rightarrow RQ$$

Versione 1..0.0	Nominativo
Redazione	Fossa Manuel, Venturelli Giovanni, Tollot Pietro, Busetto Matteo
Verifica	Petrucci Mauro
Approvazione	Fossa Manuel

Tab 2: Storico ruoli RP  $\rightarrow$  RQ

# Indice

<b>1</b>	<b>Introduzione</b>	<b>9</b>
1.1	Scopo del documento . . . . .	9
1.2	Scopo del Prodotto . . . . .	9
1.3	Glossario . . . . .	9
1.4	Riferimenti . . . . .	9
1.4.1	Normativi . . . . .	9
1.4.2	Informativi . . . . .	9
<b>2</b>	<b>Descrizione generale</b>	<b>10</b>
2.1	funzioni <sub>g</sub> del prodotto . . . . .	10
2.2	Caratteristiche degli utenti . . . . .	10
2.3	Vincoli generali . . . . .	11
<b>3</b>	<b>Standard di progetto</b>	<b>12</b>
3.1	Standard di progettazione architettuale . . . . .	12
3.2	Standard di documentazione del codice . . . . .	12
3.3	Standard di denominazione di entità e relazioni . . . . .	12
3.4	Standard di programmazione . . . . .	12
3.5	Strumenti di lavoro . . . . .	12
<b>4</b>	<b>NodeServer</b>	<b>13</b>
4.1	Risorse <sub>g</sub> e Servizi . . . . .	14
<b>5</b>	<b>Package Premi::Model</b>	<b>20</b>
5.1	Classe SlideShowElements . . . . .	20
5.1.1	Classe Text . . . . .	22
5.1.2	Classe Image . . . . .	22
5.1.3	Classe Frame . . . . .	23
5.1.4	Classe SVG . . . . .	24
5.1.5	Classe Audio . . . . .	24
5.1.6	Classe Video . . . . .	25
5.1.7	Classe Background . . . . .	25
5.2	Classe InsertEditRemove . . . . .	26
5.3	Classe Command . . . . .	33
5.3.1	Classe Invoker . . . . .	33
5.3.2	Classe AbstractCommand . . . . .	34
5.3.2.1	Classe ConcreteTextInsertCommand . . . . .	37
5.3.2.2	Classe ConcreteFrameInsertCommand . . . . .	38
5.3.2.3	Classe ConcreteImageInsertCommand . . . . .	39
5.3.2.4	Classe ConcreteSVGInsertCommand . . . . .	39
5.3.2.5	Classe ConcreteAudioInsertCommand . . . . .	40
5.3.2.6	Classe ConcreteVideoInsertCommand . . . . .	41
5.3.2.7	Classe ConcreteBackgroundInsertCommand . . . . .	41
5.3.2.8	Classe ConcreteTextRemoveCommand . . . . .	43

5.3.2.9	Classe ConcreteFrameRemoveCommand . . . . .	44
5.3.2.10	Classe ConcreteImageRemoveCommand . . . . .	45
5.3.2.11	Classe ConcreteSVGRemoveCommand . . . . .	46
5.3.2.12	Classe ConcreteAudioRemoveCommand . . . . .	47
5.3.2.13	Classe ConcreteVideoRemoveCommand . . . . .	48
5.3.2.14	Classe ConcreteEditPositionCommand . . . . .	49
5.3.2.15	Classe ConcreteEditRotationCommand . . . . .	50
5.3.2.16	Classe ConcreteEditSizeCommand . . . . .	51
5.3.2.17	Classe ConcreteEditContentCommand . . . . .	53
5.3.2.18	Classe ConcreteEditBackgroundCommand . . . . .	54
5.3.2.19	Classe ConcreteEditColorCommand . . . . .	55
5.3.2.20	Classe ConcreteEditFontCommand . . . . .	56
5.3.2.21	Classe ConcreteEditBookmarkCommand . . . . .	58
5.3.2.22	Classe ConcretePortaAvantiCommand . . . . .	59
5.3.2.23	Classe ConcretePortaDietroCommand . . . . .	60
5.3.2.24	Classe ConcreteAddToMainPathCommand . . . . .	61
5.3.2.25	Classe ConcreteRemoveFromMainPathCommand . . . . .	62
5.3.2.26	Classe ConcreteNewChoicePathCommand . . . . .	63
5.3.2.27	Classe ConcreteNewChoicePathCommand . . . . .	64
5.3.2.28	Classe ConcreteDeleteChoicePathCommand . . . . .	65
5.3.2.29	Classe ConcreteRemoveFromChoicePathCommand . . . . .	65
5.3.2.30	Classe ConcreteAddToChoicePathCommand . . . . .	66
5.3.2.31	Classe ConcreteAddToMainPathCommand . . . . .	67
5.3.2.32	Classe ConcreteRemoveFromMainPathCommand . . . . .	68
5.4	serverRelation . . . . .	69
5.4.1	Loader . . . . .	69
5.4.2	Registration . . . . .	71
5.4.3	Authentication . . . . .	71
5.4.4	MongoRelation . . . . .	73
5.4.5	fileServerRelation . . . . .	76
<b>6</b>	<b>Package Premi::Controller</b>	<b>78</b>
6.1	Controller::premiApp . . . . .	78
6.2	Controller::Services . . . . .	79
6.2.1	Services::Main . . . . .	79
6.2.2	Services::Upload . . . . .	81
6.2.3	Services::Utils . . . . .	82
6.2.4	Services::SharedData . . . . .	83
6.2.5	Services::toPages . . . . .	84
6.2.6	Controller::HeaderController . . . . .	85
6.2.7	Controller::AccessController . . . . .	87
6.2.8	Controller::HomeController . . . . .	89
6.2.9	Controller::ProfileController . . . . .	91
6.2.10	Controller::ExecutionController . . . . .	92
6.2.11	Controller::EditController . . . . .	93

<b>7</b>	<b>Package Premi::View</b>	<b>106</b>
7.1	View::Pages . . . . .	106
7.2	View::Pages::Index . . . . .	106
7.3	View::Pages::Login . . . . .	107
7.4	View::Pages::Registrazione . . . . .	107
7.5	View::Pages::Home . . . . .	108
7.6	View::Pages::Profile . . . . .	108
7.7	View::Pages::Execution . . . . .	109
7.8	View::Pages::Edit . . . . .	109
<b>8</b>	<b>Tracciamento</b>	<b>114</b>
8.1	tracciamento metodi-test . . . . .	114





1	Servizi RESTfull offerti dal Server <sub>g</sub> nodeJs . . . . .	13
2	Diagramma classe Model::serverRelation::loader::Loader . . . . .	69
3	Diagramma classe Model::serverRelation::accessControll::Registration . . . . .	71
4	Diagramma classe Model::serverRelation::accessControll::Authentication . . . . .	71
5	Diagramma classe Model::serverRelation::mongoRelation::MongoRelation . . . . .	73
6	Diagramma classe Model::serverRelation::fileServerRelation::FileServerRelation . . . . .	76

1	Versionamento del documento . . . . .	2
2	Storico ruoli RP -> RQ . . . . .	4
3	Metodi-Test Controller . . . . .	116

# 1 Introduzione

## 1.1 Scopo del documento

Il presente documento descrive la progettazione di dettaglio definita per il Progetto<sub>g</sub> Premi. Il documento si basa sulla [SpecificaTecnica\\_v.3.0.0.pdf](#). I programmatori si serviranno di tale documento per procedere con le attività di Codifica<sub>g</sub>.

## 1.2 Scopo del Prodotto

Lo scopo del Progetto<sub>g</sub> è la realizzazione un Software<sub>g</sub> per la creazione ed esecuzione di presentazioni multimediali favorendo l'uso di tecniche di storytelling e visualizzazione non lineare dei contenuti.

### 1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite sono riportate nel documento [Glossario\\_v.3.0.0.pdf](#). Ogni occorrenza di vocaboli presenti nel Glossario è marcata da una “g” minuscola in pedice.

## 1.4 Riferimenti

### 1.4.1 Normativi

- Regole del Progetto<sub>g</sub> didattico, reperibili all'Indirizzo<sub>g</sub>:  
<http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/PD01.pdf>
- Vincoli di organigramma, consultabili all'Indirizzo<sub>g</sub>:  
<http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/PD01b.html>
- Norme di Progetto<sub>g</sub>: [NormeDiProgetto\\_v.3.0.0.pdf](#);
- Specifica Tecnica: [SpecificaTecnica\\_v.3.0.0.pdf](#);
- Capitolato d'appalto C4: Premi: Software<sub>g</sub> 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<sub>g</sub> modulo A:
  - Il ciclo di vita<sub>g</sub> del Software<sub>g</sub>;
  - Gestione di Progetto<sub>g</sub>.

<http://www.math.unipd.it/~tullio/IS-1/2014/> ;

- Ingegneria del Software<sub>g</sub> - Ian Sommerville - 9a Edizione (2010).

## 2 Descrizione generale

Il sistema si pone come obiettivo quello di permettere la creazione di presentazioni efficaci dal punto di vista dello storytelling anche ad utenti non esperti.

L'applicazione Premi permette all'utente di creare ed eseguire presentazioni personalizzate. Attraverso la creazione di  $\text{Frame}_g$  e la loro modellazione, l'utente potrà definire un  $\text{Percorso}_g$  di presentazione lineare oppure più percorsi $_g$  che prevedono la possibilità di scegliere con quale continuare il flusso di esecuzione. Questo significa che il  $\text{Percorso}_g$  di  $\text{Frame}_g$  che sarà visualizzato sarà scelto dal presentatore in fase di visualizzazione.

L'applicazione è strutturata in modo gerarchico, ossia ogni  $\text{Frame}_g$  ha almeno un padre (tranne la radice che può avere solamente figli). Questo permette di creare presentazioni strutturate a livelli, rendendo molto semplice la possibilità, durante l'esecuzione, di saltare determinati rami della presentazione. Inoltre, l'utente potrà inserire  $\text{Bookmark}_g$  i quali permettono di saltare ad un  $\text{Frame}_\sigma$  padre in modo semplice e veloce.

Il sistema permetterà di creare e modificare presentazioni se connessi alla rete mentre l'utente potrà eseguire le proprie presentazioni anche offline a patto di averle precedentemente scaricate dal Server<sub>g</sub>. Il sistema sarà implementato utilizzando tecnologie WEB<sub>g</sub> che lo renderanno altamente portatile.

## 2.1 funzioni<sub>g</sub> del prodotto

Il prodotto offre un'interfaccia WEB<sub>g</sub> che permetterà di:

- Registrarsi, accedere al proprio Account<sub>g</sub> ed effettuare il Logout<sub>g</sub>;
- Gestire il proprio Account<sub>g</sub>;
- Creare una nuova presentazione da dispositivo Desktop<sub>g</sub>;
- Modificare una presentazione da dispositivo Desktop<sub>g</sub>;
- Modificare parzialmente la presentazione da dispositivo mobile<sub>g</sub>;
- Eseguire una presentazione salvata sul proprio Account<sub>g</sub>;
- Eseguire una presentazione locale;
- Creare Infografiche<sub>g</sub> a partire da una presentazione;
- Modificare Infografiche<sub>g</sub> create;
- Gestire il proprio archivio di File<sub>g</sub> media;
- Scaricare una presentazione in locale.

## 2.2 Caratteristiche degli utenti

Il prodotto si rivolge a qualsiasi tipo di utente interessato ad una facile creazione e modellazione di presentazioni ed Infografiche. Non emergono quindi restrizioni particolari riguardo le caratteristiche dell'utenza.



Il prodotto non richiede particolari Requisiti<sub>g</sub> hardware anche se gli stessi possono influenzarne la velocità di esecuzione.

### 3 Standard di progetto

### 3.1 Standard di progettazione architettuale

Gli standard di progettazione architettuale sono definiti nel documento [SpecificaTecnica v.3.0.0.pdf](#).

### 3.2 Standard di documentazione del codice

Gli standard per la scrittura di documentazione del Codice<sub>g</sub> sono definiti nelle [NormeDiProgetto v.3.0.0.pdf](#)

### 3.3 Standard di denominazione di entità e relazioni

Tutti gli elementi<sub>g</sub> (package, classi, metodi o attributi) definiti, devono avere denominazioni chiare ed autoesplicative. Nel caso il nome risulti lungo, è preferibile preferire la chiarezza alla lunghezza.

Sono ammesse abbreviazioni se:

- immediatamente comprensibili;
- non ambigue;
- sufficientemente contestualizzate.

Le regole tipografiche relative ai nomi delle entità sono definite nelle [NormeDiProgetto v.3.0.0.pdf](#).

### 3.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nelle [NormeDiProgetto v.3.0.0.pdf](#).

### 3.5 Strumenti di lavoro

Gli strumenti da adottare e le procedure per utilizzarli correttamente durante la realizzazione del prodotto Software<sub>g</sub> sono definiti nelle [NormeDiProgetto\\_v.3.0.0.pdf](#).

## 4 NodeServer

Il seguente diagramma delle classi è stato esteso con le primitive:

- «**Resource**» : rappresenta una Risorsa<sub>g</sub> associata ad un certo URL<sub>g</sub> a cui sono disponibili dei servizi
- «**Node**» : rappresenta una parte di URL<sub>g</sub> a cui non sono disponibili servizi ma è utile per suddividere quest'ultimi
- «**Server**» : rappresenta la radice dei servizi offerti dal Server<sub>g</sub>
- «**Path**» : indica una aggiunta in coda all' URL<sub>g</sub> attuale per raggiungere una nuova Risorsa<sub>g</sub> o nodo
- «**Middleware**» : indica un middleware, un insieme di funzionalità chiamate ogni qualvolta si accede a Risorse<sub>g</sub> attraversando questo Elemento<sub>g</sub>

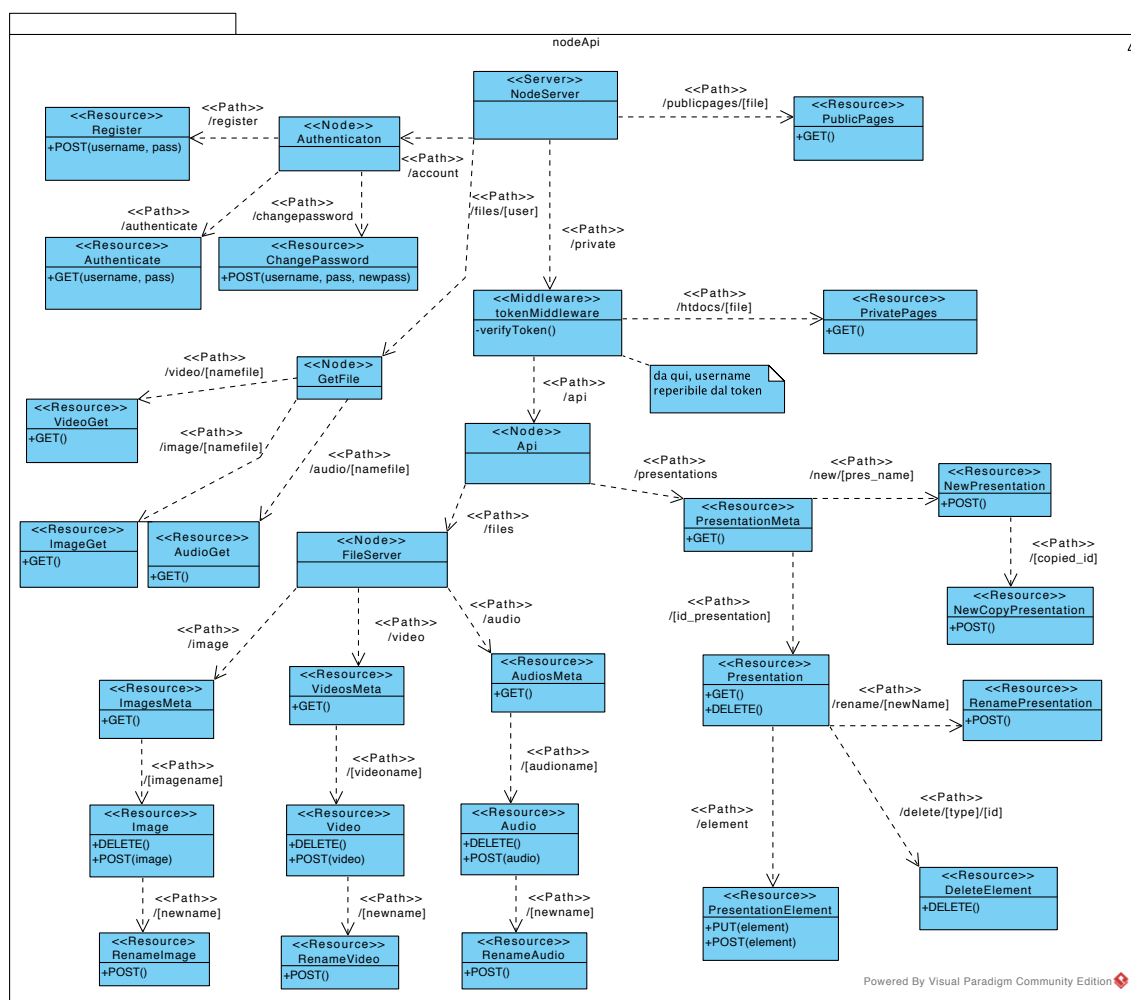


Fig 1: Servizi RESTfull offerti dal Server<sub>g</sub> nodeJs

La gestione degli errori da parte della API REST nodeApi si basa su:

- stato della risposta http del server
- campo dati 'message' nel corpo della risposta
- campo dati 'success' nel corpo della risposta

Gli stati http usati nelle risposte del server sono:

- 500, errore del server
- 400, errore del client
- 404, risorsa non trovata o richiesta con campi dati assenti
- 403, richiesta con campo authorization nell'header non impostato
- 304, nessuna modifica effettuata
- 200, richiesta accettata

Ogni risposta del server con stato diverso da 200 contiene nel campo dati "message" del corpo una descrizione verbosa dell'errore riscontrato. Ogni risposta del server contiene nel corpo un campo dati "success" con valore "true"/"false" a seconda che l'operazione richiesta abbia avuto successo rispetto alla sua definizione nel documento di "Definizione di Prodotto".

## 4.1 Risorse<sub>g</sub> e Servizi

- **NodeServer:** radice dei servizi offerti dal Server<sub>g</sub>:
  1. Server<sub>g</sub> per pagine html e File<sub>g</sub> statici associati
  2. servizi di autenticazione stateless
  3. servizi di upload e reperimento File<sub>g</sub> statici multimediali per utente
  4. servizi di interazione con MongoDB per salvataggio persistente delle presentazioni
- **Register:**
  - **POST** /account/register
    - \* **descrizione:** inserisce nuovo utente in MongoDB, crea una nuova collezione 'presentations'+username, crea le cartelle per i File<sub>g</sub> utente
    - \* **input:** header campo Authorization: "username:password"
    - \* **output:** body in formato application/json, success : boolean
- **Authenticate:**
  - **GET** /account/authenticate
    - \* **descrizione:** verifica se username e password sono corretti e ritorna un token per l'accesso ai servizi protetti
    - \* **input:** header campo Authorization: "username:password"
    - \* **output:** body in formato application/json, success : boolean; in header campo Authorisation ritorna il token per l'accesso ai servizi protetti

- **ChangePassword:**

- **POST** /account/changepassword
  - \* **descrizione:** verifica la correttezza di username e password e modifica questa'ultima con la nuova
  - \* **input:** in header campo Authorization: "username:password:newpassword"
  - \* **output:** body in formato application/json, success : boolean

- **PublicPages:**

- **GET** /publicpages/[file]
  - \* **descrizione:** se presente [file] nella cartella /public\_html del Server<sub>g</sub> ritorna il File<sub>g</sub> stesso
  - \* **input** /
  - \* **output:** fileStatico

- **tokenMiddleware:** verifica che il token passato nel campo Authorization dell' Header sia valido, dal token ricava lo username dell'utente

- PrivatePages:

- **GET** /private/htdocs/[file]
  - \* **descrizione:** se presente [file] nella cartella /private\_html del Server<sub>g</sub> ritorna il File<sub>g</sub> stesso
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** fileStatico

- PresentationMeta:

- **GET** /private/api/presentations
  - \* **descrizione:** cerca in MongoDB nella collezione associata alle presentazioni dell'utente, ritorna un array i cui elementi<sub>g</sub> sono i campi meta delle presentazioni dell'utente
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** body in formato application/json; success : boolean, message : array

- **NewPresentation:**

- **POST** /private/api/presentations/new/[presentationName]
  - \* **descrizione:** se non esiste già crea una nuova presentazione con il nome [presentationName]
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** body in formato application/json; success : boolean

- NewCopyPresentation:

- **POST** /private/api/presentations/new/[newPresentationName]/[oldPresentationName]



- \* **descrizione:** crea una nuova presentazione con nome [newPresentationName] dalla presentazione con titolo [oldPresentationName]
- \* **input:** in header campo Authorization il token di autenticazione
- \* **output:** body in formato application/json; success : boolean

- **Presentation:**

- **GET** /private/api/presentations/[presentationName]
  - \* **descrizione:** recupera se presente la presentazione dell'utente associata al titolo passato nell'URL<sub>g</sub>
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** body in formato application/json; success : boolean, presentation : object
- **DELETE** /private/api/presentations/[presentationName]
  - \* **descrizione:** elimina se presente la presentazione dell'utente associata al titolo passato nell'URL<sub>g</sub>
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** body in formato application/json; success : boolean

- **RenamePresentation:**

- **POST** /private/api/presentations/[presentationName]/rename/[newname]
  - \* **descrizione:** rinomina se presente la presentazione dell'utente associata al titolo passato nell'URL<sub>g</sub> con il nome [newname]
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** body in formato application/json; success : boolean

- **PresentationElement:**

- **POST** /private/api/presentations/[presentationName]/element
  - \* **descrizione:** inserisce nella presentazione dell'utente individuata da [presentationName] l'oggetto element passato nel body della richiesta
  - \* **input:** in header campo Authorization il token di autenticazione, nel body in formato application/json l'oggetto: element : object
  - \* **output:** body in formato application/json; success : boolean
- **PUT** /private/api/presentations/[presentationName]/element
  - \* **descrizione:** sostituisce nella presentazione dell'utente l'Elemento<sub>g</sub> passato nel body della richiesta
  - \* **input:** in header campo Authorization il token di autenticazione, nel body in formato application/json l'oggetto: element : object
  - \* **output:** body in formato application/json; success : boolean

- **DeleteElement:**

- **DELETE** /private/api/presentations/[presentationName]/delete/[type/id element]

- \* **descrizione:** elimina dalla presentazione con il titolo [presentationName] l'Elemento<sub>g</sub> con identificativo [idElement]
- \* **input:** in header campo Authorization il token di autenticazione
- \* **output:** body in formato application/json; success : boolean

- **GetImage:**

- **GET** /files/[user]/image/[imagename]
  - \* **descrizione:** ritorna il File<sub>g</sub> [imagename] nella cartella /users/[username]/image
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** File<sub>g</sub> statico

- **GetAudio:**

- **GET** /files/[user]/audio/[audioame]
  - \* **descrizione:** ritorna il File<sub>g</sub> [audioame] nella cartella /users/[username]/audios
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** File<sub>g</sub> statico

- **GetVideo:**

- **GET** /files/[user]/video/[videoname]
  - \* **descrizione:** ritorna il File<sub>g</sub> [videoname] nella cartella /users/[username]/videos
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** File<sub>g</sub> statico

- ImagesMeta:

- **GET** /private/api/files/image
  - \* **descrizione:** ritorna un array con i nomi dei File<sub>g</sub> immagine dell'utente
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** body in formato application/json; success : boolean, names : array

- Image:

- **POST** /private/api/files/image/[imagename]
  - \* **descrizione:** caricare da locale un nuovo File<sub>g</sub> immagine nella cartella /users/[username]/images
  - \* **input:** in header campo Authorization il token di autenticazione; body in formato "multipart/form-data"; File<sub>g</sub> da caricare
  - \* **output:** body in formato application/json; success : boolean
- **DELETE** /private/api/files/image/[imagename]
  - \* **descrizione:** elimina il File<sub>g</sub> immagine [imagename] dalla cartella /users/[username]/images
  - \* **input:** in header campo Authorization il token di autenticazione;
  - \* **output:** body in formato application/json; success : boolean

- **RenameImage:**

- **POST** /private/api/files/image/[imagename]/[newname]
  - \* **descrizione:** rinomina il File<sub>g</sub> immagine [imagename] con [newname] nella cartella /users/[username]/images
  - \* **input:** in header campo Authorization il token di autenticazione;
  - \* **output:** body in formato application/json; success : boolean

- AudiosMeta:

- **GET** /private/api/files/audio
  - \* **descrizione:** ritorna un array con i nomi dei File<sub>g</sub> audio dell'utente
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** body in formato application/json; success : boolean, names : array

- **Audio:**

- **POST** /private/api/files/audio/[audioname]
  - \* **descrizione:** caricare da locale un nuovo File<sub>g</sub> immagine nella cartella /users/[username]/audios
  - \* **input:** in header campo Authorization il token di autenticazione; body in formato "multipart/form-data"; File<sub>g</sub> da caricare
  - \* **output:** body in formato application/json; success : boolean
- **DELETE** /private/api/files/audio/[audioname]
  - \* **descrizione:** elimina il File<sub>g</sub> audio [audioname] dalla cartella /users/[username]/audios
  - \* **input:** in header campo Authorization il token di autenticazione;
  - \* **output:** body in formato application/json; success : boolean

- **RenameAudio:**

- **POST** /private/api/files/audio/[audioname]/[newname]
  - \* **descrizione:** rinomina il File<sub>g</sub> audio [audioname] con [newname] nella cartella /users/[username]/audios
  - \* **input:** in header campo Authorization il token di autenticazione;
  - \* **output:** body in formato application/json; success : boolean

- VideosMeta:

- **GET** /private/api/files/video
  - \* **descrizione:** ritorna un array con i nomi dei File<sub>g</sub> video dell'utente
  - \* **input:** in header campo Authorization il token di autenticazione
  - \* **output:** body in formato application/json; success : boolean, names : array

- **Video:**

- **POST** /private/api/files/video/[videoname]

- \* **descrizione:** caricare da locale un nuovo File<sub>g</sub> immagine nella cartella /users/[username]/videos
  - \* **input:** in header campo Authorization il token di autenticazione; body in formato "multipart/form-data"; File<sub>g</sub> da caricare
  - \* **output:** body in formato application/json; success : boolean
- **DELETE** /private/api/files/video/[videoname]
- \* **descrizione:** elimina il File<sub>g</sub> video [videoname] dalla cartella /users/[username]/videos
  - \* **input:** in header campo Authorization il token di autenticazione;
  - \* **output:** body in formato application/json; success : boolean

- **RenameVideo:**

- **POST** /private/api/files/video/[videoname]/[newname]
  - \* **descrizione:** rinomina il File<sub>g</sub> video [videoname] con [newname] nella cartella /users/[username]/videos
  - \* **input:** in header campo Authorization il token di autenticazione;
  - \* **output:** body in formato application/json; success : boolean

## 5 Package Premi::Model

Tutti i package seguenti appartengono al package Premi, quindi per ognuno di essi lo scope sarà: Premi::[nome package].

**Tipo, obiettivo e funzione del componente:** classe astratta, base delle classi usate per rappresentare gli elementi della presentazione.

**Relazioni d'uso di altre componenti:** è in relazione con il package Controller e con NodeAPI.

## 5.1 Classe SlideShowElements

## Funzione

Classe astratta, base delle classi usate per rappresentare gli elementi della presentazione.

## Scope

Model::SlideShow::SlideShowElements.

## Utilizzo

Contiene gli attributi e i metodi comuni degli oggetti che rappresentano gli elementi<sub>g</sub> della presentazione.

## Attributi

- **id**
  - **Accesso:** Private;
  - **Tipo:** Integer;
  - **Descrizione:** indica l'identificativo univoco dell'Elemento<sub>g</sub>.
- **yIndex**
  - **Accesso:** Private;
  - **Tipo:** Double;
  - **Descrizione:** rappresenta la posizione sull'asse delle y dell'Elemento<sub>g</sub> rispetto alla presentazione.
- **xIndex**
  - **Accesso:** Private;
  - **Tipo:** Double;
  - **Descrizione:** rappresenta la posizione sull'asse delle x dell'Elemento<sub>g</sub> rispetto alla presentazione.
- **zIndex**
  - **Accesso:** Private;
  - **Tipo:** Integer;



- Ereditata da:

- 
- Università degli studi di Padova - 2014/2015

### 5.1.1 Classe Text

## Funzione

Classe concreta, i suoi elementi, rappresentano un oggetto di tipo testo.

## Scope

```
Model::SlideShow::SlideShowElements::Text
```

## Utilizzo

Il costruttore viene invocato da `Inserter::insertText()`.

## Attributi

- **font**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il  $\text{Font}_g$  dell'oggetto.
- **fontSize**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta la dimensione del  $\text{Font}_g$  dell'oggetto.
- **content**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il contenuto del testo.
- **color**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il colore dell'oggetto.
- **type**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il tipo dell'oggetto.

### 5.1.2 Classe Image

## Funzione

Classe concreta, i suoi elementi, rappresentano un oggetto di tipo immagine.

## Scope

Model::SlideShow::SlideShowElements::Image.

## Utilizzo

Il costruttore viene invocato da `Inserter::insertImage()`.

## Attributi

- **url**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il Percorso<sub>g</sub> dell'oggetto.
- **type**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il tipo dell'oggetto.

### 5.1.3 Classe Frame

## Funzione

Classe concreta, i suoi elementi<sub>g</sub> rappresentano un oggetto di tipo Frame<sub>g</sub>.

## Scope

Model::SlideShow::SlideShowElements::Frame.

## Utilizzo

Il costruttore viene invocato da `Inserter::insertFrame()`.

## Attributi

- **bookmark**
  - **Accesso:** Private;
  - **Tipo:** Bool;
  - **Descrizione:** è a 1 se il  $\text{Frame}_g$  è un  $\text{Bookmark}_g$ , 0 altrimenti.
- **ref**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** contiene il riferimento dell'immagine di sfondo  $\text{Frame}_g$ .
- **color**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il colore dello sfondo del  $\text{Frame}_g$ .
- **type**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il tipo dell'oggetto.



#### 5.1.4 Classe SVG

## Funzione

Classe concreta, i suoi elementi, rappresentano un oggetto di tipo SVG.

## Scope

Model::SlideShow::SlideShowElements::SVG.

## Utilizzo

Il costruttore viene invocato da `Inserter::insertSVG()`.

## Attributi

- **color**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il colore dell'oggetto.
- **shape**
  - **Accesso:** Private;
  - **Tipo:** Array;
  - **Descrizione:** rappresenta le coordinate della forma dell'oggetto.
- **type**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il tipo dell'oggetto.

### 5.1.5 Classe Audio

## Funzione

Classe concreta, i suoi elementi, rappresentano un oggetto di tipo immagina.

## Scope

```
Model::SlideShow::SlideShowElements::Audio.
```

## Utilizzo

Il costruttore viene invocato da `Inserter::insertAudio()`.

## Attributi

- **url**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il Percorso<sub>g</sub> dell'oggetto.
- **type**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il tipo dell'oggetto.

### 5.1.6 Classe Video

## Funzione

Classe concreta, i suoi elementi, rappresentano un oggetto di tipo video.

## Scope

Model::SlideShow::SlideShowElements::Video.

## Utilizzo

Il costruttore viene invocato da `Inserter::insertVideo()`.

## Attributi

- **url**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il Percorso<sub>g</sub> dell’oggetto.
- **type**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il tipo dell’oggetto.

### 5.1.7 Classe Background

## Funzione

Classe concreta, i suoi elementi<sub>g</sub> rappresentano lo sfondo.

## Scope

Model::SlideShow::SlideShowElements::Background.

## Utilizzo

Il costruttore viene invocato da `Inserter::insertBackground()`.

## Attributi

- **image**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il riferimento dell'immagine dello sfondo.
- **color**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il colore dello sfondo.
- **type**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta il tipo dell'oggetto.

## 5.2 Classe InsertEditRemove

## Funzione

Classe statica in cui vengono implementati gli algoritmi di inserimento di elementi<sub>g</sub> nella presentazione.

## Scope

Model::SlideShow::SlideShowActions::InsertEditRemove.

## Utilizzo

Viene utilizzata dalla classe `command` per eseguire i comandi di inserimento.

## Attributi

- **Presentazione**
  - **Accesso:** Private;
  - **Descrizione:** oggetto json che contiene gli oggetti delle classi che rappresentano gli elementi<sub>g</sub> della presentazione.
- **id =0**
  - **Accesso:** Private;
  - **Descrizione:** attributo statico, indica gli id univoci degli oggetti generati.

## Metodi

- **constructPresentazione**(newPresentazione)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** setta Presentazione a newPresentazione.
- **getPresentazione**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** ritorna Presentazione.
- **getPresentationName**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** ritorna il nome di Presentazione.
- **getIdPresentazione**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** ritorna l'id di Presentazione.



- **Descrizione:** ritorna i percorsi<sub>g</sub> della presentazione.
- **getElement(id)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** ritorna l'Elemento<sub>g</sub> id della presentazione.
- **insertText(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce un oggetto di tipo Text() passando il parametro spec. Inserisce l'oggetto così costruito nell'oggetto Presentazione.
- **insertFrame(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Integer;
  - **Descrizione:** costruisce un oggetto di tipo Frame() passando il parametro spec. Inserisce l'oggetto così costruito nell'oggetto Presentazione.
- **insertImage(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Integer;
  - **Descrizione:** costruisce un oggetto di tipo Image() passando il parametro spec. Inserisce l'oggetto così costruito nell'oggetto Presentazione.
- **insertSVG(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Integer;
  - **Descrizione:** costruisce un oggetto di tipo SVG() passando il parametro spec. Inserisce l'oggetto così costruito nell'oggetto Presentazione.
- **insertAudio(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Integer;
  - **Descrizione:** costruisce un oggetto di tipo Audio() passando il parametro spec. Inserisce l'oggetto così costruito nell'oggetto Presentazione.
- **insertVideo(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Integer;

- **Descrizione:** costruisce un oggetto di tipo Video() passando il parametro spec. Inserisce l'oggetto così costruito nell'oggetto Presentazione.
- **insertBackground(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Integer;
  - **Descrizione:** costruisce un oggetto di tipo Background() passando il parametro spec e salvandosi in un oggetto oldBackground il background precedente. Imposta il nuovo background alla presentazione e ritorna oldBackground.
- **removeText(id:integer)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Text;
  - **Descrizione:** copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.
- **removeFrame(id:integer)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Frame<sub>g</sub>;
  - **Descrizione:** copia l'oggetto con il campo dati id corrispondente, accede al suo campo prev e ne identifica i predecessore, pone prev.next=next. Rimuove l'oggetto dall'oggetto Presentazione e restituisce l'oggetto copiato.
- **removeImage(id:integer)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Image;
  - **Descrizione:** copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.
- **removeSVG(id:integer)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** SVG;
  - **Descrizione:** copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.
- **removeAudio(id:integer)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Audio;
  - **Descrizione:** copia l'oggetto con il campo dati id corrispondente e lo rimuove dall'oggetto Presentazione. Restituisce l'oggetto copiato.

- **removeVideo**(id:integer)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Video;
  - **Descrizione:** copia l’oggetto con il campo dati id corrispondente e lo rimuove dall’oggetto Presentazione. Restituisce l’oggetto copiato.
- **editPosition**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo SlideShowElements in Presentazione per trovare l’oggetto con il campo spec.id corrispondente, crea un oggetto oldPosition per contenere i valori di xIndex e di yIndex dell’oggetto trovato e imposta xIndex e yIndex con i valori passati tramite spec. Restituisce oldPosition.
- **editRotation**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo SlideShowElements in Presentazione per trovare l’oggetto con il campo spec.id corrispondente, crea un oggetto oldRotation per contenere il valore di rotation dell’oggetto trovato e imposta rotation con il valore passato tramite spec. Restituisce oldRotation.
- **editSize**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo SlideShowElements in Presentazione per trovare l’oggetto con il campo spec.id corrispondente, crea un oggetto oldSize per contenere i valori di height e di width dell’oggetto trovato e imposta height e width con i valori passati tramite spec. Restituisce oldSize.
- **editBackground**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo SlideShowElements in Presentazione per trovare l’oggetto con il campo spec.id corrispondente. Se lo trova crea un oggetto oldBackground per salvare i valori di color e ref dell’oggetto trovato e imposta color e ref con i valori passati tramite spec. Restituisce oldBackground.
- **editColor**(spec:object)
  - **Accesso:** Public;

- **Tipo di ritorno:** Object;
- **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo `SlideShowElements` in `Presentazione` per trovare l’oggetto con il campo `spec.id` corrispondente, crea un oggetto `oldColor` per contenere il valore di `color` dell’oggetto trovato e imposta `color` con il valore passato tramite `spec`. Restituisce `oldColor`.
- **editShape(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo `SlideShowElements` in `Presentazione` per trovare l’oggetto con il campo `spec.id` corrispondente, crea un oggetto `oldShape` per contenere il valore di `shape` dell’oggetto trovato e imposta `shape` con il valore passato tramite `spec`. Restituisce `oldShape`.
- **editContent(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo `SlideShowElements::text` in `Presentazione` per trovare l’oggetto con il campo `spec.id` corrispondente. Se lo trova copia il campo `content` in un oggetto `oldContent` e imposta `content` con il valore passato tramite `spec`. Restituisce `oldContent`.
- **editFont(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo `SlideShowElements::text` in `Presentazione` per trovare l’oggetto con il campo `spec.id` corrispondente. Se lo trova copia il campo `Fontg` e `fontSize` in un oggetto `oldFont` e imposta `Fontg` con il valore di `newFont` e `fontSize` con i valori passati tramite `spec`. Restituisce `oldFont`.
- **portaAvanti(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo `SlideShowElements` in `Presentazione` per trovare l’oggetto con il campo `spec.id` corrispondente. Se lo trova aumenta il valore di `zIndex`.
- **portaDietro(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;



- **Descrizione:** scorre il campo dati che contiene gli oggetti di tipo `SlideShowElements` in `Presentazione` per trovare l'oggetto con il campo `spec.id` corrispondente. Se lo trova diminuisce il valore di `zIndex`.
- **addFrameToMainPath**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** aggiunge il `Frameg` `spec.id` nel `Percorsog` principale nella posizione `spec.pos`.
- **removeFrameFromMainPath**(id:integer)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** scorre il `Percorsog` principale in `Presentazione` per trovare l'oggetto con il campo `id` corrispondente. Se lo trova copia la posizione in un oggetto `oldFrame` e elimina dal `Percorsog` il `Frameg`. Restituisce `oldFrame`.
- **addChoicePath**(id)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Integer;
  - **Descrizione:** metodo che aggiunge un nuovo `Percorsog` a scelta `id` a `Presentazione` e ritorna l'id del `Percorsog`.
- **deleteChoicePath**(pathId)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che elimina il `Percorsog` a scelta `pathId` da `Presentazione` e ritorna l'oggetto eliminato.
- **addFrameToChoicePath**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** aggiunge il `Frameg` `spec.id` al `Percorsog` a scelta `spec.pathId`.
- **removeFrameFromChoicePath**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** scorre il `Percorsog` a scelta `spec.pathId` in `Presentazione` per trovare l'oggetto con il campo `spec.id` corrispondente. Se lo trova copia posizione, `id` e `id` del `Percorsog` in un oggetto `obj` e elimina dal `Percorsog` il `Frameg`. Restituisce `obj`.

### 5.3 Classe Command

### 5.3.1 Classe Invoker

## Funzione

Classe, componente invoker del Design Pattern Command. Implementato tramite design pattern singleton.

## Scope

```
Model::SlideShow::SlideShowActions::Command::Invoker.
```

## Utilizzo

Viene utilizzata per eseguire i comandi e memorizzarli all'interno di stack dedicate all'implementazione delle funzionalità di annulla e ripristina.

## Attributi

- **undoStack**
  - **Accesso:** Private;
  - **Tipo:** Array;
  - **Descrizione:** contiene l'elenco dei comandi eseguiti e annullabili.
- **redoStack**
  - **Accesso:** Private;
  - **Tipo:** Array;
  - **Descrizione:** contiene l'elenco dei comandi annullati e ripristinabili.

## Metodi

- **execute**(AbstractCommand)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo doAction() del comando ricevuto e invoca undoStack.push(AbstractCommand) e redoStack.clear().
- **undo**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo undoAction() dell'ultimo comando in undoStack() e invoca command=undoStack.pop() e redoStack.push(command).
- **redo**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo doAction() dell'ultimo comando in redoStack() e invoca command=redoStack.pop() e undoStack.push(command).

- **getUndoStack()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Boolean;
  - **Descrizione:** ritorna true se undoStack non è vuoto, false altrimenti.
- **getRedoStack()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Boolean;
  - **Descrizione:** ritorna true se redoStack non è vuoto, false altrimenti.

### 5.3.2 Classe AbstractCommand

## Funzione

Classe concreta, i suoi elementi, rappresentano un oggetto di tipo testo.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand.
```

## Utilizzo

È classe base per i comandi di modifica, inserimento ed eliminazione degli elementi<sub>g</sub> della presentazione.

## Attributi

- **id**
  - **Accesso:** Private;
  - **Tipo:** Integer;
  - **Descrizione:** indica il Codice<sub>g</sub> identificativo dell’oggetto su cui viene eseguito il comando.
- **executed**
  - **Accesso:** Private;
  - **Tipo:** Boolean;
  - **Descrizione:** è settata a false di default, indica se il comando è stato eseguito.
- **enabler**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** corrisponde all’oggetto InsertEditRemove.
- **obj**
  - **Accesso:** Private;
  - **Tipo:** Object;



- **Descrizione:** mantiene salvati id e tipo dell'Elemento<sub>g</sub> su cui abstractCommand è stato richiamato.

## Metodi

- **AbstractCommand(spec:object)**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** costruisce l'oggetto abstractCommand e setta:

- \* id=spec.id;
  - \* obj.type=spec.type;
  - \* obj.id=spec.id;
  - \* executed=0;
  - \* enabler=insertEditRemove();

- **getObj()**

- **Accesso:** Public;
- **Tipo di ritorno:** Object;
- **Descrizione:** metodo che ritorna obj.

- **setId(id)**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che cambia l'id con il nuovo id passato per parametro.

- **setExecuted()**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che setta executed a true.

- **getExecuted()**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che ritorna executed.

- **getId()**

- **Accesso:** Public;
- **Tipo di ritorno:** Integer;
- **Descrizione:** metodo che ritorna l'id dell'Elemento<sub>g</sub>.

- **getPosition()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che ritorna le posizioni xIndex e yIndex dell'Elemento<sub>g</sub>.
- **getRotation()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che ritorna la rotazione dell'Elemento<sub>g</sub>.
- **getSize()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che ritorna le dimensioni dell'Elemento<sub>g</sub>.
- **getEnabler()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che ritorna enabler.

Ereditata da:

- `ConcreteTextInsertCommand` (§5.3.2.1);
- `ConcreteFrameInsertCommand` (§5.3.2.2);
- `ConcreteImageInsertCommand` (§5.3.2.3);
- `ConcreteSVGInsertCommand` (§5.3.2.4);
- `ConcreteAudioInsertCommand` (§5.3.2.5);
- `ConcreteVideoInsertCommand` (§5.3.2.6);
- `ConcreteBackgroundInsertCommand` (§5.3.2.7);
- `ConcreteTextRemoveCommand` (§5.3.2.8);
- `ConcreteFrameRemoveCommand` (§5.3.2.9);
- `ConcreteImageRemoveCommand` (§5.3.2.10);
- `ConcreteSVGRemoveCommand` (§5.3.2.11);
- `ConcreteAudioRemoveCommand` (§5.3.2.12);
- `ConcreteVideoRemoveCommand` (§5.3.2.13);

- ConcreteEditSizeCommand (§5.3.2.16);
- ConcreteEditPositionCommand (§5.3.2.14);
- ConcreteEditRotationCommand (§5.3.2.15);
- ConcreteEditColorCommand (§5.3.2.19);
- ConcreteEditBackgroundCommand (§5.3.2.18);
- ConcreteEditFontCommand (§5.3.2.20);
- ConcreteEditBookmarkCommand (§5.3.2.21);
- ConcretePortaAvantiCommand (§5.3.2.22);
- ConcretePortaDietroCommand (§5.3.2.23);
- ConcreteAddToMainPathCommand (§5.3.2.31);
- ConcreteRemoveFromMainPathCommand (§5.3.2.32);
- ConcreteNewChoicePathCommand (§5.3.2.27);
- ConcreteDeleteChoicePathCommand (§5.3.2.28);
- ConcreteAddToChoicePathCommand (§5.3.2.30);
- ConcreteRemoveFromChoicePathCommand (§5.3.2.29).

#### 5.3.2.1 Classe ConcreteTextInsertCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteTextInsertCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve le coordinate di inserimento di un testo nella presentazione e invoca il metodo di `Insert` `insertText()` passandogliele.

## Metodi

- **ConcreteTextInsertCommand**(spec: object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteTextInsertCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo:** Void;

- **Descrizione:** invoca il metodo di `Inserter insertText(spec)`. Se `executed` è settato a `false` lo setta come `true`, altrimenti invoca il metodo `inserisciTesto(spec)` di `EditController`.
- **undoAction()**
  - **Accesso:** `Public`;
  - **Tipo:** `Void`;
  - **Descrizione:** invoca il metodo di `Editor_g removeText(id)` passando come parametro il campo `id`. Invoca il metodo `rimuoviElemento(spec)` di `EditController`.

### 5.3.2.2 Classe ConcreteFrameInsertCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteFrameInsertCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve le coordinate di inserimento di un testo nella presentazione e invoca il metodo di `Insert` `insertFrame()` passandoglielo.

## Metodi

- **ConcreteFrameInsertCommand**(spec: object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteFrameInsertCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Inserter insertFrame(spec). Se executed è set-  
tato a false lo setta come true, altrimenti invoca il metodo inserisciFrame(spec) di  
EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Editor<sub>g</sub> removeFrame(id) passando come parame-  
tro il campo id. Invoca il metodo rimuoviElemento(spec) di EditController.

### 5.3.2.3 Classe ConcreteImageInsertCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteImageInsertCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve le coordinate di inserimento di un'immagine nella presentazione e invoca il metodo di `Inserter` `insertImage()` passandogliele.

## Metodi

- **ConcreteImageInsertCommand**(spec: object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteImageInsertCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Inserter insertImage(spec). Se executed è set-  
tato a false lo setta come true, altrimenti invoca il metodo inserisciImmagine() di  
EditController passandogli spec.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Editor<sub>g</sub> removeImage(id) passando come parame-  
tro il campo id. Invoca il metodo rimuoviElemento(spec) di EditController.

#### 5.3.2.4 Classe ConcreteSVGInsertCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteSVGInsertCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve le coordinate di inserimento di un SVG nella presentazione e invoca il metodo di `Inserter` `insertSVG()` passandogliele.

## Metodi

- **ConcreteSVGInsertCommand**(spec: object)
  - **Accesso**: Public;



- **Tipo di ritorno:** Void;
- **Descrizione:** costruisce l'oggetto ConcreteSVGInsertCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Inserter insertSVG(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Editor<sub>g</sub> removeSVG(id) passando come parametro il campo id. Invoca il metodo remove(id) di EditController.

### 5.3.2.5 Classe ConcreteAudioInsertCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteAudioInsertCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve le coordinate di inserimento di un audio nella presentazione e invoca il metodo di `Inserter insertAudio()` passandogliele.

## Metodi

- **ConcreteAudioInsertCommand**(spec: object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteAudioInsertCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Inserter insertAudio(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo inserisciAudio() di EditController passandogli spec.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Editor<sub>g</sub> removeAudio(id) passando come parametro il campo id. Invoca il metodo rimuoviElemento(spec) di EditController.

#### 5.3.2.6 Classe ConcreteVideoInsertCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteVideoInsertCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve le coordinate di inserimento di un video nella presentazione e invoca il metodo di `Inserter insertVideo()` passandoglielo..

## Metodi

- **ConcreteVideoInsertCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteVideoInsertCommand e setta xIndex, yIndex, rotation, ref.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Inserter insertVideo(spec). Se executed è settato a false lo setta come true, altrimenti invoca il metodo inserisciVideo() di EditController passandogli spec.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Editor<sub>g</sub> removeVideo(id) passando come parametro il campo id. Invoca il metodo rimuoviElemento(spec) di EditController.

### 5.3.2.7 Classe ConcreteBackgroundInsertCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteBackgroundInsertComm
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve i parametri di inserimento di uno sfondo nella presentazione e invoca il metodo di `Insert` `insertBackground()` passandoglielo.

## Attributi

- ref

- **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** rappresenta l'URL<sub>g</sub> dell'oggetto inserito.
- **color**
    - **Accesso:** Private;
    - **Tipo:** String;
    - **Descrizione:** rappresenta il colore dello sfondo inserito.
- **oldBackground**
    - **Accesso:** Private;
    - **Tipo:** SlideShowElements::Background;
    - **Descrizione:** rappresenta il vecchio sfondo della presentazione.

## Metodi

- **ConcreteBackgroundInsertCommand**(ref: string, color:string)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteVideoInsertCommand e setta ref e color.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di Inserter insertBackground (ref, color) passando come parametri i campi dati ref e color. insertBackground restituisce un numero intero che rappresenta l'id dell'oggetto o eventualmente un oggetto di tipo Background a cui ConcreteBackgroundInsertCommand istanzia oldBackground e al cui id inizializza il campo id. Se executed è settato a false lo setta come true, altrimenti invoca il metodo updateSfondo(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** viene invocato il metodo InsertEditRemove::insertBackground(SlideShowElement) e il metodo updateSfondo(oldBackground) di EditController.



### 5.3.2.8 Classe ConcreteTextRemoveCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteTextRemoveCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> testo da rimuovere dalla presentazione e invoca il metodo di `Remover` `removeText(id)`.

## Attributi

- **oldText**
  - **Accesso:** Private;
  - **Tipo:** SlideShowElements::Text;
  - **Descrizione:** è una copia dell'Elemento<sub>o</sub> testo rimosso.

## Metodi

- **ConcreteTextRemoveCommand**(spec: object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteTextRemoveCommand e setta l'attributo id.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeText(id) passando come parametro l'id dell'Elemento<sub>g</sub>. removeText restituisce un oggetto di tipo SlideShowElements::Text a cui viene inizializzato oldText. Se executed è settato a false lo setta come true, altrimenti invoca il metodo rimuoviElemento(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::insertText(Text) passando come parametro text. Invoca il metodo inserisciTesto(oldText) di EditController.



### 5.3.2.9 Classe ConcreteFrameRemoveCommand

#### Funzione

Classe concreta, è interfaccia del Design Pattern Command.

#### Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteFrameRemoveCommand

#### Utilizzo

Viene costruito da Premi::Controller::EditController, riceve l'id del Frame<sub>g</sub> da rimuovere dalla presentazione e invoca il metodo InsertEditRemove::removeFrame(id).

#### Attributi

- **oldFrame**
  - **Accesso:** Private;
  - **Tipo:** SlideShowElements::Frame;
  - **Descrizione:** è una copia dell'oggetto rimosso.

#### Metodi

- **ConcreteFrameRemoveCommand(spec: object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteFrameRemoveCommand e setta il campo dati id.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeFrame(id) passando come parametro il campo dati id. removeFrame restituisce una copia dell'oggetto rimosso che verrà settata come campo dati oldFrame. Se executed è settato a false lo setta come true, altrimenti invoca il metodo rimuoviElemento(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::insertFrame(Frame<sub>g</sub>) passando come parametro il campo oldFrame. Invoca il metodo inserisciFrame(oldFrame) di EditController.

### 5.3.2.10 Classe ConcreteImageRemoveCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteImageRemoveCommand
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'immagine da rimuovere dalla presentazione e invoca il metodo `InsertEditRemove::removeImage(id)`.

## Attributi

- **oldImage**
  - **Accesso:** Private;
  - **Tipo:** SlideShowElements::Image;
  - **Descrizione:** è una copia dell'oggetto rimosso

## Metodi

- **ConcreteImageRemoveCommand**(spec: object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteImageRemoveCommand e setta il campo id.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeImage(id) passando come parametro l'id dell'oggetto da rimuovere. removeImage restituisce un oggetto di tipo SlideShowElements::Image cui sarà settato il campo oldImage. Se executed è settato a false lo setta come true, altrimenti invoca il metodo rimuoviElemento(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::insertImage(Image) passando come parametro il campo oldImage. Invoca il metodo inserisciImmagine() di EditController passandogli oldImage.

#### 5.3.2.11 Classe ConcreteSVGRemoveCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteSVGRemoveCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> SVG da rimuovere dalla presentazione e invoca il metodo `InsertEditRemove::removeSVG()`.

## Attributi

- **oldSVG**
  - **Accesso:** Private;
  - **Tipo:** SlideShowElements::SVG;
  - **Descrizione:** è una copia dell’oggetto che rappresenta l’Elemento<sub>o</sub> rimosso.

## Metodi

- **ConcreteSVGRemoveCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteSVGRemoveCommand e setta il campo dati id.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeSVG(id) passando come parametro il campo dati id. removeSVG restituisce un Elemento<sub>g</sub> di tipo SlideShowElements::SVG che rappresenta l'Elemento<sub>g</sub> rimosso e a cui viene inizializzato il campo oldSVG. Se executed è settato a false lo setta come true, altrimenti invoca il metodo update(id) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo di InsertEditRemove::insertSVG(SVG) passando come parametro il campo oldSVG. Invoca il metodo update(id) di EditController.

### 5.3.2.12 Classe ConcreteAudioRemoveCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteAudioRemoveCommand

## Utilizzo

Viene costruito da `Premii::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> audio da rimuovere dalla presentazione e invoca il metodo `InsertEditRemove::removeAudio(id)`.

## Attributi

- **oldAudio**
  - **Accesso**: Private;
  - **Tipo**: SlideShowElements::Audio;
  - **Descrizione**: è una copia dell’oggetto che rappresenta l’Elemento<sub>σ</sub> rimosso.

## Metodi

- **ConcreteAudioRemoveCommand(id:integer)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteAudioRemoveCommand e setta il campo dati id.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeAudio(id) passando come parametro il campo dati id. removeAudio restituisce una copia dell'oggetto Audio rimosso. Se executed è settato a false lo setta come true, altrimenti invoca il metodo rimuoviElemento(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::insertAudio(Audio) passando come parametro il campo oldAudio. Invoca il metodo inserisciAudio() di EditController passandogli oldAudio.





### 5.3.2.13 Classe ConcreteVideoRemoveCommand

#### Funzione

Classe concreta, è interfaccia del Design Pattern Command.

#### Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteVideoRemoveCommand

#### Utilizzo

Viene costruito da Premi::Controller::EditController, riceve l'id dell'Elemento<sub>g</sub> video da rimuovere dalla presentazione e invoca il metodo InsertEditRemove::removeVideo(id).

#### Attributi

- **oldVideo**
  - **Accesso:** Private;
  - **Tipo:** SlideShowElements::Video;
  - **Descrizione:** è una copia dell'oggetto che rappresenta l'Elemento<sub>g</sub> rimosso.

#### Metodi

- **ConcreteVideoRemoveCommand(id:integer)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteVideoRemoveCommand e setta il campo dati id.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeVideo(id) passando come parametro il campo dati id. removeVideo restituisce una copia dell'oggetto Video rimosso. Se executed è settato a false lo setta come true, altrimenti invoca il metodo rimuoviElemento(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::insertVideo(Video) passando come parametro il campo oldVideo. Invoca il metodo inserisciVideo() di EditController passandogli oldVideo.



#### 5.3.2.14 Classe ConcreteEditPositionCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditPositionCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione e le coordinate x e y in cui deve essere spostato l'Elemento<sub>g</sub>, invoca il metodo `InsertEditRemove::editPosition(spec)`.

## Attributi

- oldPosition

- **Accesso:** Private;
- **Tipo:** Oggetto;
- **Descrizione:** oggetto che contiene i parametri di posizione dell'Elemento<sub>g</sub> modificato, contiene al suo interno i campi:
  - \* **id**
    - **Accesso:** Private;
    - **Tipo:** Int;
    - **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
  - \* **tipo**
    - **Accesso:** Private;
    - **Tipo:** String;
    - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.
  - \* **xIndex**
    - **Accesso:** Private;
    - **Tipo:** Double;
    - **Descrizione:** indica la vecchia posizione sull'asse x dell'Elemento<sub>g</sub>.
  - \* **yIndex**
    - **Accesso:** Private;
    - **Tipo:** Double;
    - **Descrizione:** indica la vecchia posizione sull'asse y dell'Elemento<sub>g</sub>.

## Metodi

- **ConcreteEditPositionCommand(spec:object)**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** costruisce l'oggetto ConcreteEditPositionCommand.

- **doAction()**

- **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo `InsertEditRemove::editPosition(spec)` passando come parametro l'oggetto `spec`. `editPosition` ritorna un oggetto a cui `ConcreteEditPositionCommand` inizializza `oldPosition`. Se `executed` è settato a `false` lo setta come `true`, altrimenti invoca il metodo `muoviElemento(spec)` di `EditController`.
- **undoAction()**
    - **Accesso:** Public;
    - **Tipo di ritorno:** Void;
    - **Descrizione:** invoca il metodo `InsertEditRemove::editPosition(spec)` passando come parametro il campo `dati oldPosition`. Invoca il metodo `muoviElemento(oldPosition)` di `EditController`.

### 5.3.2.15 Classe ConcreteEditRotationCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditRotationCommand
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione e il grado a cui deve essere ruotato l'Elemento<sub>g</sub>, invoca il metodo `InsertEditRemove::editRotation(spec)`.

## Attributi

- **oldRotation**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che contiene i parametri di rotazione dell'Elemento<sub>g</sub> modificato, contiene al suo interno i campi:
    - \* **id**
      - **Accesso:** Private;
      - **Tipo:** Int;
      - **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
    - \* **tipo**
      - **Accesso:** Private;
      - **Tipo:** String;
      - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.
    - \* **rotation**
      - **Accesso:** Private;
      - **Tipo:** Double;

- **Descrizione:** indica il vecchio grado di rotazione dell'Elemento<sub>g</sub>.

## Metodi

- **ConcreteEditRotationCommand(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteEditRotationCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editRotation(spec) passando come parametro l'oggetto spec. editRotation ritorna un oggetto a cui ConcreteEditRotationCommand inizializza oldRotation. Se executed è settato a false lo setta come true, altrimenti invoca il metodo ruotaElemento(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editRotation(spec) passando come parametro il campo dati oldRotation. Invoca il metodo ruotaElemento(oldRotation) di EditController.

#### 5.3.2.16 Classe ConcreteEditSizeCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditSizeCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione e le dimensioni a cui deve essere ridimensionato l'Elemento<sub>g</sub>, invoca il metodo `InsertEditRemove::editSize(spec)`.

## Attributi

- **oldSize**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che contiene i parametri di dimensione dell'Elemento<sub>g</sub> modificato, contiene al suo interno i campi:
    - \* **id**
    - **Accesso:** Private;

- **Tipo:** Int;
- **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
- \* **tipo**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.
- \* **oldHeight**
  - **Accesso:** Private;
  - **Tipo:** Double;
  - **Descrizione:** indica la vecchia altezza dell'Elemento<sub>g</sub>.
- \* **oldWidth**
  - **Accesso:** Private;
  - **Tipo:** Double;
  - **Descrizione:** indica la vecchia larghezza dell'Elemento<sub>g</sub>.

## Metodi

- **ConcreteEditSizeCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteEditSizeCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editSize(spec) passando come parametro l'oggetto spec. editSize ritorna un oggetto a cui ConcreteEditSizeCommand inizializza oldSize. Se executed è settato a false lo setta come true, altrimenti invoca il metodo ridimensionaElemento(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editSize(spec) passando come parametro il campo dati oldSize. Invoca il metodo ridimensionaElemento(oldSize) di EditController.

### 5.3.2.17 Classe ConcreteEditContentCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditContentCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione e il contenuto di testo che deve essere assegnato all'Elemento<sub>g</sub>, invoca il metodo `InsertEditRemove::editContent(spec)`.

## Attributi

- **oldContent**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che contiene i parametri di dimensione dell'Elemento<sub>g</sub> modificato, contiene al suo interno i campi:
    - \* **id**
      - **Accesso:** Private;
      - **Tipo:** Int;
      - **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
    - \* **tipo**
      - **Accesso:** Private;
      - **Tipo:** String;
      - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.
    - \* **oldContent**
      - **Accesso:** Private;
      - **Tipo:** Double;
      - **Descrizione:** indica il vecchio contenuto dell'Elemento<sub>g</sub>.

## Metodi

- **ConcreteEditContentCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteEditContentCommand.
- **doAction**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;

- **Descrizione:** invoca il metodo `InsertEditRemove::editContent(spec)` passando come parametro l'oggetto `spec`. `editContent` ritorna un oggetto a cui `ConcreteEditContentCommand` inizializza `oldContent`. Se `executed` è settato a `false` lo setta come `true`, altrimenti invoca il metodo `aggiornaTesto()` di `EditController` passandogli `spec`.

- undoAction()

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** invoca il metodo InsertEditRemove::editContent(spec) passando come parametro il campo dati oldContent. Invoca il metodo aggiornaTesto() di EditController passandogli oldContent.

### 5.3.2.18 Classe ConcreteEditBackgroundCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditBackgroundCommand
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione e lo sfondo da applicargli, invoca il metodo `InsertEditRemove::editBackground(spec)`.

## Attributi

- oldBackground

- **Accesso:** Private;
- **Tipo:** Oggetto;
- **Descrizione:** oggetto che contiene i vecchi parametri dello sfondo dell'Elemento<sub>g</sub> modificato, contiene al suo interno i campi:

 $\ast \text{id}$ 

- **Accesso:** Private;
- **Tipo:** Int;
- **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.

\* tipo

- **Accesso:** Private;
- **Tipo:** String;
- **Descrizione:** indica il tipo dell'Elemento<sub>e</sub> da modificare.

\* backgroundImage

- **Accesso:** Private;
- **Tipo:** String;
- **Descrizione:** indica l'URL<sub>g</sub> dell'immagine di sfondo dell'Elemento<sub>g</sub>.

\* **backgroundColor**

- **Accesso:** Private;

- **Tipo:** string;
- **Descrizione:** indica il colore dello sfondo dell'Elemento<sub>g</sub>.

## Metodi

- **ConcreteEditBackgroundCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteEditBackgroundCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editBackground(spec) passando come parametro l'oggetto spec. editBackground ritorna un oggetto a cui ConcreteEditBackgroundCommand inizializza oldBackground. ConcreteEditBackgroundCommand assegna quindi i campi dati id e tipo dell'oggetto oldBackground. Se executed è settato a false lo setta come true, altrimenti invoca il metodo updateSfondoFrame(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editBackground(spec) passando come parametro il campo dati oldBackground. Invoca il metodo updateSfondoFrame(oldBackground) di EditController.

### 5.3.2.19 Classe ConcreteEditColorCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditColorCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione e il colore da applicargli, invoca il metodo `InsertEditRemove::editColor(spec)`.

## Attributi

- **oldColor**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che contiene i vecchi parametri dello sfondo dell'Elemento<sub>g</sub> modificato, contiene al suo interno i campi:



- \* **id**
  - **Accesso:** Private;
  - **Tipo:** Int;
  - **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
- \* **tipo**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.
- \* **color**
  - **Accesso:** Private;
  - **Tipo:** String;
  - **Descrizione:** indica il colore dell'Elemento<sub>g</sub>.

## Metodi

- **ConcreteEditColorCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteEditColorCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editColor(spec) passando come parametro l'oggetto spec. editColor ritorna un oggetto a cui ConcreteEditColorCommand inizializza oldColor. ConcreteEditColorCommand assegna quindi i campi dati id e tipo dell'oggetto oldColor. Se executed è settato a false lo setta come true, altrimenti invoca il metodo cambiaColoreTesto() di EditController passandogli spec.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editColor(spec) passando come parametro il campo dati oldColor. Invoca il metodo cambiaColoreTesto() di EditController passandogli oldColor.

#### 5.3.2.20 Classe ConcreteEditFontCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditFontCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione e lo sfondo da applicargli, invoca il metodo `InsertEditRemove::editFont(spec)`.

## Attributi

- oldFont

- **Accesso:** Private;
- **Tipo:** Oggetto;
- **Descrizione:** oggetto che contiene i vecchi parametri dello sfondo dell'Elemento<sub>g</sub> modificato, contiene al suo interno i campi:
  - \* **id**
    - **Accesso:** Private;
    - **Tipo:** Int;
    - **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
  - \* **tipo**
    - **Accesso:** Private;
    - **Tipo:** String;
    - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.
  - \* **font**
    - **Accesso:** Private;
    - **Tipo:** String;
    - **Descrizione:** indica il Font<sub>g</sub> dell'Elemento<sub>g</sub>.

## Metodi

- **ConcreteEditFontCommand(spec:object)**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** costruisce l'oggetto ConcreteEditFontCommand.

- **doAction()**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** invoca il metodo InsertEditRemove::editFont(spec) passando come parametro l'oggetto spec. editFont ritorna un oggetto a cui ConcreteEditFontCommand inizializza oldFont. ConcreteEditFontCommand assegna quindi i campi dati id e tipo dell'oggetto oldFont. Se executed è settato a false lo setta come true, altrimenti invoca il metodo cambiaFontTesto() di EditController passandogli spec.

- undoAction()

- **Accesso:** Public;

- **Tipo di ritorno:** Void;
- **Descrizione:** invoca il metodo InsertEditRemove::editFont(spec) passando come parametro il campo dati oldFont. Invoca il metodo cambiaFontTesto() di EditController passandogli oldFont.

### 5.3.2.21 Classe ConcreteEditBookmarkCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteEditFontCommand.
```

## Utilizzo

Viene costruito da `Premii::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione e lo sfondo da applicargli, invoca il metodo `InsertEditRemove::editFont(spec)`.

## Attributi

- **oldFont**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che contiene i vecchi parametri dello sfondo dell'Elemento<sub>g</sub> modificato, contiene al suo interno i campi:
    - \* **id**
      - **Accesso:** Private;
      - **Tipo:** Int;
      - **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
    - \* **tipo**
      - **Accesso:** Private;
      - **Tipo:** String;
      - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.
    - \* **font**
      - **Accesso:** Private;
      - **Tipo:** String;
      - **Descrizione:** indica il Font<sub>g</sub> dell'Elemento<sub>g</sub>.

## Metodi

- **ConcreteEditFontCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcreteEditFontCommand.
- **doAction**()
  - **Accesso:** Public;

- **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::editFont(spec) passando come parametro l'oggetto spec. editFont ritorna un oggetto a cui ConcreteEditFontCommand inizializza oldFont. ConcreteEditFontCommand assegna quindi i campi dati id e tipo dell'oggetto oldFont. Se executed è settato a false lo setta come true, altrimenti invoca il metodo updateBookmark() di EditController passandogli spec.
- **undoAction()**
- **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo updateBookmark() di EditController passandogli spec.

### 5.3.2.22 Classe ConcretePortaAvantiCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcretePortaAvantiCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione, invoca il metodo `InsertEditRemove::portaAvanti(spec)`.

## Attributi

- **oldZIndex**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che contiene i vecchi parametri dello z-index dell'Elemento<sub>g</sub> modificato. Tale oggetto contiene i seguenti campi:
    - \* **id**
      - **Accesso:** Private;
      - **Tipo:** Int;
      - **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
    - \* **tipo**
      - **Accesso:** Private;
      - **Tipo:** String;
      - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.

## Metodi

- **ConcretePortaAvantiCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;

- **Descrizione:** costruisce l'oggetto ConcretePortaAvantiCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::portaAvanti(spec) passando come parametro l'oggetto spec. Se executed è settato a false lo setta come true, altrimenti invoca il metodo portaAvanti(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::portaDietro(spec) passando come parametro il campo dati oldZIndex. Invoca il metodo portaDietro(oldZIndex) di EditController.

### 5.3.2.23 Classe ConcretePortaDietroCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcretePortaDietroCommand.
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id dell'Elemento<sub>g</sub> dalla presentazione, invoca il metodo `InsertEditRemove::portaDietro(spec)`.

## Attributi

- **oldZIndex**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che contiene i vecchi parametri dello z-index dell'Elemento<sub>g</sub> modificato. Tale oggetto contiene i seguenti campi:
    - \* **id**
      - **Accesso:** Private;
      - **Tipo:** Int;
      - **Descrizione:** indica l'id dell'Elemento<sub>g</sub> da modificare.
    - \* **tipo**
      - **Accesso:** Private;
      - **Tipo:** String;
      - **Descrizione:** indica il tipo dell'Elemento<sub>g</sub> da modificare.

## Metodi

- **ConcretePortaDietroCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto ConcretePortaDietroCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::portaDietro(spec) passando come parametro l'oggetto spec. Se executed è settato a false lo setta come true, altrimenti invoca il metodo portaDietro(spec) di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::portaAvanti(spec) passando come parametro il campo dati oldZIndex. Invoca il metodo portaAvanti(oldZIndex) di EditController.

### 5.3.2.24 Classe ConcreteAddToMainPathCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::concreteAddToMainPathCommand

## Utilizzo

Viene costruito da Premi::Controller::EditController, riceve l'id del Frame<sub>g</sub> da inserire nella presentazione e la sua posizione all'interno del Percorso<sub>g</sub> principale.

## Metodi

- **concreteAddToMainPathCommand(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto concreteAddToMainPathCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::addFrameToMainPath(spec) passando come parametro l'oggetto spec. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo aggiungiMainPath(spec) di EditController.

- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeFrameToMainPath(id) passando come parametro il campo id del parametro spec. Invoca il metodo rimuoviMainPath(spec) di EditController.

### 5.3.2.25 Classe ConcreteRemoveFromMainPathCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::concreteRemoveFromMainPathC
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id del `Frameg` da rimuovere dal `Percorsog` principale della presentazione.

## Attributi

- **oldFrame**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che contiene i vecchi parametri che indicano la posizione del  $\text{Frame}_g$  all'interno del  $\text{Percorso}_g$  originale prima della rimozione:
    - \* **id**
      - **Accesso:** Private;
      - **Tipo:** Int;
      - **Descrizione:** indica l'id del  $\text{Frame}_g$  da rimuovere all'interno del  $\text{Percorso}_g$  principale.
    - \* **pos**
      - **Accesso:** Private;
      - **Tipo:** Integer;
      - **Descrizione:** indica la posizione del  $\text{Frame}_g$  nel  $\text{Percorso}_g$  principale, prima dell'avvenuta rimozione.

## Metodi

- **concreteRemoveFromMainPathCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto concreteRemoveFromMainPathCommand.
- **doAction**()

- **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo `InsertEditRemove::removeFrameFromMainPath(spec)` passando come parametro l'oggetto `spec`. `removeFrameFromMainPath` ritorna un oggetto a cui `concreteRemoveFromMainPathCommand` inizializza `oldFrame`. Se `executed` è posto a zero, lo pone a uno, altrimenti invoca il metodo `rimuoviMainPath(null, spec)` di `EditController`.
- **undoAction()**
    - **Accesso:** Public;
    - **Tipo di ritorno:** Void;
    - **Descrizione:** invoca il metodo `InsertEditRemove::addFrameToMainPath(id)` passando come parametro l'oggetto `oldFrame`. Invoca il metodo `aggiungiMainPath(oldFrame)` di `EditController`.

### 5.3.2.26 Classe ConcreteNewChoicePathCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteNewChoicePathCommand

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id del `Frameg` da cui parte il nuovo `Percorsog` scelta.

## Attributi

- **PathId**
  - **Accesso:** Private;
  - **Tipo:** Int;
  - **Descrizione:** indica l'id del Percorso<sub>g</sub>.

## Metodi

- **concreteNewChoicePathCommand**(id: integer)
  - \* **Accesso:** Public;
  - \* **Tipo di ritorno:** Void;
  - \* **Descrizione:** costruisce l'oggetto concreteNewChoicePathCommand.
- **doAction**()
  - \* **Accesso:** Public;
  - \* **Tipo di ritorno:** Void;
  - \* **Descrizione:** invoca il metodo InsertEditRemove::addChoicePath() passando come parametro il valore spec.id ricevuto come parametro. addChoicePath ritorna il valore a cui viene inizializzato pathId. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo appropriato di EditController.



- **undoAction()**
  - \* **Accesso:** Public;
  - \* **Tipo di ritorno:** Void;
  - \* **Descrizione:** invoca il metodo InsertEditRemove::deleteChoicePath() passando come parametro il valore pathId. Invoca il metodo appropriato di EditController.

### 5.3.2.27 Classe ConcreteNewChoicePathCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteNewChoicePathCommand

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id da cui parte il nuovo Percorso<sub>g</sub> scelta.

## Attributi

- **pathId**
  - **Accesso:** Private;
  - **Tipo:** Integer;
  - **Descrizione:** copia dell'id del Percorso<sub>g</sub> eliminato.

## Metodi

- **concreteNewChoicePathCommand(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto concreteDeleteChoicePathCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::addChoicePath() passando come parametro il valore spec.id. deleteChoicePath ritorna la copia del Percorso<sub>g</sub> rimosso, a cui viene inizializzato oldPath. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo appropriato EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::deleteChoicePath() passando come parametro l'oggetto pathId. Invoca il metodo appropriato di EditController.



### 5.3.2.28 Classe ConcreteDeleteChoicePathCommand

#### Funzione

Classe concreta, è interfaccia del Design Pattern Command.

#### Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteDeleteChoicePathCommand

#### Utilizzo

Viene costruito da Premi::Controller::EditController, riceve l'id del Percorso<sub>g</sub> scelta da rimuovere.

#### Attributi

- **oldPath**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** copia del Percorso<sub>g</sub> eliminato.

#### Metodi

- **concreteDeleteChoicePathCommand(spec: object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto concreteDeleteChoicePathCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::deleteChoicePath() passando come parametro il valore spec.id. deleteChoicePath ritorna la copia del Percorso<sub>g</sub> rimosso, a cui viene inizializzato oldPath. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo appropriato di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::addChoicePath() passando come parametro l'oggetto oldPath. Invoca il metodo appropriato di EditController.

### 5.3.2.29 Classe ConcreteRemoveFromChoicePathCommand

#### Funzione

Classe concreta, è interfaccia del Design Pattern Command.

#### Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteRemoveFromChoicePathCommand

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id del `Frameg` da rimuovere da un `Percorsog` a scelta.

## Attributi

- **oldFrame**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** copia del Frame<sub>g</sub> eliminato.

## Metodi

- **concreteRemoveFromChoicePathCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto concreteRemoveFromChoicePathCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeFrameFromChoicePath(spec). removeFrameFromChoicePath ritorna la copia del Frame<sub>g</sub> rimosso, a cui viene inizializzato oldFrame. Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo appropriato di EditController.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::addFrameToChoicePath() passando come parametro l'oggetto oldFrame. Invoca il metodo appropriato di EditController.

### 5.3.2.30 Classe ConcreteAddToChoicePathCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteAddToChoicePathComm

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id del `Frameg` da aggiungere ad un `Percorsog` a scelta.

## Metodi

- **concreteAddToChoicePathCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto concreteAddToChoicePathCommand.
- **doAction**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::addFrameToChoicePath(spec). Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo appropriato di EditController.
- **undoAction**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeFrameFromChoicePath(spec) e invoca il metodo appropriato di EditController.

### 5.3.2.31 Classe ConcreteAddToMainPathCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteAddToMainPathComm
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id del `Frameg` da aggiungere al `Percorsog` principale.

## Metodi

- **concreteAddToMainPathCommand**(spec:object)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto concreteAddToMainPathCommand.
- **doAction**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::addFrameToMainPath(spec). Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo aggiungiMainPath(spec) di EditController.

- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeFrameFromMainPath(spec) e invoca il metodo rimuoviMainPath() di EditController passandogli spec.

### 5.3.2.32 Classe ConcreteRemoveFromMainPathCommand

## Funzione

Classe concreta, è interfaccia del Design Pattern Command.

## Scope

```
Model::SlideShow::SlideShowActions::Command::AbstractCommand::ConcreteRemoveFromMainPathC
```

## Utilizzo

Viene costruito da `Premi::Controller::EditController`, riceve l'id del `Frameg` da togliere dal `Percorsog` principale.

## Metodi

- **concreteRemoveFromMainPathCommand(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** costruisce l'oggetto concreteRemoveFromMainPathCommand.
- **doAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::removeFrameFromMainPath(spec). Se executed è posto a zero, lo pone a uno, altrimenti invoca il metodo rimuoviMainPath() di EditController passandogli spec.
- **undoAction()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** invoca il metodo InsertEditRemove::addFrameToMainPath(spec) e invoca il metodo aggiungiMainPath(spec) di EditController.

## 5.4 serverRelation

### 5.4.1 Loader

Loader
-toInsert : object
-toUpdate : object
-toDelete : object
-toPaths : bool
+update() : bool
+addInsert(idElement : string) : bool
+addUpdate(idElement : string) : bool
+addDelete(idElement : string) : bool
+addPaths() : bool

Fig 2: Diagramma classe Model::serverRelation::loader::Loader

**Attributi:**

- - **toInsert**
  - **Accesso:** private
  - **Tipo:** object
  - **Descrizione:** array di identificativi di elementi<sub>g</sub> inseriti in locale dall'ultima sincronizzazione verso MongoDB
- - **toUpdate**
  - **Accesso:** private
  - **Tipo:** object
  - **Descrizione:** array di identificativi di elementi<sub>g</sub> modificati in locale dall'ultima sincronizzazione verso MongoDB
- - **toDelete**
  - **Accesso:** private
  - **Tipo:** object
  - **Descrizione:** array di identificativi di elementi<sub>g</sub> eliminati in locale dall'ultima sincronizzazione verso MongoDB
- - **toPaths**
  - **Accesso:** private
  - **Tipo:** bool
  - **Descrizione:** valore booleano che indica se è stato modificato l'oggetto paths della presentazione dall'ultima update()

## Metodi:

- `update()` : bool

- **Accessibilità:** public
- **Descrizione:** scorre gli array toInsert, toUpdate, toDelete e chiama le funzioni `mongoRelation` per la sincronizzazione della presentazione sul database MongoDB
- **Tipo di ritorno:** bool
- **addInsert(idElement : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** inserisce la stringa parametro in toInsert
  - **Tipo di ritorno:** bool
- **addUpdate(idElement : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** inserisce la stringa parametro in toUpdate se non è presente in toInsert
  - **Tipo di ritorno:** bool
- **addDelete(idElement : string, typeObj : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** inserisce la stringa parametro in toDelete se non è presente in toInsert, altrimenti non inserisce nulla in toDelete e cancella l'identificativo da toInsert, sempre cancella da toUpdate
  - **Tipo di ritorno:** bool
- **addPaths() : bool**
  - **Accessibilità:** public
  - **Descrizione:** setta il valore del campo dati toPaths a true
  - **Tipo di ritorno:** bool

### 5.4.2 Registration

Registration
-messageState : String
+Registration()
+register(user : string, password : string) : bool
+getMessage() : string

Fig 3: Diagramma classe Model::serverRelation::accessControll::Registration

**Attributi:**

- **-messageState**
  - **Accesso:** private
  - **Tipo:** string
  - **Descrizione:** messaggio ritornato dall'ultima chiamata di autenticazione al Server<sub>g</sub>

## Metodi:

- **register(user : string, password : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** chiama il servizio /account/register POST con passando i parametri attuali, scrive in messageState lo stato ritornato dalla chiamata a nodeApi, ritorna true solo se la chiamata va a buon fine
  - **Tipo di ritorno:** bool
- **getMessage() : string**
  - **Accessibilità:** public
  - **Descrizione:** ritorna il contenuto di messageState
  - **Tipo di ritorno:** string

### 5.4.3 Authentication

Authentication
-token
-messageState
+Authentication()
+authenticate(user : string, pass : string) : bool
+deAuthenticate() : bool
+changePassword(user : string, password : string, newpassword : string)
+getToken() : string
+getMessage() : string

Fig 4: Diagramma classe Model::serverRelation::accessControl::Authentication



**Attributi:**

- **-messageState**
  - **Accesso:** private
  - **Tipo:** string
  - **Descrizione:** messaggio ritornato dall'ultima chiamata di autenticazione al Server<sub>g</sub>
- **-token**
  - **Accesso:** private
  - **Tipo:** string
  - **Descrizione:** stringa per l'autenticazione ai servizi Server<sub>g</sub> nodeJs
  - **Note:** valore ?empty? se non è ancora stato richiesto il token, è stato cancellato, oppure l'autenticazione è fallita

## Metodi:

- **authenticate(user : string, password : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** chiama il servizio Account<sub>g</sub>/authenticate, ritorna true se la chiamata è andata a buon fine, in questo caso mette nel campo token il token ritornato dal Server<sub>g</sub> nodeJs
  - **Tipo di ritorno:** bool
- **deAuthenticate() : bool**
  - **Accessibilità:** public
  - **Descrizione:** de-autentica l'utente dai servizi Server<sub>g</sub> cancellando il contenuto del campo dati token, ritorna true se l'operazione va a buon fine
  - **Tipo di ritorno:** bool
- **changePassword(user:string, password:string, newpassword:string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** modifica la password dell'utente specificato
  - **Tipo di ritorno:** bool
- **getToken() : string**
  - **Accessibilità:** public
  - **Descrizione:** ritorna il contenuto del campo dati token
  - **Tipo di ritorno:** string
- **getMessage() : string**
  - **Accessibilità:** public
  - **Descrizione:** ritorna il contenuto di messageState
  - **Tipo di ritorno:** string

#### 5.4.4 MongoRelation

MongoRelation
-messageState
+getPresentationsMeta() : object
+newPresentation(name : string) : bool
+newCopyPresentation(nameOldPresentation : string, nameNewPresentation : string) : bool
+getPresentation(namePresentation : string) : object
+deletePresentation(namePresentation : string) : bool
+renamePresentation(name : string, newName : string) : bool
+newElement(namePresentation : string, element : object, callback)
+updateElement(namePresentation : string, element : object, callback)
+deleteElement(namePresentation : string, typeObj : string, idElement : string, callback)
+updatePaths(namePresentation, element : object, callback)
+getMessage()

Fig 5: Diagramma classe Model::serverRelation::mongoRelation::MongoRelation

## Metodi:

- **getPresentationsMeta()** : **object**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio `"/private/api/presentations GET"` ritornando l'oggetto con le informazioni sulle presentazioni create dall'utente
  - **Tipo di ritorno:** object
- **newPresentation(name : string)** : **void**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio `"/private/api/presentations/new/[name] POST"` per creare una nuova presentazione sul database MongoDB
  - **Tipo di ritorno:** object
- **newCopyPresentation(nameOldPresentation : string, nameNewPresentation : string)** : **void**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio `"/private/api/presentations/new/[nameOldPresentation]/[nameNewPresentation] POST"` per creare una nuova presentazione sul database MongoDB
  - **Tipo di ritorno:** object
- **getPresentation(namePresentation : string)** : **object**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio `"/private/api/presentations/[namePresentation] GET"` per ricevere la presentazione `[namePresentation]` dell'utente

- **Tipo di ritorno:** object
- **deletePresentation(namePresentation : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio "/private/api/presentations/[namePresentation] DELETE" per eliminare la presentazione [namePresentation] creata dall'utente dal database MongoDB
  - **Tipo di ritorno:** bool
- **renamePresentation(name : string, newName : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio "/private/api/presentations/[name]/rename/[newName] POST" per rinominare la presentazione [name] dell'utente in [newName]
  - **Tipo di ritorno:** bool
- **updateElement(namePresentation: string, element : object, callback)**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio "/private/api/presentations/[namePresentation]/element PUT" per aggiornare l'Elemento<sub>g</sub> passato come parametro, esegue la Funzione<sub>g</sub> callback
  - **Tipo di ritorno:** bool
- **deleteElement(namePresentation: string, typeObj : string, idElement : string, callback)**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio "/private/api/presentations/[namePresentation]/[typeObj]/[idElement] DELETE" per eliminare l'Elemento<sub>g</sub> con identificativo [idElement] dalla presentazione nel database MongoDB, esegue la Funzione<sub>g</sub> callback
  - **Tipo di ritorno:** bool
- **newElement(namePresentation: string, element : object, callback)**
  - **Accessibilità:** public
  - **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio "/private/api/presentations/[namePresentation]/element POST" per inserire un nuovo Elemento<sub>g</sub> nella presentazione dell'utente(element) nella base dati MongoDB, esegue la Funzione<sub>g</sub> callback
  - **Tipo di ritorno:** bool
- **updatePaths(namePresentation: string, element : object, callback)**



- **Accessibilità:** public
- **Descrizione:** effettua una chiamata asincrona verso il Server<sub>g</sub> nodeJs al servizio "/private/api/presentations/[namePresentation]/paths PUT" per aggiornare il campo paths della presentazione con l'Elemento<sub>g</sub> passato come parametro, esegue la Funzione<sub>g</sub> callback
- **Tipo di ritorno:** bool

#### 5.4.5 fileServerRelation

FileServerRelation
-messageState : string
+getImagesMeta() : object
+getAudiosMeta() : object
+getVideosMeta() : object
+deleteImage(nameImage : string) : bool
+deleteAudio(nameAudio : string) : bool
+deleteVideo(nameVideo : string) : bool
+renameImage(name : string, newName : string) : bool
+renameAudio(name : string, newName : string) : bool
+renameVideo(name : string, newName : string) : bool
+getMessage() : string

Fig 6: Diagramma classe Model::serverRelation::fileServerRelation::FileServerRelation

**Attributi:**

- - **messageState**
  - **Accesso:** private
  - **Tipo:** string
  - **Descrizione:** messaggio ritornato dall'ultima chiamata al Server<sub>g</sub> nodeJs

## Metodi:

- **getImageMeta() : object**
  - **Accessibilità:** public
  - **Descrizione:** ritorna un oggetto contenente informazioni sulle immagini dell'utente nel Server<sub>g</sub>
  - **Tipo di ritorno:** object
- **getAudiosMeta() : object**
  - **Accessibilità:** public
  - **Descrizione:** ritorna un oggetto contenente informazioni sui File<sub>g</sub> audio dell'utente nel Server<sub>g</sub>
  - **Tipo di ritorno:** object
- **getVideosMeta() : object**
  - **Accessibilità:** public
  - **Descrizione:** ritorna un oggetto contenente informazioni sui File<sub>g</sub> video dell'utente nel Server<sub>g</sub>
  - **Tipo di ritorno:** object
- **deleteImage(nameImage : string) : bool**

- **Accessibilità:** public
- **Descrizione:** cancella il File<sub>g</sub> immagine nameImage dallo spazio dell'utente sul Server<sub>g</sub>
- **Tipo di ritorno:** bool
- **deleteAudio(nameAudio : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** cancella il File<sub>g</sub> audio nameAudio dallo spazio dell'utente sul Server<sub>g</sub>
  - **Tipo di ritorno:** bool
- **deleteVideo(nameVideo : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** cancella il File<sub>g</sub> video nameVideo dallo spazio dell'utente sul Server<sub>g</sub>
  - **Tipo di ritorno:** bool
- **renameImage(name : string, newName : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** rinomina il File<sub>g</sub> immagine name dell'utente nel Server<sub>g</sub> con il nuovo nome newName
  - **Tipo di ritorno:** bool
- **renameAudio(name : string, newName : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** rinomina il File<sub>g</sub> audio name dell'utente nel Server<sub>g</sub> con il nuovo nome newName
  - **Tipo di ritorno:** bool
- **renameVideo(name : string, newName : string) : bool**
  - **Accessibilità:** public
  - **Descrizione:** rinomina il File<sub>g</sub> video name dell'utente nel Server<sub>g</sub> con il nuovo nome newName
  - **Tipo di ritorno:** bool

## 6 Package Premi::Controller

Tutti i package seguenti appartengono al package Premi, quindi per ognuno di essi lo scope sarà: Premi::[nome package].

**Tipo, obiettivo e funzione del componente:** contiene le classi che gestiscono i segnali e le chiamate effettuati dalla View. Il Controller è interamente implementato utilizzando la tecnologia Angular.js.

**Relazioni d'uso di altre componenti:** comunica con la View ricevendone i segnali e mantenendola aggiornata e con il Model per recuperare informazioni, modificarle e salvare i dati correttamente nel server.

## 6.1 Controller::premiApp

## Funzione

Questa classe si occuperà di fare il bootstrap dell'applicazione istanziando la `RootScope` e iniettando tutti i moduli necessari. Verranno configurate tutte le impostazioni necessarie relative al funzionamento di Angular.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Controller;
- View::Pages;
- `ngRoute: Object`  
Modulo Angular che inietta i servizi necessari per il routing;
- `ngMaterial: Object`  
Modulo Angular che inietta i servizi necessari per l'utilizzo di Angular Material;
- `ngStorage: Object`  
Modulo Angular che inietta i servizi necessari per l'utilizzo dello storage locale.
- `$routeProvider: Object`  
Questo campo dati rappresenta il servizio che collega tra loro controller, view e l'URL<sub>g</sub> corrente nel Browser<sub>g</sub>;
- `$mdIconProvider` e `$mdThemingProvider`  
Moduli di Angular Material
- `$httpProvider`  
Provider Angular per configurare il servizio http;
- `$provide`  
Modulo Angular utilizzato per la gestione delle eccezioni;
- `$locationProvider`  
Provider Angular utilizzato per rendere disponibile il servizio di reindirizzamento delle pagine.

## 6.2 Controller::Services

**Tipo, obiettivo e funzione del componente:** questo package contiene tutte le funzionalità di base richieste dall'applicazione tra cui l'autenticazione al server, il reindirizzamento delle pagine e l'upload dei file nel server.

**Relazioni d'uso di altre componenti:** comunica con gli altri package del controller per la gestione delle funzioni<sub>g</sub> da eseguire e con il model per la gestione delle componenti necessarie.

### 6.2.1 Services::Main

## Funzione

Questa classe si occuperà di eseguire le funzioni<sub>g</sub> base dell'applicazione, in particolare autenticazione e registrazione al Server<sub>g</sub> degli utenti.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Controller::Services::Utils;
- Model::serverRelation::accessControl::Authentication;
- Model::serverRelation::accessControl::Registration;
- localStorage:Object

Servizio angular che permette il salvataggio in locale di oggetti necessari al garantire delle funzioni dell'applicazione.

## Attributi

- **login**
  - **Accesso:** Private;
  - **Tipo:** Oggetto;
  - **Descrizione:** oggetto che mantiene la sessione corrente.

## Metodi

- **value()**
  - **Accesso:** Private;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che controlla se è stato effettuato un refresh della pagina, in tal caso ripristina Login<sub>g</sub>.
- **register(formData, success, error)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;





- **Descrizione:** metodo che, attraverso l'oggetto formData contenente le credenziali di accesso, effettua la registrazione al Server<sub>g</sub> di un nuovo utente richiamando il metodo register() di Registration. Se l'operazione ha successo, viene autenticato l'utente ed invocato success altrimenti error.
- login(formData, success, error)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che, attraverso l'oggetto formData contenente le credenziali di accesso, effettua l'autenticazione al Server<sub>g</sub> di un utente richiamando il metodo authenticate() di Authentication. Se l'operazione ha successo viene invocato success altrimenti error.
- logout(success, error)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che effettua il Logout<sub>g</sub> dal Server<sub>g</sub> richiamando il metodo deauthenticate() di Authentication. Se l'operazione ha successo viene invocato success altrimenti error.
- changepassword(formData, success, error)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che, attraverso l'oggetto formData contenente le credenziali di accesso e la nuova password, effettua il cambio della password di un utente richiamando il metodo changepassword() di Authentication. Se l'operazione ha successo viene invocato success altrimenti error.
- getToken()
  - **Accesso:** Public;
  - **Tipo di ritorno:** JSONWebToken;
  - **Descrizione:** metodo che ritorna il token di sessione richiamando getToken() di Authentication.
- getUser()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che ritorna l'utente attualmente autenticato con il Server<sub>g</sub>.

### 6.2.2 Services::Upload

## Funzione

Questa classe si occuperà di eseguire l'upload di  $\text{File}_g$  media nel database.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Controller::Services::Main;
- Controller::Services::Utils.

## Attributi

- **image**
  - **Accesso:** Private;
  - **Tipo:** Array;
  - **Descrizione:** array contenente i formati immagini accettati per l'upload.
- **audio**
  - **Accesso:** Private;
  - **Tipo:** Array;
  - **Descrizione:** array contenente i formati audio accettati per l'upload.
- **video**
  - **Accesso:** Private;
  - **Tipo:** Array;
  - **Descrizione:** array contenente i formati video accettati per l'upload.

## Metodi

- **uploadmedia**(files<sub>g</sub>, callback)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che invia una richiesta XMLHttpRequest a *[hostname]/private/api/files/-[image/audio/video]/[nome\_file]* effettuando l'upload dei File<sub>g</sub> contenuti nell'array files<sub>g</sub> passato come parametro. Se l'operazione ha successo viene richiamato callback, altrimenti viene lanciato un errore.
- **isImage**(files<sub>g</sub>)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Bool;
  - **Descrizione:** metodo che ritorna true se i File<sub>g</sub> contenuti nell'array files<sub>g</sub>, passato come parametro, rispettano almeno uno tra i formati contenuti nell'array image, altrimenti ritorna false.

- **isAudio**(files<sub>g</sub>)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Bool;
  - **Descrizione:** metodo che ritorna true se i File<sub>g</sub> contenuti nell'array files<sub>g</sub>, passato come parametro, rispettano almeno uno tra i formati contenuti nell'array audio, altrimenti ritorna false.
- **isVideo**(files<sub>g</sub>)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Bool;
  - **Descrizione:** metodo che ritorna true se i File<sub>g</sub> contenuti nell'array files<sub>g</sub>, passato come parametro, rispettano almeno uno tra i formati contenuti nell'array video, altrimenti ritorna false.
- **getFileUrl**(File<sub>g</sub>)
  - **Accesso:** Public;
  - **Tipo di ritorno:** String;
  - **Descrizione:** metodo che ritorna il Percorso<sub>g</sub> di salvataggio del parametro File<sub>g</sub> rispetto all'utente corrente.

### 6.2.3 Services::Utils

## Funzione

Questa classe si occuperà di eseguire piccole funzionalità utili ad ogni parte dell'applicazione.

## Metodi

- **decodeToken(token)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che decodifica token, passato come parametro, e ritorna l'oggetto utente corrispondente.
- **grade(password)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** String;
  - **Descrizione:** metodo che determina la robustezza del parametro password. La lunghezza minima di una password è stata impostata a sei caratteri.
- **hostname()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** String;

- **Descrizione:** metodo che ritorna il dominio dell'applicazione.
- **isUndefined(object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Bool;
  - **Descrizione:** metodo che ritorna true se il parametro object risulta indefinito, altrimenti ritorna false.
- **isObject(object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che ritorna true se il parametro object risulta definito, altrimenti ritorna false.
- **encrypt(string)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** String;
  - **Descrizione:** metodo che ritorna il parametro string criptato. Il metodo di criptaggio scelto è lo SHA-1.

### 6.2.4 Services::SharedData

## Funzione

Questa classe mantiene in memoria la presentazione sulla quale l'utente sta lavorando.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Controller::Services::Utils;
- Controller::Services::Main;
- Model::serverRelation::mongoRelation;

- localStorage:Object

Servizio angular che permette il salvataggio in locale di oggetti necessari al garantire delle funzioni dell'applicazione.

## Attributi

- **myPresentation**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** oggetto che rappresenta l'attuale presentazione aperta.

## Metodi

- **getPresentazione**(idSlideShow)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che, nel caso in cui il parametro idSlideShow sia definito, richiama il metodo `Model::serverRelation::mongoRelation::getPresentation()` passandogli il parametro idSlideShow e assegnando il risultato a myPresentation. In ogni caso myPresentation viene ritornato.

### 6.2.5 Services::toPages

## Funzione

Questa classe si occuperà di eseguire i reindirizzamenti alle pagine corrette.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Controller::Services::Utils;
- Controller::Services::Main;
- Controller::Services::SharedData;
- \$http:Object  
Servizio Angular che permette la comunicazione in remoto con un Server<sub>g</sub>.
- \$location:Object  
Servizio Angular che gestisce gli indirizzi<sub>g</sub> URL<sub>g</sub>.

## Metodi

- **sendRequest**(dest, success, error)
  - **Accesso**: Private;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che ritorna una richiesta http all'Indirizzo<sub>g</sub> definito dal parametro dest. Se l'operazione ha successo viene invocato success altrimenti error.
- **loginpage**()
  - **Accesso**: Public;
  - **Tipo di ritorno**: Object;
  - **Descrizione**: metodo che permette di accedere alla pagina di Login<sub>g</sub>. Esso richiama il metodo sendRequest(), per effettuare una richiesta a */publicpages/login*, il quale, se ha successo, reindirizza alla pagina richiesta.
- **registrazionepage**()

- **Accesso:** Public;
- **Tipo di ritorno:** Object;
- **Descrizione:** metodo che permette di accedere alla pagina di Registrazione. Esso richiama il metodo `sendRequest()`, per effettuare una richiesta a `/publicpages/registrazione`, il quale, se ha successo, reindirizza alla pagina richiesta.
- **homepage()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che permette di accedere alla pagina Home. Esso richiama il metodo `sendRequest()`, per effettuare una richiesta a `/private/home`, il quale, se ha successo, reindirizza alla pagina richiesta.
- **profilepage()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che permette di accedere alla pagina Profile. Esso richiama il metodo `sendRequest()`, per effettuare una richiesta a `/private/profile`, il quale, se ha successo, reindirizza alla pagina richiesta.
- **editpage(slideId)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che permette di accedere alla pagina di Edit. Esso richiama il metodo `sendRequest()`, per effettuare una richiesta a `/private/edit`, il quale, se ha successo, reindirizza alla pagina richiesta e richiama il metodo `SharedData.forEdit()` passandogli il parametro `slideId`.
- **executionpage(slideId)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che permette di accedere alla pagina di Execution. Esso richiama il metodo `sendRequest()`, per effettuare una richiesta a `/private/execution`, il quale, se ha successo, reindirizza alla pagina richiesta e richiama il metodo `SharedData.forEdit()` passandogli il parametro `slideId`.

### 6.2.6 Controller::HeaderController

## Funzione

Questa classe si occuperà di controllare l'Header dell'applicazione.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- View::Pages::Index;
- Controller::Services::Utils;
- Controller::Services::Main;
- Controller::Services::toPages;
- \$scope:Object  
Oggetto Angular che lega Controller e View. Nello specifico, permette l'esecuzione di espressioni che mantengono aggiornata la View e l'implementazione del 2-way data binding. Possono esistere più scope, strutturati in modo gerarchico, i quali simulano la struttura DOM dell'applicazione;
- \$rootScope:Object  
Questo campo dati rappresenta lo scope radice dell'applicazione. Tutti gli altri scope discendono da questo.

## Metodi

- **goLogin()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama loginpage() di toPages per effettuare il reindirizzamento alla pagina di Login<sub>g</sub>.
- **goRegistrazione()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama registrazionepage() di toPages per effettuare il reindirizzamento alla pagina di registrazione.
- **goHome()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama homepage() di toPages per effettuare il reindirizzamento alla pagina home.
- **goProfile()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama profilepage() di toPages per effettuare il reindirizzamento alla pagina profile.
- **who()**

- **Accesso:** Public;
- **Tipo di ritorno:** String;
- **Descrizione:** metodo che ritorna lo username dell'utente attualmente autenticato richiamando il metodo `getUser()` di Main.
- **isToken()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Boolean;
  - **Descrizione:** metodo che verifica l'effettiva autenticazione dell'utente richiamando il metodo `getToken()` di Main.
- **logout()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama il metodo `logout()` di Main per effettuare il Logout<sub>g</sub> dal Server<sub>g</sub>. Se l'operazione va a buon fine, viene effettuato il reindirizzamento alla pagina di Login<sub>g</sub> richiamando il metodo `loginpage()` di `toPages`.
- **error()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** String;
  - **Descrizione:** metodo che ritorna l'errore individuato; esso deve essere posto all'interno di `$rootScope.error`.

### 6.2.7 Controller::AccessController

## Funzione

Questa classe si occuperà di controllare che le credenziali di accesso siano corrette nel caso dell'autenticazione oppure di registrare un nuovo utente.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- View::Pages::Login<sub>g</sub>;
- View::Pages::Registrazione;
- Controller::Services::Utils;
- Controller::Services::Main;
- Controller::Services::toPages;
- \$scope:Object

Oggetto Angular che lega Controller e View. Nello specifico, permette l'esecuzione di espressioni che mantengono aggiornata la View e l'implementazione del 2-way data binding. Possono esistere più scope, strutturati in modo gerarchico, i quali simulano la struttura DOM dell'applicazione.



## Attributi

- **user**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** oggetto contenente username e password derivanti dal form della pagina html.
- **usernameError**
  - **Accesso:** Public;
  - **Tipo:** String;
  - **Descrizione:** stringa contenente l'eventuale errore accorso con l'inserimento dello username.
- **passwordError**
  - **Accesso:** Public;
  - **Tipo:** String;
  - **Descrizione:** stringa contenente l'eventuale errore accorso con l'inserimento della password.

## Metodi

- **getData()**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** metodo che ritorna un oggetto contenente i campi dati username e password ricavati dallo \$scope. La password viene criptata grazie al metodo encrypt(password) di Utils.
- **reset()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che cancella i valori delle variabili all'interno di \$scope.
- **login()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che controlla se user è definito e, in caso affermativo, richiama login(data) di Main per effettuare il Login<sub>g</sub> al Server<sub>g</sub>. Se l'operazione ha successo viene effettuato il reindirizzamento alla pagina home richiamando il metodo homepage() di toPages.

- registration()

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che controlla se user è definito e, in caso affermativo, richiama register(data) di Main per effettuare la registrazione al Server<sub>g</sub>. Se l'operazione ha successo viene effettuato il reindirizzamento alla pagina home richiamando il metodo homepage() di toPages.

### 6.2.8 Controller::HomeController

## Funzione

Questa classe si occuperà di gestire i segnali e le chiamate provenienti dalla pagina Home.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- View::Pages::Home;
- Model::serverRelation::mongoRelation;
- Services::Utils;
- Services::Main;
- Services::toPages;
- \$scope:Object

Oggetto Angular che lega Controller e View. Nello specifico, permette l'esecuzione di espressioni che mantengono aggiornata la View e l'implementazione del 2-way data binding. Possono esistere più scope, strutturati in modo gerarchico, i quali simulano la struttura DOM dell'applicazione.

## Attributi

- mongo

- **Accesso:** Private;
- **Tipo:** Object;
- **Descrizione:** oggetto che mantiene una istanza di MongoRelation.

## Metodi

- `update()`

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che aggiorna i contenuti della pagina home.

- **goEdit**(slideId)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama editpage(slideId) di toPages per effettuare il reindirizzamento alla pagina di edit.
- **goExecute**(slideId)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama executionpage(slideId) di toPages per effettuare il reindirizzamento alla pagina di esecuzione.
- **getSS**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che richiama getPresentationsMeta() di mongoRelation per visualizzare le presentazioni dell'utente corrente.
- **deleteSlideShow**(slideId)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama deletePresentation(slideId) di mongoRelation per eliminare la presentazione slideId dal database. Se l'operazione ha successo, viene richiamato il metodo update().
- **renameSlideShow**(nameSS, rename)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama renamePresentation(nameSS, rename) di mongoRelation per rinominare la presentazione slideId. Se l'operazione ha successo, viene richiamato il metodo update().
- **createSlideShow**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama newPresentation() di mongoRelation per creare una nuova presentazione. Se l'operazione ha successo, viene richiamato il metodo update().
- **setStyleSS**(slide)

- **Accesso:** Public;
- **Tipo di ritorno:** CSS Object;
- **Descrizione:** metodo che ricava il background da slide e ritorna l’oggetto con le proprietà CSS corrispondenti. Se l’immagine di background non è definita, allora nel CSS non verrà riportata tale proprietà.

### 6.2.9 Controller::ProfileController

## Funzione

Questa classe si occupa di gestire i segnali e le chiamate provenienti dalla pagina profilo di un utente.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- Services::Utils;
- Services::Main;
- Services::toPages;
- \$scope:Object

Oggetto Angular che lega Controller e View. Nello specifico, permette l'esecuzione di espressioni che mantengono aggiornata la View e l'implementazione del 2-way data binding. Possono esistere più scope, strutturati in modo gerarchico, i quali simulano la struttura DOM dell'applicazione.

## Attributi

- **user**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** oggetto contenente username, password e nuova password derivanti dal form della pagina html.

## Metodi

- **getData()**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** metodo che ritorna un oggetto contenente i campi dati username, password e newpassword ricavati dallo \$scope. Le password vengono criptate grazie a encrypt(password) di Utils.
- **changepassword()**
  - **Accesso:** Public;

- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che controlla se user è definito e, in caso affermativo, richiama il metodo changepassword() di Main per cambiare la password dell'utente, passandogli i dati da modificare.

### 6.2.10 Controller::ExecutionController

## Funzione

Questa classe si occuperà di gestire i segnali e le chiamate provenienti dalla pagina di esecuzione.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- View::Pages::Execution;
- Services::SharedData;
- Services::Main;
- Services::toPages;
- Services::Utils;

- \$scope:Object

Oggetto Angular che lega Controller e View. Nello specifico, permette l'esecuzione di espressioni che mantengono aggiornata la View e l'implementazione del 2-way data binding. Possono esistere più scope, strutturati in modo gerarchico, i quali simulano la struttura DOM dell'applicazione;

- \$route:Object

Servizio angular che permette la gestione del routing all'interno dell'applicazione.

## Metodi

- on `$locationChangeSuccess()`

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** evento che permette l’interazione di Angular.js e Impress.js. Dato che il Framework<sub>g</sub> Impress.js cambia l’URL<sub>g</sub> della pagina in modo dinamico, è necessario istruire il routing di Angular su come comportarsi, in modo tale che non ci sia alcun reindirizzamento inopportuno.

- `translateImpress(json)`

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che richiama una Funzione<sub>g</sub> JavaScript<sub>g</sub> per la traduzione dell'oggetto json, ricavato tramite getPresentazione() di SharedData, in html eseguibile dal Framework<sub>g</sub> Impress.js.

- **goHome()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama `homepage()` di `toPages` per effettuare il reindirizzamento alla pagina home. Nonostante tale metodo sia già presente in `Header-Controller`, è necessario avere la possibilità di accedere alla home, in quanto l'header viene nascosto per permettere la piena funzionalità ad `Impress.js` e visualizzare la presentazione a pieno schermo.
- **goEdit()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama `editpage()` di `toPages` per effettuare il reindirizzamento alla pagina di edit.

### 6.2.11 Controller::EditController

## Funzione

Questa classe si occuperà di mostrare all'utente la possibilità di apportare modifiche ad una presentazione.

## Relazioni d'uso con altri moduli

Questa classe utilizzerà le seguenti classi:

- View::Pages::Edit;
- View::Pages::[javascript\_functions];
- Model::SlideShow::SlideShowActions::Command:
  - Invoker;
  - ConcreteTextInsertCommand;
  - ConcreteFrameInsertCommand;
  - ConcreteImageInsertCommand;
  - ConcreteSVGInsertCommand;
  - ConcreteAudioInsertCommand;
  - ConcreteVideoInsertCommand;
  - ConcreteBackgroundInsertCommand;
  - ConcreteTextRemoveCommand;
  - ConcreteFrameRemoveCommand;
  - ConcreteImageRemoveCommand;
  - ConcreteSVGRemoveCommand;

- ConcreteAudioRemoveCommand;
  - ConcreteVideoRemoveCommand;
  - ConcreteEditSizeCommand;
  - ConcreteEditPositionCommand;
  - ConcreteEditRotationCommand;
  - ConcreteEditColorCommand;
  - ConcreteEditBackgroundCommand;
  - ConcreteEditFontCommand;
  - ConcreteEditContentCommand;
  - ConcretePortaAvantiCommand;
  - ConcretePortaDietroCommand;
  - ConcreteAddToMainPathCommand;
  - ConcreteRemoveFromMainPathCommand
- Model::serverRelation::Loader;
  - Services::Main;
  - Services::toPages;
  - Services::Utils;
  - Services::SharedData;
  - Services::Upload;
  - \$scope:Object  
Oggetto Angular che lega Controller e View. Nello specifico, permette l'esecuzione di espressioni che mantengono aggiornata la View e l'implementazione del 2-way data binding. Possono esistere più scope, strutturati in modo gerarchico, i quali simulano la struttura DOM dell'applicazione;
  - \$interval:Object  
servizio Angular che permette di eseguire determinate operazioni ad ogni intervallo di tempo T;
  - \$q::Object  
Servizio Angular che permette di eseguire funzioni<sub>g</sub> in modo asincrono;
  - \$mdSideNav::Object  
Servizio Angular Material per il controllo della barra laterale;
  - \$mdBottomSheet::Object  
Servizio Angular Material per il controllo dell'oggetto mdBottomsSheet.

## Attributi

- **inv**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** oggetto che mantiene una istanza di Invoker.
- **mongo**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** oggetto che mantiene una istanza di mongoRelation.
- **canRemoveBookmark**
  - **Accesso:** Public;
  - **Tipo:** Boolean;
  - **Descrizione:** oggetto utilizzato per il data-binding, in base al quale, se settato a true, il frame selezionato non ha un bookmark e per questo è possibile aggiungerlo.
- **canRemoveBookmark**
  - **Accesso:** Public;
  - **Tipo:** Boolean;
  - **Descrizione:** oggetto utilizzato per il data-binding, in base al quale, se settato a true, il frame selezionato ha un bookmark e per questo è possibile la sua rimozione.
- **allFonts**
  - **Accesso:** Private;
  - **Tipo:** Object;
  - **Descrizione:** oggetto che contiene tutti caratteri possibili che un elemento testo può avere.

## Metodi

- **translateEdit(json)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che permette la traduzione dell’oggetto json, ricavato tramite `getPresentazione()` di `SharedData`, in html permettendo all’utente di modificare la presentazione.
- **goExecute()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;



- **Descrizione:** metodo che richiama `executionpage()` di `toPages` per effettuare il reindirizzamento alla pagina di `execution`.
- **`toggleList()`**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che gestisce `$mdBottomSheet` e `$mdSidenav`.
- **`showPathBottomSheet($event)`**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che fa apparire `$mdBottomSheet` per la visualizzazione dei percorsi<sub>g</sub>.
- **`show(id)`**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che gestisce la comparsa/scomparsa dei bottoni di gestione della presentazione (inserimento, rimozione, etc.) in base all'id dell'Elemento<sub>g</sub> html cliccato.
- **`salvaPresentazione()`**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che richiama `update()` di `Loader` per salvare la presentazione nel database. Questo metodo viene utilizzato in congiunta ad `$interval` in modo da assicurare un salvataggio continuo della presentazione ma senza creare troppo traffico in rete.
- **`inserisciFrame(spec:object)`**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che inserisce un `Frameg` nel piano della presentazione<sub>g</sub>, attraverso la Funzione<sub>g</sub> `JavaScriptg inserisciFrame(spec)` di `Edit` `inserisciFrame(spec)`, e che richiama, utilizzando il metodo `execute` di `inv`, `ConcreteFrameInsertCommand()` del `Command` passandogli le specifiche del `Frameg` inserito. Infine, richiama `addInsert()` di `Loader` passandogli l'identificativo del `Frameg` inserito. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di `undo/redo` da `Command`, per cui il metodo si occuperà solamente di aggiornare la `view`.
- **`inserisciTesto(spec:object)`**

- **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che inserisce un `Elementog` testo nel piano della presentazione<sub>g</sub>, attraverso la `Funzioneg JavaScriptg inserisciTesto(spec)` di `Edit`, e che richiama, utilizzando il metodo `execute` di `inv`, `ConcreteTextInsertCommand()` di `Command` passandogli le specifiche dell'`Elementog` testo inserito. Infine, richiama `addInsert()` di `Loader` passandogli l'identificativo del testo inserito. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di undo/redo da `Command`, per cui, il metodo si occuperà solamente di aggiornare la view.
- **inserisciImmagini(files<sub>g</sub>, spec)**
    - **Accesso:** Public;
    - **Tipo di ritorno:** Void;
    - **Descrizione:** metodo che prima richiama `isImage(frames)` di `Upload` per controllare che le estensioni<sub>g</sub> siano corrette, successivamente `uploadmedia(filesg, callback)` di `Upload` per il caricamento dei `Fileg` immagine nel `Serverg`. Se l'operazione ha successo, viene invocato `callback()` il quale inserisce ogni immagine nel piano della presentazione<sub>g</sub>, attraverso la `Funzioneg javascript inserisciImmagine(percorso_Fileg, spec)` di `Edit`, e richiama, utilizzando il metodo `execute` di `inv`, `ConcreteImageInsertCommand()` di `Command` passandogli le specifiche degli elementi<sub>g</sub> immagine inseriti. Infine, richiama `addInsert()` di `Loader` passandogli l'identificativo dell'immagine inserita. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di undo/redo da `Command`, per cui, il metodo si occuperà solamente di aggiornare la view.
  - **inserisciAudio(files<sub>g</sub>, spec)**
    - **Accesso:** Public;
    - **Tipo di ritorno:** Void;
    - **Descrizione:** metodo che prima richiama `isAudio(frames)` di `Upload` per controllare che le estensioni<sub>g</sub> siano corrette, successivamente `uploadmedia(filesg, callback)` di `Upload` per il caricamento dei `Fileg` audio nel `Serverg`. Se l'operazione ha successo, viene invocato `callback()` il quale inserisce ogni audio nel piano della presentazione<sub>g</sub>, attraverso la `Funzioneg JavaScriptg inserisciAudio(percorso_Fileg, spec)` di `Edit`, e richiama, utilizzando il metodo `execute` di `inv`, `ConcreteAudioInsertCommand()` di `Command` passandogli le specifiche degli elementi<sub>g</sub> audio inseriti. Infine, richiama `addInsert()` di `Loader` passandogli l'identificativo dell'audio inserito. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di undo/redo da `Command`, per cui, il metodo si occuperà solamente di aggiornare la view.
  - **inserisciVideo(files<sub>g</sub>, spec)**
    - **Accesso:** Public;
    - **Tipo di ritorno:** Void;

- **Descrizione:** metodo che prima richiama `isVideo(frames)` di `Upload` per controllare che le estensioni<sub>g</sub> siano corrette, successivamente `uploadmedia(filesg, callback)` di `Upload` per il caricamento dei `Fileg` video nel `Serverg`. Se l'operazione ha successo, viene invocato `callback()` il quale inserisce ogni video nel piano della presentazione<sub>g</sub>, attraverso la Funzione<sub>g</sub> `JavaScriptg` `inserisciVideo(percorso_Fileg, spec)` di `Edit`, e richiama, utilizzando il metodo `execute` di `inv`, `ConcreteVideoInsertCommand()` di `Command` passandogli le specifiche degli elementi<sub>g</sub> video inseriti. Infine, richiama `addInsert()` di `Loader` passandogli l'identificativo del video inserito. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di `undo/redo` da `Command`, per cui, il metodo si occuperà solamente di aggiornare la `view`.
- **dragMedia(files<sub>g</sub>, spec)**
  - **Accesso:** `Public`;
  - **Tipo di ritorno:** `Void`;
  - **Descrizione:** metodo apposito per la gestione del `drag and drop` di `Fileg` media all'interno della presentazione. Gestisce l'inserimento di immagini, audio e video richiamando una tra le seguenti funzioni<sub>g</sub> `JavaScriptg` di `Edit`:
    - \* `inserisciImmagine(percorso_Fileg, spec);`
    - \* `inserisciAudio(percorso_Fileg, spec);`
    - \* `inserisciVideo(percorso_Fileg, spec).`e successivamente uno tra i seguenti metodi di `Command`, dandolo in pasto al metodo `execute()` di `inv`:
    - \* `ConcreteImageInsertCommand;`
    - \* `ConcreteAudioInsertCommand;`
    - \* `ConcreteVideoInsertCommand.`Infine, richiama `addInsert()` di `Loader` passandogli l'identificativo del video inserito.
- **getMediaSpec(ele, tipo, URL<sub>g</sub>)**
  - **Accesso:** `Private`;
  - **Tipo di ritorno:** `Object`;
  - **Descrizione:** metodo che ritorna le specifiche dell'Elemento<sub>g</sub> `ele` inserito nel piano della presentazione<sub>g</sub>, da passare al `Command`. Tale metodo viene richiamato dai metodi adibiti all'inserimento di immagini, audio o video.
- **rimuoviElemento(spec:object)**
  - **Accesso:** `Public`;
  - **Tipo di ritorno:** `Void`;
  - **Descrizione:** metodo che rimuove l'Elemento<sub>g</sub> corrente dal piano della presentazione<sub>g</sub>, richiamando la Funzione<sub>g</sub> `JavaScriptg` `elimina(id_Elementog)` di `Edit` e successivamente, in base al tipo dell'Elemento<sub>g</sub>, utilizzando il metodo `execute` di `inv`, richiama uno tra i seguenti:

- ```
* ConcreteTextRemoveCommand;
* ConcreteFrameRemoveCommand;
* ConcreteImageRemoveCommand;
* ConcreteAudioRemoveCommand;
* ConcreteVideoRemoveCommand.
```

Infine, richiama `addDelete()` di `Loader` passandogli l'identificativo dell'Elemento<sub>g</sub> eliminato. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di `undo/redo` da `Command`, per cui, il metodo si occuperà solamente di aggiornare la `view`.

- `updateSfondo(spec:object)`

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che viene richiamato solo in seguito ad una operazione di undo/redo e che per questo, si occupa solamente di aggiornare il background della presentazione nella view.

- **cambiaColoreSfondo(color)**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che assegna al background della presentazione il valore color, passato come parametro. Successivamente richiama, utilizzando la Funzione<sub>g</sub> execute di inv, ConcreteBackgroundInsertCommand() di Command passandogli le specifiche del background. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> background modificato.

- `cambiaImmagineSfondo(filesg)`

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che assegna al background della presentazione un nuovo sfondo in base al parametro `files_g`. Successivamente richiama, utilizzando la Funzione `execute di inv, ConcreteBackgroundInsertCommand()` di `Command` passandogli le specifiche del background. Infine, richiama `addUpdate()` di `Loader` passandogli l'identificativo dell'Elemento `g` background modificato.

- `rimuoviSfondo()`

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che rimuove colore e sfondo dal background e che successivamente, utilizzando la Funzione<sub>g</sub> execute di inv, richiama ConcreteBackgroundInsertCommand() di Command passandogli le specifiche del background. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> background modificato.

- **updateSfondoFrame(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che viene richiamato solo in seguito ad una operazione di undo/redo e che per questo, si occupa solamente di aggiornare il background del Frame<sub>g</sub> spec.id nella view.
- **cambiaColoreSfondoFrame(color)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che assegna al background del Frame<sub>g</sub> selezionato il valore color, passato come parametro. Successivamente richiama, utilizzando la Funzione<sub>g</sub> execute di inv, ConcreteEditBackgroundColorCommand() di Command passandogli le specifiche del background del Frame<sub>g</sub>. Infine, richiama addUpdate() di Loader passandogli l'identificativo del Frame<sub>g</sub> modificato.
- **cambiaImmagineSfondoFrame(files<sub>g</sub>)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che assegna al background del Frame<sub>g</sub> selezionato un nuovo sfondo in base al parametro files<sub>g</sub>. Successivamente richiama, utilizzando la Funzione<sub>g</sub> execute di inv, ConcreteEditBackgroundColorCommand() di Command passandogli le specifiche del background. Infine, richiama addUpdate() di Loader passandogli l'identificativo del Frame<sub>g</sub> modificato.
- **rimuoviSfondoFrame()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che rimuove colore e sfondo dal background del Frame<sub>g</sub> selezionato e che successivamente, utilizzando la Funzione<sub>g</sub> execute di inv, richiama ConcreteEditBackgroundColorCommand() di Command passandogli le specifiche del background. Infine, richiama addUpdate() di Loader passandogli l'identificativo del Frame<sub>g</sub> modificato.
- **cambiaColoreTesto(color)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che cambia il colore dei caratteri dell'Elemento<sub>g</sub> testo selezionato, in base al parametro color, e che successivamente, utilizzando la Funzione<sub>g</sub> execute di inv, richiama ConcreteEditColorCommand() di Command passandogli le specifiche del colore. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> testo modificato.

- **cambiaSizeTesto**(value)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che cambia la dimensione dei caratteri dell'Elemento<sub>g</sub> testo selezionato, in base al parametro value, e che successivamente, utilizzando la Funzione<sub>g</sub> execute di inv, richiama ConcreteEditFontCommand() di Command passandogli le specifiche del testo modificato. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> testo modificato.
- **cambiaFontTesto**(Font<sub>g</sub>)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che cambia la dimensione dei caratteri dell'Elemento<sub>g</sub> testo selezionato, in base al parametro Font<sub>g</sub>, e che successivamente, utilizzando la Funzione<sub>g</sub> execute di inv, richiama ConcreteEditFontCommand() di Command passandogli le specifiche del testo modificato. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> testo modificato.
- **aggiornaTesto**(textId, textContent, spec)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che cambia il contenuto dell'Elemento<sub>g</sub> testo textId, in base al parametro textContent, e che successivamente, utilizzando la Funzione<sub>g</sub> execute di inv, richiama ConcreteEditContentCommand() di Command passandogli le specifiche del testo modificato. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> testo modificato. Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Command, per cui, il metodo si occuperà solamente di aggiornare la view.
- **mediaControl**()
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che attiva le funzionalità per la riproduzione di un File<sub>g</sub> media.
- **ruotaElemento**(value, spec)
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che ruota l'Elemento<sub>g</sub> selezionato in base al parametro value richiamando la Funzione<sub>g</sub> JavaScript<sub>g</sub> rotate(id\_Elemento<sub>g</sub>, value) di Edit. Successivamente richiama, utilizzando la Funzione<sub>g</sub> execute di inv, ConcreteEditRotationCommand() di Command passandogli le specifiche della nuova rotazione. Infine,

richiama `addUpdate()` di `Loader` passandogli l'identificativo dell'`Elemento_g` modificato. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di `undo/redo` da `Command`, per cui, il metodo si occuperà solamente di aggiornare la `view`.

- **muoviElemento(spec:object)**

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che, in base al nuovo posizionamento dell'Elemento<sub>g</sub> selezionato all'interno del piano della presentazione<sub>g</sub>, richiama, utilizzando la Funzione<sub>g</sub> execute di inv, ConcreteEditPositionCommand() di Command passandogli le specifiche della nuova posizione. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> modificato. Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Command, per cui, il metodo si occuperà solamente di aggiornare la view.

- **ridimensionaElemento**(spec:object)

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che, in base alla nuova dimensione dell'Elemento<sub>g</sub> selezionato all'interno del piano della presentazione<sub>g</sub>, richiama, utilizzando la Funzione<sub>g</sub> execute di inv, ConcreteEditSizeCommand(spec) di Command passandogli le specifiche della nuova dimensione. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> modificato. Nel caso in cui il parametro spec sia definito, significa che è stata inviata una richiesta di undo/redo da Command, per cui, il metodo si occuperà solamente di aggiornare la view.

- `aggiungiMainPath(spec:object)`

- **Accesso:** Public;
- **Tipo di ritorno:** Void;
- **Descrizione:** metodo che aggiunge il  $\text{Frame}_g$  selezionato al  $\text{Percorso}_g$  principale salvato nell'oggetto  $\text{mainPath}$  di  $\text{Edit}$ , e che richiama, utilizzando la  $\text{Funzione}_g$   $\text{execute}$  di  $\text{inv}$ ,  $\text{AddToMainPathCommand}()$  di  $\text{Command}$  passandogli le specifiche del  $\text{Frame}_g$  da aggiungere al  $\text{Percorso}_g$  principale. Infine, richiama  $\text{addPaths}()$  di  $\text{Loader}$ . Nel caso in cui il parametro  $\text{spec}$  sia definito, significa che è stata inviata una richiesta di undo/redo da  $\text{Command}$ , per cui, il metodo si occuperà solamente di aggiornare la view.

- `rimuoviMainPath(spec:object)`

- **Accesso:** Public;
- **Tipo di ritorno:** Void;



- **Descrizione:** metodo che rimuove il `Frameg` selezionato dal `Percorsog` principale salvato nell'oggetto `mainPath` di `Edit`, e che richiama, utilizzando la `Funzioneg` `execute` di `inv`, `RemoveFromMainPathCommand()` di `Command` passandogli le specifiche del `Frameg` da togliere dal `Percorsog` principale. Infine, richiama `addPaths()` di `Loader`. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di undo/redo da `Command`, per cui, il metodo si occuperà solamente di aggiornare la view.
- **inPath(id)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Boolean;
  - **Descrizione:** metodo che controlla se un frame è all'interno del percorso principale attraverso il metodo `getPaths()` di `InsertEditRemove`. Ritorna true se il frame è nel path, altrimenti false.
- **portaAvanti(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che aggiorna il valore `zIndex` dell'`Elementog` correntemente selezionato attraverso la `Funzioneg` `JavaScriptg` `portaAvanti(id_Elementog)` di `Edit`, e che richiama, utilizzando la `Funzioneg` `execute` di `inv`, `concretePortaAvantiCommand()` di `Command` passandogli le specifiche con l'`Elementog` da aggiornare. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di undo/redo da `Command`, per cui, il metodo si occuperà solamente di aggiornare la view.
- **portaDietro(spec:object)**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che aggiorna il valore `zIndex` dell'`Elementog` correntemente selezionato attraverso la `Funzioneg` `JavaScriptg` `portaDietro(id_Elementog)` di `Edit`, e che richiama, utilizzando la `Funzioneg` `execute` di `inv`, `concretePortaDietroCommand()` di `Command` passandogli le specifiche con l'`Elementog` da aggiornare. Nel caso in cui il parametro `spec` sia definito, significa che è stata inviata una richiesta di undo/redo da `Command`, per cui, il metodo si occuperà solamente di aggiornare la view.
- **impostaPrimoSfondo()**
  - **Accesso:** Private;
  - **Tipo di ritorno:** Void;



- **Descrizione:** metodo che viene richiamato nel caso in cui la presentazione sia vuota, oppure se il valore background non è impostato. Esso richiama ConcreteBackgroundInsertCommand() di Command passandogli le dimensioni del background. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> background modificato.
- **updateBookmark(id)**
  - **Accesso:** Private;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che aggiorna il campo Bookmark<sub>g</sub> del Frame<sub>g</sub> id. Esso richiama ConcreteEditBookmarkCommand() di Command passandogli le specifiche del Frame<sub>g</sub>. Infine, richiama addUpdate() di Loader passandogli l'identificativo dell'Elemento<sub>g</sub> background modificato.
- **AddBookmark(spec:object)**
  - **Accesso:** Private;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che mantiene aggiornata l'icona del pulsante Bookmark<sub>g</sub> nella view, in base al parametro spec.
- **RemoveBookmark(spec:object)**
  - **Accesso:** Private;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che mantiene aggiornata l'icona del pulsante Bookmark<sub>g</sub> nella view, in base al parametro spec.
- **annullaModifica()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che annulla una modifica effettuata richiamando il metodo undo() di inv e che richiama uno tra i metodi addInsert, addUpdate, addDelete o addPaths di Loader in base al tipo di azione effettuata con l'undo.
- **ripristinaModifica()**
  - **Accesso:** Public;
  - **Tipo di ritorno:** Void;
  - **Descrizione:** metodo che ripristina una modifica annullata richiamando il metodo redo() di inv e che richiama uno tra i metodi addInsert, addUpdate, addDelete o addPaths di Loader in base al tipo di azione effettuata con il redo.
- **getUndoStack()**

- **Accesso:** Public;
  - **Tipo di ritorno:** Object;
  - **Descrizione:** metodo che ritorna lo stack dell'undo richiamando il metodo `getUndoStack()` di `inv`.
- **getRedoStack()**
    - **Accesso:** Public;
    - **Tipo di ritorno:** Object;
    - **Descrizione:** metodo che ritorna lo stack del redo richiamando il metodo `getRedoStack()` di `inv`.

## 7 Package Premi::View

Tutti i package seguenti appartengono al package Premi, quindi per ognuno di essi lo scope sarà: Premi::[nome package].

**Tipo, obiettivo e funzione del componente:** contiene le classi che istanzieranno gli oggetti per l'interfaccia grafica del Software<sub>g</sub>.

**Relazioni d'uso di altre componenti:** utilizza le classi contenute nel package Controller per le comunicazioni con il Model.

**Attività svolte e dati trattati:** rappresenta l'intera GUI del nostro sistema.

## 7.1 View::Pages

**Tipo, obiettivo e funzione del componente:** contiene le pagine in Html che rappresentano l'interfaccia grafica vera e propria.

**Relazioni d'uso di altre componenti:** utilizza le classi contenute nel package Controller.

**Attività svolte e dati trattati:** rappresenta le pagine fisiche del Software<sub>g</sub>. Sono realizzate utilizzando HTML5 e Angular Material per la grafica e il data-binding di Angular.js per mantenere la view aggiornata rispetto ai cambiamenti che avvengono nel Model.

## 7.2 View::Pages::Index

## Funzione

Questa pagina si occuperà di mostrare all'utente header e footer validi per ogni pagina html e permettendogli di accedere alle pagine Login<sub>g</sub>, Registrazione, Home e Profile e di poter effettuare il Logout<sub>g</sub> dal sistema .

## Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::HeaderController.

Input utente

- `bottoneAccedi()`: richiama `Controller::HeaderController::goLogin()` che reindirizza alla pagina `View::Pages::Login`;
- `bottoneRegistrati()`: richiama `Controller::HeaderController::goRegistrazione()` che reindirizza alla pagina `View::Pages::Registrazione`;
- `bottoneHome()`: richiama `Controller::HeaderController::goHome()` che reindirizza alla pagina `View::Pages::Home`;
- `bottoneProfilo()`: richiama `Controller::HeaderController::goProfile()` che reindirizza alla pagina `View::Pages::Profilo`;

## Data binding

- `data-ng-hide`: utilizzato per la comparsa/scomparsa dei bottoni di cui sopra, in base al metodo `isToken()` di `Controller::HeaderController`;
- `ng-click`: utilizzato dai bottoni di cui sopra per richiamare il rispettivo metodo di `Controller::HeaderController`.

### 7.3 View::Pages::Login

## Funzione

Questa pagina si occuperà di mostrare all'utente la possibilità di effettuare il Login.

## Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::AccessController.

Input utente

- `bottonoLogin()`: attiva il metodo `Controller::AccessController::login()` che controlla se i campi della form sono stati compilati correttamente. Se l'operazione ha successo, viene effettuato il reindirizzamento alla pagina `View::Pages::Home`;

## Data binding

- ng-click: utilizzato dal bottone di cui sopra per richiamare il rispettivo metodo di `Controller::AccessController`;
- ng-bind: utilizzato per mostrare gli eventuali errori accorsi in fase di autenticazione. L'errore è ricavato dagli attributi `usernameError` e `passwordError` di `Controller::AccessController`.

## 7.4 View::Pages::Registrazione

## Funzione

Questa pagina si occuperà di mostrare all'utente la possibilità di effettuare la registrazione al sistema.

## Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::AccessController.

Input utente

- `bottoneRegistrati()`: attiva il metodo `Controller::AccessController::registration()` che controlla se i campi della form sono stati compilati correttamente. Se l'operazione ha successo, viene effettuato il reindirizzamento alla pagina `View::Pages::Home`;

## Data binding

- ng-click: utilizzato dal bottone di cui sopra per richiamare il rispettivo metodo di `Controller::AccessController`;
- ng-bind: utilizzato per mostrare gli eventuali errori accorsi in fase di registrazione. L'errore è ricavato dagli attributi `usernameError` e `passwordError` di `Controller::AccessController`.

## 7.5 View::Pages::Home

## Funzione

Questa pagina si occuperà di mostrare all'utente le presentazioni salvate nel server dando la possibilità di eliminarle, eseguirle, scaricarle in locale, rinominarle, modificarle o crearne di nuove.

## Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::HomeController

Input utente

- `bottoneNuovaPresentazione()`: bottone che richiama il metodo di `Controller::HomeController::createSlideShow()` passandogli il nome della presentazione da creare;
- `bottoneElimina()`: bottone che richiama il metodo di `Controller::HomeController::deleteSlideShow()` passandogli il nome della presentazione da eliminare;
- `bottoneRinomina()`: bottone che richiama il metodo di `Controller::HomeController::renameSlideShow()` passandogli il nome della presentazione da rinominare;
- `bottoneEsegui()`: bottone che richiama il metodo di `Controller::HomeController::goExecute()` passandogli il nome della presentazione da eseguire;
- `bottoneEdit()`: bottone che richiama il metodo di `Controller::HomeController::goEdit()` passandogli il nome della presentazione da modificare;

## Data binding

- `ng-click`: utilizzato dai bottoni di cui sopra per richiamare il rispettivo metodo di `Controller::HomeController`;
- `ng-repeat`: utilizzato per stampare nella pagina tutte le presentazioni dell'utente presenti nell'oggetto ritornato da `getSS()` di `Controller::HomeController`;
- `ng-style`: utilizzato per fornire l'anteprima di ogni presentazione richiamando `setStyleSS()` di `Controller::HomeController` passandogli una delle presentazioni ricavate tramite `getSS()`.

## 7.6 View::Pages::Profile

## Funzione

Questa pagina si occuperà di mostrare all'utente la possibilità di cambiare i propri dati personali.

## Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::ProfileController

Input utente

- `bottoneCambiaPassword()`: bottone che richiama il metodo di `Controller::ProfileController::changePa`  
Il risultato dell'operazione viene in ogni caso comunicato all'utente.

## Data binding

- ng-submit: utilizzato dal bottone di cui sopra per richiamare il rispettivo metodo di Controller::ProfileController.

## 7.7 View::Pages::Execution

## Funzione

Questa pagina si occuperà di gestire l'esecuzione di una presentazione utilizzando il Framework\_g Impress.js associato alla pagina.

## Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::ExecutionController

Input utente

- `next()`: metodo che viene invocato premendo il tasto "freccia destra" e che invoca a sua volta il metodo `impress().next()` implementato all'interno del `Framework_g Impress.js` e che visualizzerà il `Frame_g` successivo della presentazione;
- `prev()`: metodo che viene invocato premendo il tasto "freccia sinistra" e che invoca a sua volta il metodo `impress().prev()` implementato all'interno del `Framework_g Impress.js` e che visualizzerà il `Frame_g` precedente della presentazione;
- `bookmark()`: metodo che viene invocato premendo il tasto "barra spaziatrice" e che invoca a sua volta il metodo `impress().bookmark()` implementato all'interno del `Framework_g Impress.js` e che visualizzerà il `Frame_g` con `Bookmark_g` successivo.

## 7.8 View::Pages::Edit

## Funzione

Questa pagina si occuperà di mostrare all'utente la possibilità di apportare modifiche ad una presentazione.

## Relazioni d'uso con altri moduli

Questa pagina utilizzerà le seguenti classi:

- Controller::EditController

## Attributi

- **active**: oggetto che rappresenta l'Elemento<sub>g</sub> attualmente selezionato. Metodi:
  - `getId()`: ritorna l'Elemento<sub>g</sub> selezionato;
  - `getTipo()`: ritorna il tipo dell'Elemento<sub>g</sub> selezionato;

- select(id): seleziona l'Elemento<sub>g</sub> id nel piano delle presentazioni;
- deselect(): deseleziona l'Elemento<sub>g</sub> attivo.
- mainPath: oggetto che rappresenta il Percorso<sub>g</sub> principale della presentazione. Metodi:
  - addToMainPath(id, position): metodo che aggiunge il Frame<sub>g</sub> id nella posizione position del Percorso<sub>g</sub> principale;
  - removeFromMainPath(id): metodo che elimina il Frame<sub>g</sub> id dal Percorso<sub>g</sub> principale;

Input utente

- `bottoneEseguiPresentazione()`: bottone che richiama `Controller::EditController::goExecute()` per eseguire la presentazione;
- `bottoneAnnulla`: bottone che richiama `annullaModifica()` di `EditController` per annullare l'ultima modifica eseguita;
- `bottoneRipristina`: bottone che richiama `ripristinaModifica()` di `EditController` per ripristinare l'ultima modifica annullata;
- `bottoneInserisciFrame`: bottone che richiama `inserisciFrame()` di `EditController` per inserire un `Frameg` nel piano della presentazione<sub>g</sub>;
- `bottoneInserisciTesto`: bottone che richiama `inserisciTesto()` di `EditController` per inserire un `Elementog` testo nel piano della presentazione<sub>g</sub>;
- `bottoneInserisciImmagine`: bottone che richiama `inserisciImmagine()` di `EditController` passandogli le immagini da inserire;
- `bottoneInserisciAudio`: bottone che richiama `inserisciAudio()` di `EditController` passandogli gli audio da inserire;
- `bottoneInserisciVideo`: bottone che richiama `inserisciVideo()` di `EditController` passandogli i video da inserire;
- `bottoneRuota`: bottone che richiama `ruotaElemento()` di `EditController` passandogli il valore della rotazione da applicare all'`Elementog` selezionato;
- `bottoneCambiaColoreSfondo`: bottone che richiama `cambiaColoreSfondo()` di `EditController` passandogli il valore del colore da applicare al background della presentazione;
- `bottoneCambiaImmagineSfondo`: bottone che richiama `cambiaImmagineSfondo()` di `EditController` passandogli l'immagine da applicare al background della presentazione;
- `eliminaSfondoPresentazione`: bottone che richiama `rimuoviSfondo()` di `EditController` per resettare lo sfondo della presentazione;
- `bottoneCambiaColoreSfondoFrame`: bottone che richiama `cambiaColoreSfondoFrame()` di `EditController` passandogli il valore del colore da applicare al `Frameg` selezionato;

- `bottoneCambiaImmagineSfondoFrame`: bottone che richiama `cambiaImmagineSfondoFrame()` di `EditController` passandogli l'immagine da applicare al background del `Frameg` selezionato;
- `bottoneCambiaFontTesto`: bottone che richiama `cambiaCambiaFontTesto()` di `EditController` passandogli il font da applicare all'elemento testo selezionato;
- `bottoneCambiaDimensioniTesto`: bottone che richiama `cambiaCambiaSizeTesto()` di `EditController` passandogli la dimensione da applicare all'elemento testo selezionato;
- `bottoneCambiaColoreTesto`: bottone che richiama `cambiaCambiaColoreTesto()` di `EditController` passandogli il colore del carattere da applicare all'elemento testo selezionato;
- `bottoneEliminaSfondoFrame`: bottone che richiama `rimuoviSfondoFrame()` di `EditController` per resettare lo sfondo del `Frameg` selezionato;
- `bottoneAggiungiPercorsoPrincipale`: bottone che richiama `aggiungiMainPath()` di `EditController` per aggiungere il `Frameg` corrente al `Percorsog` principale di presentazione;
- `bottoneRimuoviPercorsoPrincipale`: bottone che richiama `rimuoviMainPath()` di `EditController` per rimuovere il `Frameg` corrente dal `Percorsog` principale di presentazione;
- `bottonePortaAvanti()`: bottone che richiama `portaAvanti()` di `EditController` per cambiare il parametro z-index dell'`Elementog` selezionato;
- `bottonePortaDietro()`: bottone che richiama `portaDietro()` di `EditController` per cambiare il parametro z-index dell'`Elementog` selezionato;
- `bottoneBookmark()`: bottone che assegna o rimuove il `Bookmarkg` al `Frameg`, attiva il metodo di `updateBookmark()` di `EditController` che si occuperà dell'aggiornamento della presentazione;

## Data binding ed eventi HTML

- **ng-click**: utilizzato dai bottoni di cui sopra per richiamare il rispettivo metodo di `EditController`;
- **ng-show**: utilizzato per far comparire/scompare i bottoni in base allo stato corrente della presentazione. In particolare, viene utilizzato per i seguenti bottoni:
  - **Annulla e Ripristina**: richiamando i metodi `getUndoStack()` e `getRedoStack()` di `EditController`;
  - **Aggiungi Percorso Principale**: richiamando il metodo `inPath()` di `EditController`;
  - **Rimuovi Bookmark e Aggiungi Bookmark**: utilizzando rispettivamente gli attributi `canAddBookmark` e `canRemoveBookmark` di `EditController`.
- **ng-change**: utilizzato dai bottoni **Cambia Colore Sfondo**, **Cambia Colore Testo**, **Cambia Dimensione Testo** e **Cambia Font Testo** per richiamare i rispettivi metodi di `EditController`;
- **ng-repeat**: utilizzato dal bottone **Cambia Font Testo** per rappresentare tutti i font contenuti in `allFonts` di `EditController`;



- **onchange**: utilizzato dai bottoni `InserisciImmagine`, `InserisciAudio`, `InserisciVideo`, `CambiaImmagineSfondo` e `CambiaImmagineSfondoFrame` richiamando i metodi specificati sopra;
- **onblur**: utilizzato dagli elementi testo per richiamare `aggiornaTesto()` di `EditController` passandogli l'id dell'elemento e il contenuto del testo;
- **onresize**: utilizzato dagli elementi frame, immagine, video e testo per il ridimensionamento. Richiama `resizeDiv()` delle funzioni Javascript della pagina `Edit`;
- **Drag&Drop**: evento `JavaScriptg` che permette di poter spostare un `Elementog` all'interno del piano della presentazione<sub>g</sub> e che richiama `muoviElemento()` di `EditController`;
- **Resizable**: evento `JavaScriptg` che permette di poter ridimensionare un `Elementog` all'interno del piano della presentazione<sub>g</sub> e che richiama `ridimensionaElemento()` di `EditController`;

## Funzioni Javascript

- `inserisciFrame(spec)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che permette l'inserimento di un nuovo `Frameg`. Se `spec` è definito, le proprietà contenute in esso vengono assegnate al `Frameg` appena inserito;
- `inserisciTesto(spec)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che permette l'inserimento di un nuovo `Elementog` testo. Se `spec` è definito, le proprietà contenute in esso vengono assegnate al testo appena inserito;
- `inserisciImmagine(x, spec)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che permette l'inserimento di un nuovo `Elementog` immagine con `path` passato tramite il parametro `x`. Se `spec` è definito, le proprietà contenute in esso vengono assegnate all'immagine appena inserita;
- `inserisciAudio(x, spec)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che permette l'inserimento di un nuovo `Elementog` audio con `path` passato tramite il parametro `x`. Se `spec` è definito, le proprietà contenute in esso vengono assegnate all'audio appena inserito;
- `inserisciVideo(x, spec)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che permette l'inserimento di un nuovo `Elementog` video con `path` passato tramite il parametro `x`. Se `spec` è definito, le proprietà contenute in esso vengono assegnate al video appena inserito;
- `elimina(id)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che permette l'eliminazione dell'`Elementog` `id` dal piano della presentazione<sub>g</sub>;
- `rotate(el, value)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che permette di ruotare l'`Elementog` `el` in base al valore definito dal parametro `value`;
- `portaAvanti(id)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che incrementa la proprietà `zIndex` dell'`Elementog` `id`;
- `mandaDietro(id)`: Funzione<sub>g</sub> JavaScript<sub>g</sub> che decrementa la proprietà `zIndex` dell'`Elementog` `id`;



- `resizeDiv(element, type)`: metodo che aggiusta il ridimensionamento dell'elemento `element` di tipo `type`;
- `mediaControl()`: metodo che permette di riprodurre un `File_g` media o di fermarne la riproduzione;
- `zoom(element)`: metodo che permette di effettuare lo zoom sull'elemento `element`.

## 8 Tracciamento

## 8.1 tracciamento metodi-test

| Classe e Metodo                                                                            | Test |
|--------------------------------------------------------------------------------------------|------|
| serverRelation::Loader::addInsert(id:string)                                               | TU52 |
| serverRelation::Loader::addDelete(type:string,id:string)                                   | TU52 |
| serverRelation::Loader::addUpdate(id:string)                                               | TU52 |
| serverRelation::Loader::addpahts()                                                         | TU52 |
| serverRelation::Loader::update()                                                           | TU52 |
| serverRelation::MongoRelation::ServerRelation::Loader::addInsert()                         | TU51 |
| serverRelation::MongoRelation::deleteElement(presentation:string,type:string,id:string)    | TU50 |
| serverRelation::MongoRelation::updateElement(presentation:string,object,callback:function) | TU49 |
| serverRelation::MongoRelation::newElement(presentation:string,object,callback:function)    | TU48 |
| serverRelation::MongoRelation::renamePresentation(old:string,new:string)                   | TU47 |
| serverRelation::MongoRelation::deletePresentation(name:string)                             | TU46 |
| serverRelation::MongoRelation::getPresentation(name:string)                                | TU45 |
| serverRelation::MongoRelation::newCopyPresentation(name:string)                            | TU44 |
| serverRelation::MongoRelation::newPresentation(name:string)                                | TU43 |
| serverRelation::MongoRelation::getPresentationsMeta()                                      | TU42 |
| serverRelation::FileServerRelation::renameVideo(old:string,new:string)                     | TU41 |
| serverRelation::FileServerRelation::renameAudio(old:string,new:string)                     | TU40 |
| serverRelation::FileServerRelation::renameImage(old:string,new:string)                     | TU39 |
| serverRelation::FileServerRelation::deleteVideo(img:string)                                | TU38 |
| serverRelation::FileServerRelation::deleteAudio(img:string)                                | TU37 |
| serverRelation::FileServerRelationdeleteImage(img:string)                                  | TU36 |
| serverRelation::FileServerRelation::getImageDimension(img:string)                          | TU35 |
| serverRelation::FileServerRelation::getMetasVideos()                                       | TU34 |
| serverRelation::FileServerRelation::getMetasAudios()                                       | TU33 |
| serverRelation::FileServerRelation::getMetasImages()                                       | TU32 |
| Server_g::POST private/api/files/[video]/[newname]                                         | TU31 |

|                                                                                           |                  |
|-------------------------------------------------------------------------------------------|------------------|
| Server <sub>g</sub> ::POST private/api/files/[video]                                      | TU30             |
| Server <sub>g</sub> ::DELETE private/api/files/[video]                                    | TU29             |
| Server <sub>g</sub> ::GET private/api/files/video                                         | TU28             |
| Server <sub>g</sub> ::POST private/api/files/[audio]/[newname]                            | TU27             |
| Server <sub>g</sub> ::POST private/api/files/[audio]                                      | TU26             |
| Server <sub>g</sub> ::DELETE private/api/files/[audio]                                    | TU25             |
| Server <sub>g</sub> ::GET private/api/files/audio                                         | TU24             |
| Server <sub>g</sub> ::GET private/api/files/sizeImage/[image]                             | TU23             |
| Server <sub>g</sub> ::POST private/api/files/[image]/[newname]                            | TU22             |
| Server <sub>g</sub> ::POST private/api/files/[image]                                      | TU21             |
| Server <sub>g</sub> ::DELETE private/api/files/[image]                                    | TU20             |
| Server <sub>g</sub> ::GET private/api/files/image                                         | TU19             |
| Server <sub>g</sub> ::PUT private/api/presentations/[presname]/element                    | TU18             |
| Server <sub>g</sub> ::PUT private/api/presentations/[presname]/paths                      | TU17             |
| Server <sub>g</sub> ::POST private/api/presentations/[presname]/element                   | TU16             |
| Server <sub>g</sub> ::DELETE private/api/presentation-<br>s/[presname]/delete/[type]/[id] | TU15             |
| Server <sub>g</sub> ::PUT private/api/presentations/[oldname]/rename/[new]                | TU14             |
| Server <sub>g</sub> ::DELETE private/api/presentations/[presentation]                     | TU13             |
| Server <sub>g</sub> ::GET private/api/presentations/[presentation]                        | TU12             |
| Server <sub>g</sub> ::POST private/api/presentations/new/[oldpres]/[newpres]              | TU11             |
| Server <sub>g</sub> ::POST private/api/presentations/new/[newpres]                        | TU10             |
| Server <sub>g</sub> ::GET private/api/presentations                                       | TU9              |
| Server <sub>g</sub> ::POST Account <sub>g</sub> /changepassword                           | TU8, TU7,<br>TU6 |
| Server <sub>g</sub> ::GET Account <sub>g</sub> /authenticate                              | TU5, TU4,<br>TU3 |
| POST Account <sub>g</sub> /register                                                       | TU2, TU1         |
| Premi::Services::Main                                                                     | TU 53            |
| Premi::Services::Upload                                                                   | TU 53            |

|                             |       |
|-----------------------------|-------|
| Premi::Services::Utils      | TU 53 |
| Premi::Services::SharedData | TU 53 |
| Premi::Services::ToPages    | TU 53 |
| Premi::HeaderController     | TU 54 |
| Premi::AccessController     | TU 55 |
| Premi::HomeController       | TU 56 |
| Premi::ProfileController    | TU 57 |
| Premi::ExecutionController  | TU 58 |
| Premi::EditController       | TU 59 |

Tab 3: Metodi-Test Controller