NORRIS Framework

FLAMETECH Inc.

Definizione di Prodotto

Informazioni sul documento

Versione	1.0.0
Redazione	Cardin Andrea Merlo Gianluca Persegona Mattia
Verifica	Zanetti Davide
Responsabile	Meneguzzo Francesco
Uso	Esterno
Lista di distribuzione	**FlameTech Inc.** Prof. Vardanega Tullio Prof. Cardin Riccardo CoffeeStrap

Descrizione

Questo documento descrive la progettazione di dettaglio definita dal gruppo **FlameTech Inc.** relativa al progetto Norris.

Stato	Modifica	Autore	Ruolo	Data	Versione
Approvato	Approvazione documento	Meneguzzo Francesco	Responsabile	2015/05/01	1.0.0
Verificato	Verifica documento	Zanetti Davide	Verificatore	2015/05/01	0.1.0
In Lavorazione	Stesura tracciamento	Cardin Andrea	Progettista	2015/04/30	0.0.8
In Lavorazione	Termine stesura sezione 3	Merlo Gianluca	Progettista	2015/04/30	0.0.7
In Lavorazione	Termine stesura sezione 4	Persegona Mattia	Progettista	2015/04/30	0.0.6
In Lavorazione	Inizio stesura sezione 4	Persegona Mattia	Progettista	2015/04/29	0.0.5
In Lavorazione	Inizio stesura sezione 3	Merlo Gianluca	Progettista	2015/04/29	0.0.4
In Lavorazione	Stesura sezione Standard di Progetto	Persegna Mattia	Progettista	2015/04/28	0.0.3
In Lavorazione	Stesura sezione Introduzione	Cardin Andrea	Progettista	2015/04/28	0.0.2
In Lavorazione	Impostazione scheletro del documento	Persegona Mattia	Progettista	2015/04/28	0.0.1

Indice

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il presente documento ha lo scopo di definire la progettazione di dettaglio del progetto Norris.
Il documento si basa sulla *Specifica Tecnica v2.0.0*. I programmatori si serviranno di tale documento per procedere con le attività di codifica.

1.2 Scopo del prodotto

Lo scopo del prodotto è la realizzazione di un *framework*$_G$ per *Node.js*$_G$, compatibile con l'utilizzo standard dei *middleware*$_G$ di *Express*$_G$ in versione 4.x, per la realizzazione rapida di grafici aggiornabili in tempo reale.

1.3 Glossario

Per evitare ogni possibile ambiguità che potrebbe sorgere verrà allegato il *Glossario_ver4.0.0* dove verranno inseriti termini tecnici, acronimi, termini di dominio ed eventuali parole che potrebbero comportare delle incomprensioni o delle ambiguità nella lettura dei documenti. Per rendere la lettura più facile i termini verranno riportati in corsivo ed in pedice verrà posta una "G" maiuscola. (Esempio: *Android*$_G$).

1.4 Riferimenti

1.4.1 Normativi

- **Analisi dei Requisiti:** *AnalisiRequisiti_ver4.0.0*;
- **Norme di Progetto:** *NormeDiProgetto_ver3.0.0*;
- **Capitolato d'appalto C3 Norris:** Node Real-time Intelligence http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C3.pdf;

1.4.2 Informativi

- Presentazione capitolato d'appalto: http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C3ps.pdf;
- Ingegneria del software - Ian Sommerville - 9a edizione (2011), Parte terza: Advance Software Engineering, Capitolo 18.3: Architectural patterns for distributed systems;
- Design Patterns - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides - 1a edizione italiana (2008);

2 Standard di Progetto

2.1 Standard di progettazione architetturale

Gli standard di progettazione sono definiti nella *Specifica Tecnica v2.0.0*.
Per chiarezza, si evidenzia che in aggiunta al formalismo UML_G 2.0 è stata utilizzata una notazione ad hoc per rappresentare il tipo di dato di una funzione **function(nomeParametro:tipo):tipodiritorno** rappresenta quindi il tipo di dato di una funzione che richiede i parametri **nomeParametro:tipo** e che ritorna un oggetto di **tipodiritorno**.

2.2 Standard di documentazione del codice

Gli standard per la scrittura della documentazione del codice sono definiti nelle *Norme di Progetto v3.0.0*.

2.3 Standard di denominazione di entità e relazioni

Tutti gli elementi definiti come $package_G$, classi, metodi o attributi, devono avere denominazioni chiare ed esplicative. Il nome deve avere una lunghezza tale da non pregiudicarne la leggibilità e chiarezza. È preferibile utilizzare dei sostantivi per le entità e dei verbi per le relazioni. Le abbreviazioni sono ammesse se:

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

Le regole tipografiche relative ai nomi delle entità sono definite nelle *Norme di Progetto v3.0.0*.

2.4 Standard di programmazione

Gli standard di programmazione sono definiti e descritti nelle *Norme di Progetto v3.0.0*.

2.5 Strumenti di lavoro

Per gli strumenti di lavoro da utilizzare durante la codifica e le procedure per il loro corretto funzionamento e coordinamento si rimanda al documento *Norme di Progetto v3.0.0*.

3 Specifica classi del *back-end*$_G$

3.1 Componente Norris::Lib::BusinessLayer

3.1.1 Classe ActiveResourcesController

ActiveResourcesController
+ storeGraph(id:Integer, graph:Graph) + retrieveGraph(id:Integer) + retrievePage(id:Integer) + storePage(id:Integer, page:Page)

Tabella 2: Classe ActiveResourcesController

Descrizione

Questa classe comprende i metodi per gestire le pagine e i grafici attivi.

Utilizzo

Sarà utilizzata dalle classi del BusinessLayer per indicizzare e accedere alle risorse attive.

Relazioni con altre classi

- Norris::Lib::DataLayer:ActiveResources Relazione uscente. La classe ActiveResourcesContoller utilizza questa classe per indicizzare e accedere alle risorse attive;
- Norris::Lib::Utils:NorrisError Relazione uscente. La classe ActiveResourcesController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:BarChartController Relazione entrante. La classe BarChartController utilizza le funzionalità della classe ActiveResourcesController;
- Norris::Lib::BusinessLayer:LineChartController Relazione entrante. La classe LineChartController utilizza le funzionalità della classe ActiveResourcesController;
- Norris::Lib::BusinessLayer:MapChartController Relazione entrante. La classe MapChartController utilizza le funzionalità della classe ActiveResourcesController;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe ActiveResourcesController per memorizzare il riferimento ad un oggetto di tipo PageModel dopo averlo creato;
- Norris::Lib::BusinessLayer:TableController Relazione entrante. La classe TableController utilizza le funzionalità della classe ActiveResourcesController.

Attributi

Assenti.

Metodi

`+ storeGraph(id:Integer, graph:Graph):void`
Questo metodo è usato per salvare un riferimento ad un grafico quando questo viene creato.

- `id:Integer`
  Intero che contiene l'id di un grafico.
- `graph:Graph`
  Un oggetto di tipo BarChart, LineChart, MapChart o Table.

`+ retrieveGraph(id:Integer):Graph`
Ritorna l'oggetto Graph il cui id corrisponde all'id passato come parametro.

- `id:Integer`
  Intero che contiene l'id del grafico che verrà ritornato.

`+ retrievePage(id:Integer):Page`
Ritorna l'oggetto Page il cui id corrisponde all'id passato come parametro.

- `id:Integer`
  Intero che contiene l'id dell'oggetto di tipo Page che verrà ritornato.

`+ storePage(id:Integer, page:Page):void`
Questo metodo viene usato per salvare il riferimento ad un oggetto di tipo Page quando questo viene creato.

- `id:Integer`
  Intero che contiene l'id di un oggetto di tipo Page.
- `page:Page`
  Oggetto di tipo Page il cui riferimento verrà salvato.

3.1.2 Classe BarChartController

BarChartController

`+ createBarChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object)`
`+ getChartInfo(graphID:Integer)`
`+ updateInPlace(graphID:Integer, label:String, set:Integer, newValue:Integer)`

Tabella 3: Classe BarChartController

Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Bar Chart*$_G$.

Utilizzo

Sarà utilizzata da PresentationLayer::BarChart per l'utilizzo delle *API*$_G$ per la creazione e l'aggiornamento dei grafici di tipo *Bar Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:ActiveResourcesController Relazione uscente. La classe BarChartController utilizza le funzionalità della classe ActiveResourcesController;
- Norris::Lib::DataLayer:BarChartModel Relazione uscente. La classe BarChartContoller utilizza questa classe per la creazione di oggetti di tipo BarChartModel;
- Norris::Lib::Utils:ColorManager Relazione uscente. La classe BarChartController utilizza le funzionalità della classe ColorGenerator;
- Norris::Lib::BusinessLayer:DataConsistency Relazione uscente. La classe BarChartController utilizza le funzionalità della classe DataConsitency;
- Norris::Lib::Utils:NorrisError Relazione uscente. La classe BarChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::Utils:ProgressiveID Relazione uscente. La classe BarChartController utilizza le funzionalità della classe ProgressiveID;
- Norris::Lib::BusinessLayer:SocketController Relazione uscente. La classe BarChartController utilizza le funzionalità della classe SocketController;
- Norris::Lib::PresentationLayer:BarChart Relazione entrante. La classe BarChart utilizza le funzionalità della classe BarChartController;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe BarChartController per recuperare le informazioni di un oggetto di tipo BarChartModel.

Attributi

Assenti.

Metodi

`+ createBarChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object):Integer`
Questo metodo crea un oggetto di tipo BarChart e ritorna un intero contenente l'id del grafico appena creato.

- `title:String`
  Stringa che contiene il titolo del grafico.

- **xAxisName:String**
  Stringa che contiene l'etichetta dell'asse delle X.
- **yAxisName:String**
  Stringa che contiene l'etichetta dell'asse delle Y.
- **labels:Array**
  Array che contiene le etichette dei set di dati.
- **data:Array**
  Array che contiene i dati iniziali del grafico.
- **options:Object**
  Parametro opzionale, oggetto *JSON*$_G$ che contiene le opzioni aggiuntive da assegnare al grafico. Se non presente verranno settate le opzioni di default.

`+ getChartInfo(graphID:Integer):Object`
Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

- **graphID:Integer**
  Intero che contiene l'id del grafico di cui si desidera ottenere i dati.

`+ updateInPlace(graphID:Integer, label:String, set:Integer, newValue:Integer):void`
Aggiorna un dato del grafico con modalità in place.

- **graphID:Integer**
  Intero che contiene l'id del grafico che si vuole aggiornare.
- **label:String**
  Indica il nome della label del dato da cambiare.
- **set:Integer**
  Intero che indica l'incide della serie a cui appartiene il dato da cambiare.
- **newValue:Integer**
  Intero che contiene il nuovo valore da assegnare al dato.

3.1.3 Classe DataConsistency

DataConsistency
- checkTemplate(opt:String, template:Object) - checkOrientation(obj:String) - checkLegendPosition(obj:String) - checkMapLegendPosition(obj:String) - checkMapZoom(obj:String) - pathMode(obj:String) - checkOrderBy(obj:Object, template:Object) - checkInsertPosition(obj:String) - checkDisplayedLines(obj:String) - checkColors(obj:Array) - checkHex(obj:String) - checkColorArray(obj:Object, opt:String) - checkColorMatrix(obj:Object, opt:String) - checkSeries(obj:Array) - checkValueType(obj:String) - checkDecimals(obj:Integer) - checkTableFormat(obj:Array) - checkBounds(obj:Array, opt:String) - checkAllColors(obj:Array, opt:String) + jsonConsistencyCheck(obj:Object, template:Object) + inPlaceTableOptionsConsistency(obj:Object, template:Object) + streamTableOptionsConsistency(obj:Object, template:Object) + labelConsistency(labels:Array) + seriesConsistency(labels:Array, series:Array)

Tabella 4: Classe DataConsistency

Descrizione

Questa classe si occupa di effettuare i controlli sui dati .

Utilizzo

Sarà utilizzata allo scopo di verificare la forma e la consistenza dei dati passati dallo sviluppatore.

Relazioni con altre classi

- Norris::Lib::Utils:NorrisError Relazione uscente. La classe DataConsistency utilizza le funzionalità della classe NorrisError per la gestione degli errori;

- Norris::Lib::DataLayer:ActiveResources Relazione entrante. La classe DataConsistency utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:BarChartController Relazione entrante. La classe BarChartController utilizza le funzionalità della classe DataConsitency;
- Norris::Lib::BusinessLayer:LineChartController Relazione entrante. La classe LineChartController utilizza le funzionalità della classe DataConsitency;
- Norris::Lib::BusinessLayer:MapChartController Relazione entrante. La classe MapChartController utilizza le funzionalità della classe DataConsitency;
- Norris::Lib::BusinessLayer:TableController Relazione entrante. La classe TableController utilizza le funzionalità della classe DataConsitency.

Attributi

Assenti.

Metodi

`- checkTemplate(opt:String, template:Object):void`
Controlla che la proprietà opt sia presente nel template delle opzioni.

- **`opt:String`**
  Nome della proprietà da controllare.
- **`template:Object`**
  Template$_G$ delle opzioni in cui controllare la presenza di una proprietà.

`- checkOrientation(obj:String):void`
Controlla che l'opzione 'orientation' abbia un valore corretto.

- **`obj:String`**
  Valore della proprietà da controllare.

`- checkLegendPosition(obj:String):void`
Controlla che la proprietà 'legendPosition' dei grafici di tipo *Bar Chart*$_G$ e *Line Chart*$_G$ abbia un valore corretto.

- **`obj:String`**
  Valore della proprietà da controllare.

`- checkMapLegendPosition(obj:String):void`
Controlla che la proprietà 'legendPosition' dei grafici di tipo *Map Chart*$_G$ abbia un valore corretto.

- **`obj:String`**
  Valore della proprietà da controllare.

`- checkMapZoom(obj:String):void`
Controlla che la proprietà 'zoom' di un grafico di tipo *Map Chart*$_G$ abbia un valore corretto.

- **obj:String**
  Valore della proprietà da controllare.

- pathMode(obj:String):void
Controlla che la proprietà 'pathMode' abbia un valore corretto.

  - **obj:String**
    Valore della proprietà da controllare.

- checkOrderBy(obj:Object, template:Object):void
Controlla che la proprietà 'orderBy' dei grafici di tipo *Table*$_G$ abbia un valore corretto.

  - **obj:Object**
    Contiene le proprietà e i valori da controllare.
  - **template:Object**
    Template$_G$ delle opzioni in cui controllare la presenza di una proprietà.

- checkInsertPosition(obj:String):void
Controlla che la proprietà 'insertPosition' dei grafici di tipo *Table*$_G$ abbia un valore corretto.

  - **obj:String**
    Valore della proprietà da controllare.

- checkDisplayedLines(obj:String):void
Controlla che la proprietà 'displayedLines' dei grafici di tipo *Table*$_G$ abbia un valore corretto.

  - **obj:String**
    Valore della proprietà da controllare.

- checkColors(obj:Array):void
Controlla che la proprietà 'colors' sia un array contenente valori esadecimali validi.

  - **obj:Array**
    Array contenente i colori da controllare.

- checkHex(obj:String):void
Controlla che il colore passato sia un valore esadecimale valido.

  - **obj:String**
    Valore della proprietà da controllare.

- checkColorArray(obj:Object, opt:String):void
Controlla che le proprietà 'colorColumn', 'colorRow', 'colorColumnFont', 'colorRowFont' dei grafici di tipo *Table*$_G$ abbiano una forma corretta.

  - **obj:Object**
    Contiene le proprietà da controllare.
  - **opt:String**
    Identifica la proprietà da controllare.

- `checkColorMatrix(obj:Object, opt:String):void`
Controlla che le proprietà 'colorFont' e 'colorCell' dei grafici di tipo *Table*$_G$ abbiano una forma corretta.

  - **`obj:Object`**
    Contiene le proprietà da controllare.
  - **`opt:String`**
    Identifica la proprietà da controllare.

- `checkSeries(obj:Array):void`
Controlla che la proprietà 'series' abbia tipo corretto.

  - **`obj:Array`**
    Proprietà da controllare.

- `checkValueType(obj:String):void`
Controlla che la proprietà 'valueType' dei grafici di tipo *Bar Chart*$_G$ e *Line Chart*$_G$ abbia una forma corretta.

  - **`obj:String`**
    Valore della proprietà da controllare.

- `checkDecimals(obj:Integer):void`
Controlla che la proprietà 'decimals' dei grafici di tipo *Bar Chart*$_G$ e *Line Chart*$_G$ abbia una forma corretta.

  - **`obj:Integer`**
    Valore della proprietà da controllare.

- `checkTableFormat(obj:Array):void`
Controlla che la proprietà 'format' dei grafici di tipo *Table*$_G$ abbia una forma corretta.

  - **`obj:Array`**
    Contiene le proprietà da controllare.

- `checkBounds(obj:Array, opt:String):void`
Controlla che le proprietà 'labelsLimit' e 'rowsLimit' rispettivamente dei grafici di tipo *Line Chart*$_G$ e *Table*$_G$ abbiano un valore corretto.

  - **`obj:Array`**
    Contiene le opzioni da controllare.
  - **`opt:String`**
    Contiene il nome dell'opzione da controllare.

- `checkAllColors(obj:Array, opt:String):void`
Richiama tutti gli altri metodi che controllano le opzioni dei colori.

  - **`obj:Array`**
    Contiene le proprietà da controllare.
  - **`opt:String`**
    Contiene il nome della proprietà da controllare.

`+ jsonConsistencyCheck(obj:Object, template:Object):boolean`
Controlla che il parametro 'options' dei grafici abbia una forma corretta.

- **`obj:Object`**
  Oggetto che contiene tutte le opzioni aggiuntive del grafico.
- **`template:Object`**
  Oggetto che contiene un *template*$_G$ di opzioni con cui fare il confronto.

`+ inPlaceTableOptionsConsistency(obj:Object, template:Object):boolean`
Controlla che il parametro 'options' dell'update con modalità in place dei grafici di tipo *Table*$_G$ abbia una forma corretta.

- **`obj:Object`**
  Oggetto che contiene tutte le opzioni aggiuntive dell'update con modalità in place dei grafici di tipo *Table*$_G$.
- **`template:Object`**
  Oggetto che contiene un *template*$_G$ di opzioni con cui fare il confronto.

`+ streamTableOptionsConsistency(obj:Object, template:Object):boolean`
Controlla che il parametro 'options' dell'update con modalità stream dei grafici di tipo *Table*$_G$ abbia una forma corretta.

- **`obj:Object`**
  Oggetto che contiene tutte le opzioni aggiuntive dell'update con modalità stream dei grafici di tipo *Table*$_G$.
- **`template:Object`**
  Oggetto che contiene un *template*$_G$ di opzioni con cui fare il confronto.

`+ labelConsistency(labels:Array):boolean`
Il metodo controlla la correttezza del label.

- **`labels:Array`**
  Array contenente i label.

`+ seriesConsistency(labels:Array, series:Array):boolean`
Controlla se le dimensioni di labels coincide con quella di series.

- **`labels:Array`**
  Array contenente i label.
- **`series:Array`**
  Array contenente i nomi delle serie dei dati.

3.1.4 Classe LineChartController

LineChartController
+ createLineChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object) + updateStream(graphID:Integer, newLabel:String, newValues:Array) + getChartInfo(graphID:Integer) + updateInPlace(graphID:Integer, label:String, set:String, newValue:Array)

Tabella 5: Classe LineChartController

Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Line Chart*$_G$.

Utilizzo

Sarà utilizzata da PresentationLayer::LineChart per l'utilizzo delle *API*$_G$ per la creazione e l'aggiornamento dei grafici di tipo *Line Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:ActiveResourcesController Relazione uscente. La classe LineChartController utilizza le funzionalità della classe ActiveResourcesController;
- Norris::Lib::Utils:ColorManager Relazione uscente. La classe LineChartController utilizza le funzionalità della classe ColorGenerator;
- Norris::Lib::BusinessLayer:DataConsistency Relazione uscente. La classe LineChartController utilizza le funzionalità della classe DataConsitency;
- Norris::Lib::DataLayer:LineChartModel Relazione uscente. La classe LineChartContoller utilizza questa classe per la creazione di oggetti di tipo LineChartModel;
- Norris::Lib::Utils:NorrisError Relazione uscente. La classe LineChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::Utils:ProgressiveID Relazione uscente. La classe LineChartController utilizza le funzionalità della classe ProgressiveID;
- Norris::Lib::BusinessLayer:SocketController Relazione uscente. La classe LineChartController utilizza le funzionalità della classe SocketController;
- Norris::Lib::PresentationLayer:LineChart Relazione entrante. La classe LineChart utilizza le funzionalità della classe LineChartController;

- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe LineChartController per recuperare le informazioni di un oggetto di tipo LineChartModel.

Attributi

Assenti.

Metodi

`+ createLineChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object):Integer`
Questo metodo crea un oggetto di tipo LineChart e ritorna un intero contenente l'id del grafico appena creato.

- **`title:String`**
  Stringa che contiene il titolo del grafico.
- **`xAxisName:String`**
  Stringa che contiene l'etichetta dell'asse delle X.
- **`yAxisName:String`**
  Stringa che contiene l'etichetta dell'asse delle Y.
- **`labels:Array`**
  Array che contiene le etichette dei set di dati.
- **`data:Array`**
  Array che contiene i dati iniziali del grafico.
- **`options:Object`**
  Parametro opzionale, oggetto *JSON*$_G$ che contiene le opzioni aggiuntive da assegnare al grafico. Se non presente verranno settate le opzioni di default.

`+ updateStream(graphID:Integer, newLabel:String, newValues:Array):void`
Metodo che aggiunge una nuova label e un valore per ogni set di dati in modalità stream.

- **`graphID:Integer`**
  Intero che contiene l'id del grafico che si desidera aggiornare.
- **`newLabel:String`**
  Stringa che contiene il nome della label che verrà aggiunta.
- **`newValues:Array`**
  Array che contiene i nuovi valori per ogni set di dati.

`+ getChartInfo(graphID:Integer):Object`
Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

- **`graphID:Integer`**
  Intero che contiene l'id del grafico di cui si desidera ottenere i dati.

`+ updateInPlace(graphID:Integer, label:String, set:String, newValue:Array):void`
Metodo che aggiorna un dato del grafico con modalità in place.

- **`graphID:Integer`**
  Intero che contiene l'id del grafico che si vuole aggiornare.

- `label:String`
  Indica il nome della label del dato da cambiare.
- `set:String`
  Indica l'incide della serie a cui appartiene il dato da cambiare.
- `newValue:Array`
  Array che contiene i nuovi valori del dato.

3.1.5 Classe MapChartController

MapChartController

`+ createMapChart(title:String, paths:Array, points:Array, centerLatitude:Float, centerLongitude:Float, options:Object)`
`+ updateMovie(graphID:Integer, newPoints:Array)`
`+ getChartInfo(graphID:Integer)`
`+ updateInPlace(graphID:Integer, label:Array, latitude:Float, longitude:Float)`
`- pointsConsistency(points:Array)`

Tabella 6: Classe MapChartController

Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Map Chart*$_G$.

Utilizzo

Sarà utilizzata da PresentationLayer::MapChart per l'utilizzo delle *API*$_G$ per la creazione e l'aggiornamento dei grafici di tipo *Map Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:ActiveResourcesController Relazione uscente. La classe MapChartController utilizza le funzionalità della classe ActiveResourcesController;
- Norris::Lib::Utils:ColorManager Relazione uscente. La classe MapChartController utilizza le funzionalità della classe ColorGenerator;
- Norris::Lib::BusinessLayer:DataConsistency Relazione uscente. La classe MapChartController utilizza le funzionalità della classe DataConsitency;
- Norris::Lib::DataLayer:MapChartModel Relazione uscente. La classe MapChartContoller utilizza questa classe per la creazione di oggetti di tipo MapChartModel;

- Norris::Lib::Utils:NorrisError Relazione uscente. La classe MapChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::Utils:ProgressiveID Relazione uscente. La classe MapChartController utilizza le funzionalità della classe ProgressiveID;
- Norris::Lib::BusinessLayer:SocketController Relazione uscente. La classe MapChartController utilizza le funzionalità della classe SocketController;
- Norris::Lib::PresentationLayer:MapChart Relazione entrante. La classe MapChart utilizza le funzionalità della classe MapChartController;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe MapChartController per recuperare le informazioni di un oggetto di tipo MapChartModel.

Attributi

Assenti.

Metodi

`+ createMapChart(title:String, paths:Array, points:Array, centerLatitude:Float, centerLongitude:Float, options:Object):Integer`
Costruttore della classe MapChartController.

- `title:String`
  Stringa che contiene il titolo del grafico.
- `paths:Array`
  Array che contiene i percorsi da tracciare sulla mappa.
- `points:Array`
  Array che contiene le coordinate dei punti da visualizzare sulla mappa.
- `centerLatitude:Float`
  Latitudine del punto centrale della mappa.
- `centerLongitude:Float`
  Longitudine del punto centrale della mappa.
- `options:Object`
  Oggetto che contiene le opzioni aggiuntive da assegnare al grafico.

`+ updateMovie(graphID:Integer, newPoints:Array):void`
Aggiorna i punti della mappa con modalità movie.

- `graphID:Integer`
  Intero che contiene l'id del grafico che si vuole aggiornare.
- `newPoints:Array`
  Array contenente i nuovi punti.

`+ getChartInfo(graphID:Integer):Object`
Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

- `graphID:Integer`
  Intero che contiene l'id del grafico di cui si desidera ottenere i dati.

`+ updateInPlace(graphID:Integer, label:Array, latitude:Float, longitude:Float):voi`
Aggiorna un punto della mappa con modalità in place.

- **graphID:Integer**
  Intero che contiene l'id del grafico che si vuole aggiornare.
- **label:Array**
  Array contenente i label.
- **latitude:Float**
  Nuova latitudine del punto da modificare.
- **longitude:Float**
  Nuova longitudine del punto da modificare.

`- pointsConsistency(points:Array):boolean`
Controlla che le etichette non siano contengano duplicati.

- **points:Array**
  Array di punti da controllare.

3.1.6 Classe PageController

PageController

`+ createPage(title:String, options:Object)`
`+ addGraphToPage(pageID:Integer, graph:Graph)`
`+ getPageInfo(pageID:Integer)`
`- createPageOptions(options:Object)`

Tabella 7: Classe PageController

Descrizione

Questa classe comprende i metodi per la creazione delle pagine.

Utilizzo

Sarà utilizzata da PresentationLayer::Page per l'utilizzo delle API_G per la creazione delle pagine.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:ActiveResourcesController Relazione uscente. La classe PageController utilizza le funzionalità della classe ActiveResourcesController per memorizzare il riferimento ad un oggetto di tipo PageModel dopo averlo creato;
- Norris::Lib::PresentationLayer:BarChart Relazione uscente. La classe PageController utilizza le funzionalità della classe BarChart per verificare che la tipologia

di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina;

- Norris::Lib::BusinessLayer:BarChartController Relazione uscente. La classe PageController utilizza le funzionalità della classe BarChartController per recuperare le informazioni di un oggetto di tipo BarChartModel;
- Norris::Lib::DataLayer:BarChartModel Relazione uscente. La classe PageController utilizza le funzionalità della classe BarChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe BarChartController;
- Norris::Lib::PresentationLayer:LineChart Relazione uscente. La classe PageController utilizza le funzionalità della classe LineChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina;
- Norris::Lib::BusinessLayer:LineChartController Relazione uscente. La classe PageController utilizza le funzionalità della classe LineChartController per recuperare le informazioni di un oggetto di tipo LineChartModel;
- Norris::Lib::DataLayer:LineChartModel Relazione uscente. La classe PageController utilizza le funzionalità della classe LineChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe LineChartController;
- Norris::Lib::PresentationLayer:MapChart Relazione uscente. La classe PageController utilizza le funzionalità della classe MapChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina;
- Norris::Lib::BusinessLayer:MapChartController Relazione uscente. La classe PageController utilizza le funzionalità della classe MapChartController per recuperare le informazioni di un oggetto di tipo MapChartModel;
- Norris::Lib::DataLayer:MapChartModel Relazione uscente. La classe PageController utilizza le funzionalità della classe MapChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe MapChartController;
- Norris::Lib::Utils:NorrisError Relazione uscente. La classe PageController utilizza le funzionalità della classe NorrisError per la gestione degli errori ;
- Norris::Lib::DataLayer:PageModel Relazione uscente. La classe PageController utilizza questa classe per la creazione di oggetti di tipo PageModel;
- Norris::Lib::Utils:ProgressiveID Relazione uscente. La classe PageController utilizza le funzionalità della classe ProgressiveID per assegnare il codice ID univoco durante la creazione di un oggetto di tipo PageModel;
- Norris::Lib::PresentationLayer:Table Relazione uscente. La classe PageController utilizza le funzionalità della classe Table per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina;

- Norris::Lib::BusinessLayer:TableController Relazione uscente. La classe PageController utilizza le funzionalità della classe TableController per recuperare le informazioni di un oggetto di tipo TableModel;
- Norris::Lib::DataLayer:TableModel Relazione uscente. La classe PageController utilizza le funzionalità della classe TableModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe TableController;
- Norris::Lib::PresentationLayer:Page Relazione entrante. La classe Page utilizza le funzionalità della classe PageController.

Attributi

Assenti.

Metodi

`+ createPage(title:String, options:Object):Integer`
Crea un oggetto di tipo Page e ritorna l'id assegnato all'oggetto appena creato.

- **`title:String`**
  Stringa che contiene il titolo da assegnare all'oggetto Page che si va a creare.
- **`options:Object`**
  Contiene le opzioni aggiuntive da assegnare alla pagina.

`+ addGraphToPage(pageID:Integer, graph:Graph):void`
Questo metodo aggiunge un grafico ad una pagina.

- **`pageID:Integer`**
  Intero che contiene l'id dell'oggetto Page a cui si vuole aggiungere il grafico.
- **`graph:Graph`**
  Il grafico che si vuole aggiungere alla pagina.

`+ getPageInfo(pageID:Integer):Object`
Metodo che ritorna tutte le informazioni sulla pagina: attributi e grafici contenuti.

- **`pageID:Integer`**
  Intero che contiene l'id dell'oggetto Page di cui si vogliono le informazioni.

`- createPageOptions(options:Object):Object`
Questo metodo controlla che le opzioni fornite per la creazione di una pagina siano corrette, e ritorna delle opzioni completate con valori di default nel caso le originali fossero mancanti completamente o parzialmente.

- **`options:Object`**
  Contiene le opzioni aggiuntive da controllare.

3.1.7 Classe SocketController

SocketController
+ setSocket(nsp:Object) + sendUpdate(room:String, info:Object) + socketNamespace()

Tabella 8: Classe SocketController

Descrizione

Questa classe si occupa di gestire le richieste *socket*$_G$ della libreria Norris..

Utilizzo

La classe verrà utilizzata allo scopo di inviare gli aggiornamenti tramite *Socket.io*$_G$ ai vari *client*$_G$..

Relazioni con altre classi

- Norris::Lib::Utils:NorrisError Relazione uscente. La classe SocketController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::Utils:SocketService Relazione uscente. La classe SocketController utilizza le funzionalità della classe SocketService;
- Norris::Lib::DataLayer:ActiveResources Relazione entrante. La classe SocketController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:BarChartController Relazione entrante. La classe BarChartController utilizza le funzionalità della classe SocketController;
- Norris::Lib::BusinessLayer:LineChartController Relazione entrante. La classe LineChartController utilizza le funzionalità della classe SocketController;
- Norris::Lib::BusinessLayer:MapChartController Relazione entrante. La classe MapChartController utilizza le funzionalità della classe SocketController;
- Norris::Lib::PresentationLayer:Norris Relazione entrante. La classe Norris utilizza le funzionalità della classe SocketController per configurare il *namespace*$_G$;
- Norris::Lib::PresentationLayer:PageRouter Relazione entrante. La classe PageRouter utilizza le funzionalità della classe SocketController.

Attributi

Assenti.

Metodi

`+ setSocket(nsp:Object):void`
Questo metodo imposta il *namespace*$_G$ di *Socket.io*$_G$ che verrà utilizzato dalla libreria Norris e avvia il gestore di connessioni tra il *server*$_G$ e il *client*$_G$.

- `nsp:Object`
  Il *namespace*$_G$ di *Socket.io*$_G$ che verrà utilizzato dalla libreria Norris.

`+ sendUpdate(room:String, info:Object):void`
Questo metodo invia un *JSON*$_G$ dal *server*$_G$ al *client*$_G$.

- `room:String`
  Contiene l'id della room.
- `info:Object`
  I dati che vengono inviati al *client*$_G$ per l'aggiornamento.

`+ socketNamespace():String`
Questo metodo ritorna il *namespace*$_G$ di *Socket.io*$_G$ correntemente utilizzato da Norris.

3.1.8 Classe TableController

TableController
+ createTable(title:String, header:Array, data:Array, options:Object) + getChartInfo(graphID:Integer) + updateStream(graphID:Integer, data:Array, options:Object) + updateInPlace(graphID:Integer, row:Integer, column:Integer, newValue:Array, options:Object) - buildTempl() - fillDefaultOpts(tableOptions:Object) - fillDevOpts(options:Object, tableOptions:Object, columns:Integer) - fillDefaultColorOpts(tableOptions:Object, rows:Integer, columns:Integer) - fillDevBgColorOpts(options:Object, tableOptions:Object, rows:Integer, columns:Integer) - fillDevFontColorOpts(options:Object, tableOptions:Object, rows:Integer, columns:Integer) - fillDevUpdateOpts(options:Object, columns:Integer, fontColor:String, cellColor:String, rowColor:String, rowFontColor:String) - dataModelPush(_tableModel:Table, data:Array, fontColor:String, cellColor:String, rowColor:String, rowFontColor:String) - fillClientColors(_tableModel:Table, columns:Integer) - removeRow(_tableModel:Table) - getColors(graphID:Integer, rows:Integer, columns:Integer)

Tabella 9: Classe TableController

Descrizione

Questa classe comprende i metodi per la creazione e l'aggiornamento dei grafici di tipo *Table*$_G$.

Utilizzo

Sarà utilizzata da PresentationLayer::Table per l'utilizzo delle *API*$_G$ per la creazione e l'aggiornamento dei grafici di tipo *Table*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:ActiveResourcesController Relazione uscente. La classe TableController utilizza le funzionalità della classe ActiveResourcesController;

- Norris::Lib::BusinessLayer:DataConsistency Relazione uscente. La classe TableController utilizza le funzionalità della classe DataConsitency;
- Norris::Lib::Utils:NorrisError Relazione uscente. La classe TableController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::Utils:ProgressiveID Relazione uscente. La classe TableController utilizza le funzionalità della classe ProgressiveID;
- Norris::Lib::DataLayer:TableModel Relazione uscente. La classe TableContoller utilizza questa classe per la creazione di oggetti di tipo TableModel;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe TableController per recuperare le informazioni di un oggetto di tipo TableModel;
- Norris::Lib::PresentationLayer:Table Relazione entrante. La classe Table utilizza le funzionalità della classe TableController.

Attributi

Assenti.

Metodi

`+ createTable(title:String, header:Array, data:Array, options:Object):Integer`
Questo metodo crea un oggetto di tipo Table e ritorna un intero contenente l'id della tabella appena creata.

- **`title:String`**
  Stringa che contiene il titolo della tabella.
- **`header:Array`**
  Array che contiene gli header delle colonne della tabella.
- **`data:Array`**
  Array che contiene i dati iniziali della tabella.
- **`options:Object`**
  Parametro opzionale, oggetto *JSON*$_G$ che contiene le opzioni aggiuntive da assegnare alla tabella. Se non presente verranno settate le opzioni di default.

`+ getChartInfo(graphID:Integer):Object`
Metodo che ritorna tutte le informazioni sulla tabella: attributi, dati ed opzioni.

- **`graphID:Integer`**
  Intero che contiene l'id del grafico di cui si vogliono le informazioni.

`+ updateStream(graphID:Integer, data:Array, options:Object):void`
Aggiunge una nuova riga alla tabella in modalità stream.

- **`graphID:Integer`**
  Intero che contiene l'id della tabella che si vuole aggiornare.
- **`data:Array`**
  Array che contiene tutti i dati della nuova riga che verrà aggiunta alla tabella.

- **options:Object**
  Opzioni aggiuntive.

`+ updateInPlace(graphID:Integer, row:Integer, column:Integer, newValue:Array, options:Object):void`
Aggiorna una cella della tabella in modalità in place.

- **graphID:Integer**
  Intero che contiene l'id del grafico che si vuole aggiornare.
- **row:Integer**
  Intero che indica il numero di riga della cella da aggiornare.
- **column:Integer**
  Intero che indica il numero di colonna della cella da aggiornare.
- **newValue:Array**
  Array che contiene i nuovi valori della cella da aggiornare.
- **options:Object**
  Opzioni aggiuntive.

`- buildTempl():Object`
Costruisce un *template*$_G$ per le opzioni aggiuntive dei grafici di tipo *Table*$_G$.

`- fillDefaultOpts(tableOptions:Object):void`
Riempie il t*emplate*$_G$ passato come parametro con le opzioni di default.

- **tableOptions:Object**
  Template$_G$ delle opzioni aggiuntive di un grafico di tipo *Table*$_G$.

`- fillDevOpts(options:Object, tableOptions:Object, columns:Integer):void`
Riempie il *template*$_G$ di opzioni aggiuntive di un grafico di tipo *Table*$_G$ con le opzioni impostate dallo sviluppatore.

- **options:Object**
  Opzioni impostate dallo sviluppatore.
- **tableOptions:Object**
  Template$_G$ di opzioni aggiuntive di un grafico di tipo *Table*$_G$ da riempire.
- **columns:Integer**
  Numero di colonne nella tabella.

`- fillDefaultColorOpts(tableOptions:Object, rows:Integer, columns:Integer):void`
Riempie 'tableOptions' con le opzioni di colore di default.

- **tableOptions:Object**
  Template$_G$ di opzioni aggiuntive di un grafico di tipo *Table*$_G$ da riempire.
- **rows:Integer**
  Numero delle righe della tabella.
- **columns:Integer**
  Numero di colonne della tabella.

`- fillDevBgColorOpts(options:Object, tableOptions:Object, rows:Integer, columns:Integer):void`
Riempie 'tableOptions' con le opzioni 'cellBackGroundColor' impostate dallo sviluppatore.

- `options:Object`
  Opzioni impostate dallo sviluppatore.
- `tableOptions:Object`
  Template$_G$ di opzioni aggiuntive di un grafico di tipo *Table*$_G$ da riempire.
- `rows:Integer`
  Numero di righe della tabella.
- `columns:Integer`
  Numero di colonne della tabella.

`- fillDevFontColorOpts(options:Object, tableOptions:Object, rows:Integer, columns:Integer):void`
Riempie 'tableOptions' con con le opzioni di colore del font impostate dallo sviluppatore.

- `options:Object`
  Opzioni impostate dallo sviluppatore.
- `tableOptions:Object`
  Template di opzioni aggiuntive di un grafico di tipo *Table*$_G$ da riempire.
- `rows:Integer`
  Numero di righe della tabella.
- `columns:Integer`
  Numero di colonne della tabella.

`- fillDevUpdateOpts(options:Object, columns:Integer, fontColor:String, cellColor:String, rowColor:String, rowFontColor:String):void`
Riempie gli array di colori con le opzioni impostate dallo sviluppatore.

- `options:Object`
  Opzioni da riempire.
- `columns:Integer`
  Numero di colonne della tabella.
- `fontColor:String`
  Colore del font.
- `cellColor:String`
  Colore di una cella.
- `rowColor:String`
  Colore delle celle di una riga della tabella.
- `rowFontColor:String`
  Colore del font di una riga della tabella.

`- dataModelPush(_tableModel:Table, data:Array, fontColor:String, cellColor:String, rowColor:String, rowFontColor:String):void`
Inserisce i nuovi dati nel modello di un grafico di tipo *Table*$_G$.

- `_tableModel:Table`
  Oggetto di tipo Table.
- `data:Array`
  Valori della nuova riga da inserire nella tabella.

- **fontColor:String**
  Opzione 'fontColor' della nuova riga da inserire nella tabella.
- **cellColor:String**
  Opzione 'cellColor' della nuova riga da inserire nella tabella.
- **rowColor:String**
  Opzione 'rowColor' della nuova riga da inserire nella tabella.
- **rowFontColor:String**
  Opzione 'rowFontColor' della nuova riga da inserire nella tabella.

`- fillClientColors(_tableModel:Table, columns:Integer):Array`
Costruisce l'array dei colori per il *client*$_G$.

- **_tableModel:Table**
  Oggetto di tipo Table.
- **columns:Integer**
  Numero di colonne della tabella.

`- removeRow(_tableModel:Table):void`
Rimuove una riga dalla tabella.

- **_tableModel:Table**
  Oggetto di tipo Table.

`- getColors(graphID:Integer, rows:Integer, columns:Integer):Array`
Crea un array di colori.

- **graphID:Integer**
  ID del grafico.
- **rows:Integer**
  Numero di righe della tabella.
- **columns:Integer**
  Numero di colonne della tabella.

3.2 Componente Norris::Lib::DataLayer

3.2.1 Classe ActiveResources

ActiveResources
+ graphs:Array + pages:Array
+ ActiveResources()

Tabella 10: Classe ActiveResources

Descrizione

Questa classe rappresenta le risorse attive, siano esse pagine o grafici.

Utilizzo

Sarà utilizzata da BusinessLayer::ActiveResourcesController per indicizzare e accedere alle risorse attive.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:DataConsistency Relazione uscente. La classe DataConsistency utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:SocketController Relazione uscente. La classe SocketController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:ActiveResourcesController Relazione entrante. La classe ActiveResourcesContoller utilizza questa classe per indicizzare e accedere alle risorse attive.

Attributi

`+ graphs:Array`
Contiene i riferimenti ai grafici attivi in una determinata istanza di Norris;

`+ pages:Array`
Contiene i riferimenti alle pagine attive in una determinata istanza di Norris.

Metodi

`+ ActiveResources():Object`
Restituisce i riferimenti ai due array contenenti i riferimenti alle risorse attive in una determinata istanza di Norris.

3.2.2 Classe BarChartModel

BarChartModel
- _id:Integer - _title:String - _xAxisName:String - _yAxisName:String - _labels:Array - _data:Array - _orientation:String - _showGrid:String - _showLegend:String - _legendPosition:String - _colors:Array - _valueType:String - _decimals:Integer - _series:Array
+ BarChartModel(id:Integer, title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object)

Tabella 11: Classe BarChartModel

Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Bar Chart*$_G$.

Utilizzo

Sarà utilizzata da BusinessLayer::BarChartController per la creazione di grafici di tipo *Bar Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:BarChartController Relazione entrante. La classe BarChartContoller utilizza questa classe per la creazione di oggetti di tipo BarChartModel;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe BarChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe BarChartController.

Attributi

`- _id:Integer`
Codice ID univoco che identifica l'oggetto ;

- `_title:String`
Titolo del grafico;

- `_xAxisName:String`
Nome dell'asse delle ascisse;

- `_yAxisName:String`
Nome dell'asse delle ordinate;

- `_labels:Array`
Array che contiene i valori che verranno riportati nell"asse delle ascisse;

- `_data:Array`
Array che contiene le serie di valori che verranno visualizzati nell'asse delle ordinate;

- `_orientation:String`
Indica l'orientamento delle barre del grafico, può assumere il valore vertical o horizontal ;

- `_showGrid:String`
Indica se la griglia del grafico è visibile o meno, può assumere il valore shown o hidden;

- `_showLegend:String`
Indica se la legenda del grafico è visibile o meno, può assumere il valore shown o hidden;

- `_legendPosition:String`
Indica la posizione della legenda del grafico, può assumere il valore right, left, bottom o top;

- `_colors:Array`
Indica i colori delle serie di dati del grafico;

- `_valueType:String`
Indica se i valori del grafico rappresentano un tipo di valuta, può assumere valore euro, dollars, pounds o null;

- `_decimals:Integer`
Indica quante cifre decimali vengono visualizzate per ogni valore, può assumere valore da 0 a 6;

- `_series:Array`
Indica i nomi delle serie di dati presenti nel grafico.

Metodi

`+ BarChartModel(id:Integer, title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object):BarChartModel`
Costruttore della classe BarChartModel.

- `id:Integer`
  Intero che contiene l'id del grafico.

- **title:String**
  Stringa che contiene il titolo del grafico.
- **xAxisName:String**
  Stringa che contiene l'etichetta dell'asse delle X.
- **yAxisName:String**
  Stringa che contiene l'etichetta dell'asse delle Y.
- **labels:Array**
  Array che contiene le etichette dei set di dati.
- **data:Array**
  Array che contiene i dati iniziali del grafico.
- **options:Object**
  Oggetto che contiene le opzioni aggiuntive da assegnare al grafico.

3.2.3 Classe LineChartModel

LineChartModel
- _id:Integer - _title:String - _xAxisName:String - _yAxisName:String - _labels:Array - _data:Array - _showGrid:String - _showLegend:String - _legendPosition:String - _colors:Array - _valueType:String - _decimals:Integer - _labelsLimit:Integer - _series:Array
+ LineChartModel(id:Integer, title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object)

Tabella 12: Classe LineChartModel

Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Line Chart*$_G$.

Utilizzo

Sarà utilizzata da BusinessLayer::LineChartController per la creazione e la modifica dei grafici di tipo *Line Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:LineChartController Relazione entrante. La classe LineChartContoller utilizza questa classe per la creazione di oggetti di tipo LineChartModel;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe LineChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe LineChartController.

Attributi

`- _id:Integer`
Codice ID univoco che identifica l'oggetto;

`- _title:String`
Titolo del grafico;

`- _xAxisName:String`
Nome dell'asse delle ascisse;

`- _yAxisName:String`
Nome dell'asse delle ordinate;

`- _labels:Array`
Array che contiene i valori che verranno riportati nell"asse delle ascisse;

`- _data:Array`
Array che contiene le serie di valori che verranno visualizzati nell'asse delle ordinate;

`- _showGrid:String`
Indica se la griglia del grafico è visibile o meno, può assumere il valore shown o hidden;

`- _showLegend:String`
Indica se la legenda del grafico è visibile o meno, può assumere il valore shown o hidden;

`- _legendPosition:String`
Indica la posizione della legenda del grafico, può assumere il valore right, left, bottom o top;

`- _colors:Array`
Indica i colori delle serie di dati del grafico;

`- _valueType:String`
Indica se i valori del grafico rappresentano un tipo di valuta, può assumere valore euro, dollars, pounds o null;

`- _decimals:Integer`
Indica quante cifre decimali vengono visualizzate per ogni valore, può assumere valore da 0 a 6;

- `_labelsLimit:Integer`
Indica la lunghezza massima delle serie di valori, può assumere valore maggiore di zero;

- `_series:Array`
Indica i nomi delle serie di dati presenti nel grafico.

Metodi

`+ LineChartModel(id:Integer, title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object):LineChartModel`
Costruttore della classe LineChartModel.

- `id:Integer`
  Intero che contiene l'id del grafico.
- `title:String`
  Stringa che contiene il titolo del grafico.
- `xAxisName:String`
  Stringa che contiene l'etichetta dell'asse delle X.
- `yAxisName:String`
  Stringa che contiene l'etichetta dell'asse delle Y.
- `labels:Array`
  Array che contiene le etichette dei set di dati.
- `data:Array`
  Array che contiene i dati iniziali del grafico.
- `options:Object`
  Oggetto *JSON*$_G$ che contiene le opzioni aggiuntive da assegnare al grafico.

3.2.4 Classe MapChartModel

MapChartModel
- _id:Integer - _title:String - _points:Array - _paths:Array - _center:Object - _colors:Array - _zoom:Integer - _showLegend:String - _mapLegendPosition:String - _pathName:Array - _pathMode:String
+ MapChartModel(id:Integer, title:String, paths:Array, points:Array, center:Object, options:Object)

Tabella 13: Classe MapChartModel

Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Map Chart*$_G$.

Utilizzo

Sarà utilizzata da BusinessLayer::MapChartController per la creazione e la modifica dei grafici di tipo *Map Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:MapChartController Relazione entrante. La classe MapChartContoller utilizza questa classe per la creazione di oggetti di tipo MapChartModel;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe MapChartModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe MapChartController.

Attributi

- _id:Integer
Codice ID univoco che identifica l'oggetto;

- _title:String
Titolo del grafico;

- _points:Array
Array che contiene i punti che verranno rappresentati sul grafico;

- _paths:Array
Array che contiene i percorsi che verranno rappresentati sul grafico;

- _center:Object
JSON$_G$ che contiene i valori di latitudine e longitudine del punto centrale della mappa;

- _colors:Array
Indica i colori dei percorsi del grafico;

- _zoom:Integer
Indica il livello di zoom della mappa e può assumere un valore compreso tra 0 e 19;

- _showLegend:String
Indica se la legenda del grafico è visibile o meno, può assumere il valore shown o hidden;

- _mapLegendPosition:String
Indica la posizione della legenda del grafico, può assumere il valore top-right, top-left, bottom-right o bottom-left;

- _pathName:Array
Array che contiene i nomi dei percorsi che verranno rappresentati sulla legenda;

- _pathMode:String
Stringa che identifica il metodo di calcolo del percorso.

Metodi

+ MapChartModel(id:Integer, title:String, paths:Array, points:Array, center:Object, options:Object):MapChartModel
Costruttore della classe MapChartModel.

- id:Integer
  Intero che contiene l'id del grafico.
- title:String
  Stringa che contiene il titolo del grafico.
- paths:Array
  Array che contiene i percorsi da tracciare sulla mappa.
- points:Array
  Array che contiene le coordinate dei punti da visualizzare sulla mappa.
- center:Object
  Oggetto che contiene latitudine e longitudine del punto centrale della mappa.
- options:Object
  Oggetto che contiene le opzioni aggiuntive da assegnare al grafico.

3.2.5 Classe PageModel

PageModel
- _id:Integer - _title:String - _data:Array - _pageWidth:Integer - _columns:Integer
+ PageModel(id:Integer, title:String)

Tabella 14: Classe PageModel

Descrizione

Questa classe rappresenta il modello dei dati di una pagina.

Utilizzo

Sarà utilizzata da BusinessLayer::PageController per la creazione delle pagine.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza questa classe per la creazione di oggetti di tipo PageModel.

Attributi

`- _id:Integer`
Codice ID univoco che identifica l'oggetto;

`- _title:String`
Titolo della pagina;

`- _data:Array`
Array contenente gli ID dei grafici associati a quella pagina;

`- _pageWidth:Integer`
Indica la larghezza di visualizzazione della pagina *HTML*$_G$ in pixel, può assumere un valore maggiore di 800;

`- _columns:Integer`
Indica il numero massimo di grafici per riga, può assumere un valore tra 1 e 12.

Metodi

`+ PageModel(id:Integer, title:String):PageModel`
Costruttore della classe PageModel.

- `id:Integer`
  Intero che contiene l'id della pagina.
- `title:String`
  Stringa che contiene il titolo della pagina.

3.2.6 Classe TableModel

TableModel
- _id:Integer - _title:String - _headers:Array - _data:Array - _insertPosition:String - _orderBy:Object - _displayedLines:Integer - _showBorder:String - _colorColumn:Array - _colorRow:Array - _colorCell:Array
+ TableModel(id:Integer, title:String, headers:Array, data:Array, options:Object)

Tabella 15: Classe TableModel

Descrizione

Questa classe rappresenta il modello dei dati di un grafico di tipo *Table*$_G$.

Utilizzo

Sarà utilizzata da BusinessLayer::TableController per la creazione e la modifica dei grafici di tipo *Table*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe TableModel per verificare che la tipologia di un oggetto sia corretta per potervi invocare un metodo della classe TableController;
- Norris::Lib::BusinessLayer:TableController Relazione entrante. La classe TableContoller utilizza questa classe per la creazione di oggetti di tipo TableModel.

Attributi

`- _id:Integer`
Codice ID univoco che identifica l'oggetto;

`- _title:String`
Titolo del grafico;

`- _headers:Array`
Array che contiene i valori che verranno riportati negli header delle colonne;

`- _data:Array`
Array che contiene i valori che verranno visualizzati nelle celle della tabella;

`- _insertPosition:String`
Indica il punto in cui inserire i nuovi dati, può assumere il valore top o bottom;

`- _orderBy:Object`
Indica il criterio di ordinamento dei dati della tabella e la colonna di riferimento;

`- _displayedLines:Integer`
Indica il massimo numero di elementi visualizzati in ogni pagina della tabella;

`- _showBorder:String`
Indica se i bordi della tabella sono visibili o meno, può assumere il valore shown o hidden;

`- _colorColumn:Array`
Array che contiene l'indice della colonna e il colore da impostare;

`- _colorRow:Array`
Array che contiene l'indice della riga e il colore da impostare;

`- _colorCell:Array`
Array che contiene oggetti *JSON*$_G$, i quali contengono gli indici di riga e colonna, e il colore da impostare.

Metodi

`+ TableModel(id:Integer, title:String, headers:Array, data:Array, options:Object):`
Costruttore della classe TableModel.

- **`id:Integer`**
  Intero che contiene l'id della tabella.
- **`title:String`**
  Stringa che contiene il titolo della tabella.
- **`headers:Array`**
  Array che contiene gli header delle colonne della tabella.
- **`data:Array`**
  Array che contiene i dati iniziali della tabella.
- **`options:Object`**
  Oggetto *JSON*$_G$ che contiene le opzioni aggiuntive da assegnare alla tabella.

3.3 Componente Norris::Lib::PresentationLayer

3.3.1 Classe BarChart

BarChart

`+ BarChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object)`
`+ getChartInfo()`
`+ updateInPlace(label:String, set:Integer, newValue:Integer)`

Tabella 16: Classe BarChart

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Bar Chart*$_G$.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Bar Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:BarChartController Relazione uscente. La classe BarChart utilizza le funzionalità della classe BarChartController;
- Norris::Lib::PresentationLayer:Norris Relazione entrante. La classe NorrisIndex importa le funzioni di BarChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto BarChart;

- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe BarChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

Attributi

Assenti.

Metodi

`+ BarChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object):BarChart`
Costruttore della classe BarChart.

- **`title:String`**
  Stringa che contiene il titolo del grafico.
- **`xAxisName:String`**
  Stringa che contiene l'etichetta dell'asse delle X.
- **`yAxisName:String`**
  Stringa che contiene l'etichetta dell'asse delle Y.
- **`labels:Array`**
  Array che contiene le etichette dei set di dati.
- **`data:Array`**
  Array che contiene i dati iniziali del grafico.
- **`options:Object`**
  Parametro opzionale, oggetto *JSON*$_G$ che contiene le opzioni aggiuntive da assegnare al grafico. Se non presente verranno settate le opzioni di default.

`+ getChartInfo():Object`
Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

`+ updateInPlace(label:String, set:Integer, newValue:Integer):void`
Aggiorna un dato del grafico con modalità in place.

- **`label:String`**
  Indica il nome della label del dato da cambiare.
- **`set:Integer`**
  Intero che indica l'incide della serie a cui appartiene il dato da cambiare.
- **`newValue:Integer`**
  Intero che contiene il nuovo valore da assegnare al dato.

3.3.2 Classe LineChart

LineChart
+ LineChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object) + getChartInfo() + updateInPlace(label:String, set:Integer, newValue:Array) + updateStream(newLabel:String, newValue:Array)

Tabella 17: Classe LineChart

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Line Chart*$_G$.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Line Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:LineChartController Relazione uscente. La classe LineChart utilizza le funzionalità della classe LineChartController;
- Norris::Lib::PresentationLayer:Norris Relazione entrante. La classe NorrisIndex importa le funzioni di LineChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto LineChart;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe LineChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

Attributi

Assenti.

Metodi

`+ LineChart(title:String, xAxisName:String, yAxisName:String, labels:Array, data:Array, options:Object):LineChart`
Costruttore della classe LineChart.

- `title:String`
  Stringa che contiene il titolo del grafico.

- `xAxisName:String`
  Stringa che contiene l'etichetta dell'asse delle X.
- `yAxisName:String`
  Stringa che contiene l'etichetta dell'asse delle Y.
- `labels:Array`
  Array che contiene le etichette dei set di dati.
- `data:Array`
  Array che contiene i dati iniziali del grafico.
- `options:Object`
  Parametro opzionale, oggetto *JSON*$_G$ che contiene le opzioni aggiuntive da assegnare al grafico. Se non presente verranno settate le opzioni di default.

`+ getChartInfo():Object`
Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

`+ updateInPlace(label:String, set:Integer, newValue:Array):void`
Aggiorna un dato del grafico con modalità in place.

- `label:String`
  Indica il nome della label del dato da cambiare.
- `set:Integer`
  Indica l'incide della serie a cui appartiene il dato da cambiare.
- `newValue:Array`
  Array che contiene i nuovi valori del dato.

`+ updateStream(newLabel:String, newValue:Array):void`
Aggiunge una nuova label e un valore per ogni set di dati con modalità stream.

- `newLabel:String`
  Stringa che contiene il nome della label che verrà aggiunta.
- `newValue:Array`
  Array che contiene i nuovi valori per ogni set di dati.

3.3.3 Classe MapChart

MapChart
+ MapChart(title:String, paths:Array, points:Array, centerLatitude:Float, centerLongitude:Float, options:Object) + getChartInfo() + updateInPlace(point:Array, latitude:Float, longitude:Float) + updateMovie(newPoints:Array)

Tabella 18: Classe MapChart

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Map Chart*$_G$.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Map Chart*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:MapChartController Relazione uscente. La classe MapChart utilizza le funzionalità della classe MapChartController;
- Norris::Lib::PresentationLayer:Norris Relazione entrante. La classe NorrisIndex importa le funzioni di MapChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto MapChart;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe MapChart per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

Attributi

Assenti.

Metodi

`+ MapChart(title:String, paths:Array, points:Array, centerLatitude:Float, centerLongitude:Float, options:Object):MapChart`
Costruttore della classe MapChart.

- `title:String`
  Stringa che contiene il titolo del grafico.
- `paths:Array`
  Array che contiene i percorsi da tracciare sulla mappa.
- `points:Array`
  Array che contiene le coordinate dei punti da visualizzare sulla mappa.
- `centerLatitude:Float`
  Latitudine del punto centrale della mappa.
- `centerLongitude:Float`
  Longitudine del punto centrale della mappa.
- `options:Object`
  Oggetto che contiene le opzioni aggiuntive da assegnare al grafico.

`+ getChartInfo():Object`
Metodo che ritorna tutte le informazioni sul grafico: attributi, dati ed opzioni.

`+ updateInPlace(point:Array, latitude:Float, longitude:Float):void`
Aggiorna un punto della mappa con modalità in place.

- `point:Array`
  Array contenente i nuovi punti.
- `latitude:Float`
  Nuova latitudine del punto da modificare.
- `longitude:Float`
  Nuova longitudine del punto da modificare.

`+ updateMovie(newPoints:Array):void`
Aggiorna i punti della mappa con modalità movie.

- `newPoints:Array`
  Array contenente i nuovi punti.

3.3.4 Classe Norris

Norris
`+ Norris(nsp:Socket)`

Tabella 19: Classe Norris

Descrizione

Questa classe fornisce un punto d'accesso allo sviluppatore alla libreria Norris..

Utilizzo

Sarà utilizzata dallo sviluppatore per poter creare ed utilizzare gli altri oggetti forniti dalla libreria Norris. La classe, inoltre, istanzierà un subrouting sul punto di mount designato dallo sviluppatore..

Relazioni con altre classi

- Norris::Lib::PresentationLayer:BarChart Relazione uscente. La classe NorrisIndex importa le funzioni di BarChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto BarChart;
- Norris::Lib::PresentationLayer:LineChart Relazione uscente. La classe NorrisIndex importa le funzioni di LineChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto LineChart;
- Norris::Lib::PresentationLayer:MapChart Relazione uscente. La classe NorrisIndex importa le funzioni di MapChart per dare accesso allo sviluppatore alle funzionalità dell'oggetto MapChart;
- Norris::Lib::PresentationLayer:Page Relazione uscente. La classe NorrisIndex importa le funzioni di Page per dare accesso allo sviluppatore alle funzionalità dell'oggetto Page;

- Norris::Lib::PresentationLayer:PageRouter Relazione uscente. La classe NorrisIndex utilizza le funzioni di PageRouter per la creazione di un subrouting interno a Norris;
- Norris::Lib::BusinessLayer:SocketController Relazione uscente. La classe Norris utilizza le funzionalità della classe SocketController per configurare il *namespace*$_G$;
- Norris::Lib::PresentationLayer:Table Relazione uscente. La classe NorrisIndex importa le funzioni di Table per dare accesso allo sviluppatore alle funzionalità dell'oggetto Table.

Attributi

Assenti.

Metodi

`+ Norris(nsp:Socket):Object`
Costruttore della classe Norris.

- `nsp:Socket`
  Il *namespace*$_G$ di *Socket.io* che verrà utilizzato dalla libreria Norris.

3.3.5 Classe Page

Page

`+ Page(title:String, options:Object)`
`+ getPageInfo()`
`+ addGraph(graph:Graph)`

Tabella 20: Classe Page

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione alle pagine.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione delle pagine.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:PageController Relazione uscente. La classe Page utilizza le funzionalità della classe PageController;
- Norris::Lib::PresentationLayer:Norris Relazione entrante. La classe NorrisIndex importa le funzioni di Page per dare accesso allo sviluppatore alle funzionalità dell'oggetto Page.

Attributi

Assenti.

Metodi

`+ Page(title:String, options:Object):Page`
Costruttore della classe Page.

- `title:String`
  Stringa che contiene il titolo della nuova pagina.
- `options:Object`
  Oggetto che contiene le opzioni aggiuntive da assegnare alla pagina.

`+ getPageInfo():Object`
Metodo che ritorna tutte le informazioni sulla pagina: attributi e grafici contenuti.

`+ addGraph(graph:Graph):void`
Aggiunge un grafico alla pagina.

- `graph:Graph`
  Grafico di tipo *Bar Chart*$_G$, *Line Chart*$_G$, *Map Chart*$_G$ o *Table*$_G$ che si vuole aggiungere alla pagina.

3.3.6 Classe PageRouter

PageRouter
+ PageRouter(page:Page)

Tabella 21: Classe PageRouter

Descrizione

Questa classe gestisce la creazione e l'utilizzo del subrouting.

Utilizzo

La classe verrà utilizzata per la creazione di un subrouting, sul punto di mount designato dallo sviluppatore, che si occuperà di gestire le richieste GET dei *client*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:SocketController Relazione uscente. La classe PageRouter utilizza le funzionalità della classe SocketController;
- Norris::Lib::PresentationLayer:Norris Relazione entrante. La classe NorrisIndex utilizza le funzioni di PageRouter per la creazione di un subrouting interno a Norris.

Attributi

Assenti.

Metodi

`+ PageRouter(page:Page):express`
Questo metodo crea un oggetto PageRouter.

- `page:Page`
  Un oggetto di tipo Page.

3.3.7 Classe Table

Table

`+ Table(title:String, headers:Array, data:Array, options:Object)`
`+ getChartInfo()`
`+ updateInPlace(row:Integer, column:Integer, newValue:Array)`
`+ updateStream(data:Array)`

Tabella 22: Classe Table

Descrizione

Questa classe presenta le operazioni che verranno effettuate dallo sviluppatore in relazione ai grafici di tipo *Table*$_G$.

Utilizzo

Sarà utilizzata dallo sviluppatore per la creazione e la gestione dei grafici di tipo *Table*$_G$.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:TableController Relazione uscente. La classe Table utilizza le funzionalità della classe TableController;
- Norris::Lib::PresentationLayer:Norris Relazione entrante. La classe NorrisIndex importa le funzioni di Table per dare accesso allo sviluppatore alle funzionalità dell'oggetto Table;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe Table per verificare che la tipologia di un oggetto sia corretta nel momento in cui un grafico viene aggiunto ad una pagina.

Attributi

Assenti.

Metodi

`+ Table(title:String, headers:Array, data:Array, options:Object):Table`
Costruttore della classe Table.

- `title:String`
  Stringa che contiene il titolo della tabella.
- `headers:Array`
  Array che contiene gli header delle colonne della tabella.
- `data:Array`
  Array che contiene i dati iniziali della tabella.
- `options:Object`
  Parametro opzionale, oggetto *JSON*$_G$ che contiene le opzioni aggiuntive da assegnare alla tabella. Se non presente verranno settate le opzioni di default.

`+ getChartInfo():Object`
Metodo che ritorna tutte le informazioni sulla tabella: attributi, dati ed opzioni.

`+ updateInPlace(row:Integer, column:Integer, newValue:Array):void`
Aggiorna una cella della tabella in modalità in place.

- `row:Integer`
  Intero che indica il numero di riga della cella da aggiornare.
- `column:Integer`
  Intero che indica il numero di colonna della cella da aggiornare.
- `newValue:Array`
  Array che contiene i nuovi valori della cella da aggiornare.

`+ updateStream(data:Array):void`
Aggiunge una nuova riga alla tabella in modalità stream.

- `data:Array`
  Array che contiene tutti i dati della nuova riga che verrà aggiunta alla tabella.

3.4 Componente Norris::Lib::Utils

3.4.1 Classe ColorManager

ColorManager
`+ ColorGenerator(num:Integer)` `+ hexColorParse(color:String)`

Tabella 23: Classe ColorManager

Descrizione

Questa classe si occupa di generare dei colori di default per i grafici.

Utilizzo

Sarà utilizzata da BusinessLayer::BarChartController, BusinessLayer::LineChartController e BusinessLayer::MapChartController per generare dei colori di default.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:BarChartController Relazione entrante. La classe BarChartController utilizza le funzionalità della classe ColorGenerator;
- Norris::Lib::BusinessLayer:LineChartController Relazione entrante. La classe LineChartController utilizza le funzionalità della classe ColorGenerator;
- Norris::Lib::BusinessLayer:MapChartController Relazione entrante. La classe MapChartController utilizza le funzionalità della classe ColorGenerator.

Attributi

Assenti.

Metodi

`+ ColorGenerator(num:Integer):Array`
Questo metodo genera dei colori di default per i dati dei grafici e li ritorna in un array.

- `num:Integer`
  Intero che indica il numero di colori da generare.

`+ hexColorParse(color:String):Object`
Controllare la correttezza del colore esadecimale e ritorna un oggetto con la rappresentazione RGB del medesimo colore; altrimenti ritorna un oggetto vuoto.

- `color:String`
  Il colore esadecimale da trasformare.

3.4.2 Classe NorrisError

NorrisError
`+ NorrisError(err:Integer)` `+ toString()`

Tabella 24: Classe NorrisError

Descrizione

Questa classe si occupa di gestire i messaggi di errore.

Utilizzo

Sarà utilizzata dalle classi del *package*$_G$ BusinessLayer quando si verifica un errore.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:ActiveResourcesController Relazione entrante. La classe ActiveResourcesController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:BarChartController Relazione entrante. La classe BarChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:DataConsistency Relazione entrante. La classe DataConsistency utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:LineChartController Relazione entrante. La classe LineChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:MapChartController Relazione entrante. La classe MapChartController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe NorrisError per la gestione degli errori ;
- Norris::Lib::BusinessLayer:SocketController Relazione entrante. La classe SocketController utilizza le funzionalità della classe NorrisError per la gestione degli errori;
- Norris::Lib::BusinessLayer:TableController Relazione entrante. La classe TableController utilizza le funzionalità della classe NorrisError per la gestione degli errori.

Attributi

Assenti.

Metodi

`+ NorrisError(err:Integer):void`
La funzione crea un oggetto di tipo NorrisError.

- `err:Integer`
  Intero che contiene il codice dell'errore da sollevare.

`+ toString():String`
La funzione crea la stringa NorrisError.

3.4.3 Classe ProgressiveID

ProgressiveID
+ ProgressiveID()

Tabella 25: Classe ProgressiveID

Descrizione

Questa classe si occupa di generare un ID univoco per ogni risorsa.

Utilizzo

Sarà utilizzata dalle classi di BusinessLayer che si occupano della creazione di pagine e grafici.

Relazioni con altre classi

- Norris::Lib::BusinessLayer:BarChartController Relazione entrante. La classe BarChartController utilizza le funzionalità della classe ProgressiveID;
- Norris::Lib::BusinessLayer:LineChartController Relazione entrante. La classe LineChartController utilizza le funzionalità della classe ProgressiveID;
- Norris::Lib::BusinessLayer:MapChartController Relazione entrante. La classe MapChartController utilizza le funzionalità della classe ProgressiveID;
- Norris::Lib::BusinessLayer:PageController Relazione entrante. La classe PageController utilizza le funzionalità della classe ProgressiveID per assegnare il codice ID univoco durante la creazione di un oggetto di tipo PageModel;
- Norris::Lib::BusinessLayer:TableController Relazione entrante. La classe TableController utilizza le funzionalità della classe ProgressiveID.

Attributi

Assenti.

Metodi

`+ ProgressiveID():Integer`
Quando chiamato, questo metodo ritorna il valore corrente dell'id progressivo e poi lo incrementa.

3.4.4 Classe SocketService

SocketService
+ setSocket(socket:Object) + connectionManager() + sendUpdate(room:Integer, info:Object) + getSocketNamespace()

Tabella 26: Classe SocketService

Descrizione

Questa classe contiene i metodi riguardanti *Socket.io*$_G$ che verranno utilizzati dalla libreria Norris..

Utilizzo

Sarà utilizzata dalla classe BusinessLayer::SocketController che si occupa della gestione del *socket*$_G$..

Relazioni con altre classi

- Norris::Lib::BusinessLayer:SocketController Relazione entrante. La classe SocketController utilizza le funzionalità della classe SocketService.

Attributi

Assenti.

Metodi

`+ setSocket(socket:Object):void`
Questo metodo imposta il *namespace*$_G$ di *Socket.io*$_G$ che verrà utilizzato dalla libreria Norris.

- `socket:Object`
  Oggetto di tipo Socket.

`+ connectionManager():void`
Questo metodo avvia il gestore di connessioni tra il *server*$_G$ e il *client*$_G$.

`+ sendUpdate(room:Integer, info:Object):void`
Questo metodo invia un *JSON*$_G$ dal *server*$_G$ al *client*$_G$.

- `room:Integer`
  Integer che contiene l'id della room.

- `info:Object`
  I dati che vengono inviati al *client*$_G$ per l'aggiornamento sottoforma di *JSON*$_G$.

`+ getSocketNamespace():String`
Questo metodo ritorna il *namespace*$_G$ di *Socket.io*$_G$ correntemente utilizzato da Norris.

4 Specifica classi del *front-end*$_G$

4.1 Componente Norris::NorrisApp::Controllers

4.1.1 Classe BarLineChartCtrl

BarLineChartCtrl
+ scope:Object + window:Object + scope.graph:Object
+ BarLineChartCtrl(scope:Object, SocketsSvc:Object, BarLineSvc:Object, window:Object)

Tabella 27: Classe BarLineChartCtrl

Descrizione

Questa classe contiene i metodi necessari per gestire i grafici di tipo *Bar Chart*$_G$ e *Line Chart*$_G$ lato *front-end*$_G$.

Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end*$_G$ che riguardano i grafici di tipo *Bar Chart*$_G$ e *Line Chart*$_G$.

Relazioni con altre classi

- Norris::NorrisApp::Model:BarChartMdl Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe BarChartMdl;
- Norris::NorrisApp::Views:BarChartView Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe BarChartView;
- Norris::NorrisApp::Services:BarLineSvc Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe BarLineSvc;
- Norris::NorrisApp::Model:LineChartMdl Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe LineChartMdl;
- Norris::NorrisApp::Views:LineChartView Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe LineChartView;
- Norris::NorrisApp::Services:SocketsSvc Relazione uscente. La classe BarLineChartCtrl utilizzerà le funzionalità della classe SocketsSvc;
- Norris::NorrisApp::Controllers:FrontCtrl Relazione entrante. La classe FrontCtrl utilizzerà le funzionalità della classe BarLineChartCtrl .

Attributi

`+ scope:Object`
Questo campo dati rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato;

`+ window:Object`
Questo campo dati rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri;

`+ scope.graph:Object`
Questo campo dati rappresenta il grafico generato dalla libreria Norris, con tutte le sue proprietà. Contiene l'id del grafico, il titolo, le serie di dati, le etichette ad esse associate e le proprietà del grafico stesso.

Metodi

`+ BarLineChartCtrl(scope:Object, SocketsSvc:Object, BarLineSvc:Object, window:Object):BarLineChartCtrl`
Questo metodo costruisce la classe BarLineChartCtrl.

- `scope:Object`
  Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato.
- `SocketsSvc:Object`
  Questo parametro rappresenta un riferimento al servizio SocketsSvc.
- `BarLineSvc:Object`
  Questo parametro rappresenta un riferimento al servizio BarLineSvc.
- `window:Object`
  Questo parametro rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri .

4.1.2 Classe FrontCtrl

FrontCtrl
+ scope:Object - _data:Object - colClass:Integer
+ FrontCtrl(scope:Object, FrontSvc:Object, FirstConnectSvc:Object)

Tabella 28: Classe FrontCtrl

Descrizione

Questa classe si occuppa di gestire il *front-end*$_G$.

Utilizzo

Sarà utilizzata allo scopo di eseguire le prime chiamate necessarie al *front-end*$_G$.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:BarLineChartCtrl Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe BarLineChartCtrl ;
- Norris::NorrisApp::Services:FirstConnectSvc Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe FirstConnectSvc;
- Norris::NorrisApp::Model:FrontMdl Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe FrontMdl;
- Norris::NorrisApp::Services:FrontSvc Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe FrontSvc;
- Norris::NorrisApp::Controllers:MapChartCtrl Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe MapChartCtrl;
- Norris::NorrisApp::Controllers:TableCtrl Relazione uscente. La classe FrontCtrl utilizzerà le funzionalità della classe TableCtrl.

Attributi

`+ scope:Object`
Questo campo dati rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato;

`- _data:Object`
Questo campo dati contiene i dati di un oggetto pagina, comprensivi del titolo della pagina, del *namespace*$_G$ del *socket*$_G$ e dei dati dei grafici;

`- colClass:Integer`
Questo campo dati è calcolato per ottenere il valore da 1 a 12 che rappresenta la larghezza delle colonne secondo le specifiche di Bootstrap.

Metodi

`+ FrontCtrl(scope:Object, FrontSvc:Object, FirstConnectSvc:Object):FrontCtrl`
Questo metodo costruisce la classe FrontCtrl.

- `scope:Object`
  Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato .
- `FrontSvc:Object`
  Questo parametro rappresenta un riferimento al servizio FrontSvc.

- `FirstConnectSvc:Object`
  Questo parametro rappresenta un riferimento al servizio FirstConnectSvc.

4.1.3 Classe MapChartCtrl

MapChartCtrl
+ scope:Object + scope.graph:Object + window:Object
+ MapChartCtrl(scope:Object, SocketsSvc:Object, MapSvc:Object, window:Object)

Tabella 29: Classe MapChartCtrl

Descrizione

Questa classe contiene i metodi necessari per gestire il grafico di tipo *Map Chart*$_G$ lato *front-end*$_G$.

Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end*$_G$ che riguardano il grafico di tipo *Map Chart*$_G$.

Relazioni con altre classi

- Norris::NorrisApp::Model:MapChartMdl Relazione uscente. La classe MapChartCtrl utilizzerà le funzionalità della classe MapChartMdl;
- Norris::NorrisApp::Views:MapChartView Relazione uscente. La classe MapChartCtrl utilizzerà le funzionalità della classe MapChartView;
- Norris::NorrisApp::Services:MapSvc Relazione uscente. La classe MapChartCtrl utilizzerà le funzionalità della classe MapSvc;
- Norris::NorrisApp::Services:SocketsSvc Relazione uscente. La classe MapChartCtrl utilizzerà le funzionalità della classe SocketSvc;
- Norris::NorrisApp::Controllers:FrontCtrl Relazione entrante. La classe FrontCtrl utilizzerà le funzionalità della classe MapChartCtrl.

Attributi

`+ scope:Object`
Questo campo dati rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato ;

`+ scope.graph:Object`
Questo campo dati rappresenta il grafico generato dalla libreria Norris, con tutte le sue proprietà. Contiene l'id del grafico, il titolo, i punti e i percorsi da rappresentare nonchè le proprietà del grafico stesso ;

`+ window:Object`
Questo campo dati rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri .

Metodi

`+ MapChartCtrl(scope:Object, SocketsSvc:Object, MapSvc:Object, window:Object):MapC`
Questo metodo costruisce la classe MapChartCtrl.

- `scope:Object`
  Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato .
- `SocketsSvc:Object`
  Questo parametro rappresenta un riferimento al servizio SocketsSvc .
- `MapSvc:Object`
  Questo parametro rappresenta un riferimento al servizio MapSvc .
- `window:Object`
  Questo parametro rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri .

4.1.4 Classe TableCtrl

TableCtrl
+ scope:Object + window:Object + scope.graph:Object
+ TableCtrl(scope:Object, SocketsSvc:Object, TableSvc:Object, window:Object)

Tabella 30: Classe TableCtrl

Descrizione

Questa classe contiene i metodi necessari per gestire il grafico di tipo *Table*$_G$ lato *front-end*$_G$.

Utilizzo

Sarà utilizzata allo scopo di gestire i dati ricevuti dal *back-end*$_G$ che riguardano il grafico di tipo *Table*$_G$.

Relazioni con altre classi

- Norris::NorrisApp::Services:SocketsSvc Relazione uscente. La classe TableCtrl utilizzerà le funzionalità della classe SocketsSvc;
- Norris::NorrisApp::Model:TableMdl Relazione uscente. La classe TableCtrl utilizzerà le funzionalità della classe TableMdl;
- Norris::NorrisApp::Services:TableSvc Relazione uscente. La classe TableCtrl utilizzerà le funzionalità della classe TableSvc;
- Norris::NorrisApp::Views:TableView Relazione uscente. La classe TableCtrl utilizzerà le funzionalità della classe TableView;
- Norris::NorrisApp::Controllers:FrontCtrl Relazione entrante. La classe FrontCtrl utilizzerà le funzionalità della classe TableCtrl.

Attributi

`+ scope:Object`
Questo campo dati rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato ;

`+ window:Object`
Questo campo dati rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri ;

`+ scope.graph:Object`
Questo campo dati rappresenta il grafico generato dalla libreria Norris, con tutte le sue proprietà. Contiene l'id del grafico, il titolo, le serie di dati, le etichette ad esse associate e le proprietà del grafico stesso .

Metodi

`+ TableCtrl(scope:Object, SocketsSvc:Object, TableSvc:Object, window:Object):Table`
Questo metodo costruisce la classe TableCtrl .

- `scope:Object`
  Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato .
- `SocketsSvc:Object`
  Questo parametro rappresenta un riferimento al servizio SocketsSvc .
- `TableSvc:Object`
  Questo parametro rappresenta un riferimento al servizio TableSvc .

- `window:Object`
  Questo parametro rappresenta la finestra del browser che visualizza la pagina, e permette di effettuare operazioni sulla stessa, leggendone o modificandone dei parametri .

4.2 Componente Norris::NorrisApp::Model

4.2.1 Classe BarChartMdl

BarChartMdl
+ graphG:Object

Tabella 31: Classe BarChartMdl

Descrizione

Questa classe conterrà i valori relativi al grafico *Bar Chart*$_G$ dal lato *front-end*$_G$.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS*$_G$ prevede l'inserimento dei valori all'interno delle classi del *package*$_G$ Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end*$_G$ di Norris.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:BarLineChartCtrl Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe BarChartMdl.

Attributi

`+ graphG:Object`
Questo campo dati, istanziato in BarLineChartCtrl rappresenta il modello dei dati di un grafico di tipo *Bar Chart*$_G$.

Metodi

Assenti.

4.2.2 Classe FrontMdl

FrontMdl
+ colMdClass:String + colSmClass:String + nspSock:String

Tabella 32: Classe FrontMdl

Descrizione

Questa classe conterrà i valori relativi ai grafici di una pagina dal lato *front-end*$_G$.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS*$_G$ prevede l'inserimento dei valori all'interno delle classi del *package*$_G$ Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end*$_G$ di Norris.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:FrontCtrl Relazione entrante. La classe FrontCtrl utilizzerà le funzionalità della classe FrontMdl.

Attributi

`+ colMdClass:String`
Questo campo dati rappresenta la proprietà CSS da assegnare alle colonne per rappresentarne la larghezza in modalità schermo medio secondo le specifiche di Bootstrap;

`+ colSmClass:String`
Questo campo dati rappresenta la proprietà CSS da assegnare alle colonne per rappresentarne la larghezza in modalità *tablet*$_G$ secondo le specifiche di Bootstrap ;

`+ nspSock:String`
Rappresenta il *namespace*$_G$ a cui sarà necessario connettersi per ottenere dati dal *socket*$_G$ del *server*$_G$.

Metodi

Assenti.

4.2.3 Classe LineChartMdl

LineChartMdl
+ graphG:Object

Tabella 33: Classe LineChartMdl

Descrizione

Questa classe conterrà i valori relativi al grafico *Line Chart*$_G$ dal lato *front-end*$_G$.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS*$_G$ prevede l'inserimento dei valori all'interno delle classi del *package*$_G$ Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end*$_G$ di Norris.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:BarLineChartCtrl Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe LineChartMdl.

Attributi

`+ graphG:Object`
Questo campo dati, istanziato in BarLineChartCtrl rappresenta il modello dei dati di un grafico di tipo *Line Chart*$_G$.

Metodi

Assenti.

4.2.4 Classe MapChartMdl

MapChartMdl
+ title:String + center:Array + zoom:Number + markers:Array + polylines:Array + colors:Array + legendItems:Array

Tabella 34: Classe MapChartMdl

Descrizione

Questa classe conterrà i valori relativi al grafico *Map Chart*$_G$ dal lato *front-end*$_G$.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS*$_G$ prevede l'inserimento dei valori all'interno delle classi del *package*$_G$ Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end*$_G$ di Norris.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:MapChartCtrl Relazione entrante. La classe MapChartCtrl utilizzerà le funzionalità della classe MapChartMdl.

Attributi

+ title:String
Rappresenta il titolo del grafico di tipo *Map Chart*$_G$;

+ center:Array
Rappresenta il punto centrale del *Map Chart*$_G$;

+ zoom:Number
Rappresenta il livello di zoom iniziale del *Map Chart*$_G$;

+ markers:Array
Rappresenta i punti che verranno disegnati sul *Map Chart*$_G$;

+ polylines:Array
Rappresenta i percorsi che verranno tracciati sul *Map Chart*$_G$;

+ colors:Array
Rappresenta i colori dei percorsi rappresentati in polylines;

`+ legendItems:Array`
Rappresenta i nomi dei percorsi che verranno inseriti nella legenda, se attiva.

Metodi

Assenti.

4.2.5 Classe TableMdl

TableMdl
+ graphG:Object

Tabella 35: Classe TableMdl

Descrizione

Questa classe conterrà i valori relativi al grafico *Table*$_G$ dal lato *front-end*$_G$.

Utilizzo

Questa classe non verrà direttamente sviluppata in quanto l'utilizzo di *AngularJS*$_G$ prevede l'inserimento dei valori all'interno delle classi del *package*$_G$ Controllers. Si è ritenuto tuttavia necessario definirla a livello architetturale per meglio intendere il funzionamento del *front-end*$_G$ di Norris.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:TableCtrl Relazione entrante. La classe TableCtrl utilizzerà le funzionalità della classe TableMdl.

Attributi

`+ graphG:Object`
Questo campo dati, istanziato in TableCtrl rappresenta il modello dei dati di un grafico di tipo *Table*$_G$.

Metodi

Assenti.

4.3 Componente Norris::NorrisApp::Services

4.3.1 Classe BarLineSvc

BarLineSvc
+ BarLineSvc(ColorsSvc:Object) - fillLineData(series:Array, labels:Array, inData:Array, outData:Array) - setColors(colors:Array) - setOpts(title:String, xAxisName:String, yAxisName:String, showGrid:boolean, showLegend:boolean, legendPosition:String, seriesCount:String)

Tabella 36: Classe BarLineSvc

Descrizione

Questa classe contiene i metodi per le operazioni di supporto alla classe Controllers::BarLineChartCtrl .

Utilizzo

Sarà utilizzata dalla classe Controllers::BarLineChartCtrl per alcune operazioni di supporto.

Relazioni con altre classi

- Norris::NorrisApp::Services:ColorsSvc Relazione uscente. La classe BarLineSvc utilizza le funzionalità della classe ColorError per la gestione degli errori;
- Norris::NorrisApp::Controllers:BarLineChartCtrl Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe BarLineSvc.

Attributi

Assenti.

Metodi

`+ BarLineSvc(ColorsSvc:Object):BarLineSvc`
Costruisce la classe BarLineSvc.

- `ColorsSvc:Object`
  Rappresenta un riferimento alla classe ColorsSvc.

`- fillLineData(series:Array, labels:Array, inData:Array, outData:Array):Array`
Riempie l'array dei dati di un grafico di tipo *Line Chart*$_G$ o *Bar Chart*$_G$.

- **series:Array**
  Rappresenta l'array dei nomi delle serie di dati.
- **labels:Array**
  Rappresenta le etichette dell'asse X per i dati.
- **inData:Array**
  Dati grezzi in arrivo dal *back-end*$_G$.
- **outData:Array**
  Array dei dati in formato corretto da inserire nel modello del grafico.

`- setColors(colors:Array):Array`
Il metodo scorre l'array dei colori delle serie e li inserisce nel modello dati.

- **colors:Array**
  Colori delle varie serie di dati.

`- setOpts(title:String, xAxisName:String, yAxisName:String, showGrid:boolean, showLegend:boolean, legendPosition:String, seriesCount:String):Object`
Il metodo crea l'oggetto contenente le opzioni del grafico *Line Chart*$_G$ o *Bar Chart*$_G$.

- **title:String**
  Titolo del grafico.
- **xAxisName:String**
  Etichetta dell'asse delle X.
- **yAxisName:String**
  Etichetta dell'asse delle Y.
- **showGrid:boolean**
  Stato di attivazione della griglia.
- **showLegend:boolean**
  Stato di attivazione della legenda del grafico .
- **legendPosition:String**
  Posizione della legenda rispetto all'area del grafico.
- **seriesCount:String**
  Numero massimo di serie di valori da mantenere nel modello.

4.3.2 Classe ColorsSvc

ColorsSvc

`+ ColorsSvc()`
`- componentToHex(component:Number)`
`- rgbToHex(r:Number, g:Number, b:Number)`

Tabella 37: Classe ColorsSvc

Descrizione

Questa classe contiene i metodi necessari a convertire un colore da formato RGB a esadecimale.

Utilizzo

Servirà per convertire i colori dal formato ricevuto dal *server*$_G$ a quello conforme alle librerie utilizzate lato *front-end*$_G$.

Relazioni con altre classi

- Norris::NorrisApp::Services:BarLineSvc Relazione entrante. La classe BarLineSvc utilizza le funzionalità della classe ColorError per la gestione degli errori;
- Norris::NorrisApp::Services:MapSvc Relazione entrante. La classe MapSvc utilizza le funzionalità della classe ColorError per la gestione degli errori;
- Norris::NorrisApp::Services:TableSvc Relazione entrante. La classe TableSvc utilizza le funzionalità della classe ColorsSvc per la gestione delle funzionalità relative ai colori.

Attributi

Assenti.

Metodi

`+ ColorsSvc():ColorsSvc`
Costruisce la classe ColorsSvc.

`- componentToHex(component:Number):String`
Converte un valore numerico in esadecimale.

- **`component:Number`**
  Rappresenta un valore decimale da convertire in esadecimale.

`- rgbToHex(r:Number, g:Number, b:Number):String`
Converte un tre valori numerici in una stringa esadecimale.

- **`r:Number`**
  Rappresenta il valore per il colore rosso (0-255).
- **`g:Number`**
  Rappresenta il valore per il colore verde (0-255).
- **`b:Number`**
  Rappresenta il valore per il colore blu (0-255).

4.3.3 Classe FirstConnectSvc

FirstConnectSvc
+ FirstConnectSvc(resource:Object, location:Object)

Tabella 38: Classe FirstConnectSvc

Descrizione

Questa classe contiene il metodo necessario al recupero dell'*URL*$_G$ del *server*$_G$.

Utilizzo

Sarà utilizzata come classe di supporto alla classe Controllers::FirstConnectCtrl.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:FrontCtrl Relazione entrante. La classe FrontCtrl utilizzerà le funzionalità della classe FirstConnectSvc.

Attributi

Assenti.

Metodi

`+ FirstConnectSvc(resource:Object, location:Object):Object`
Il metodo legge l'*URL*$_G$ corrente ed effettua la richiesta al *back-end*$_G$.

- `resource:Object`
  Rappresenta una risorsa *REST*$_G$, è un oggetto *AngularJS*$_G$.
- `location:Object`
  Rappresenta una *URL*$_G$, è un oggetto fornito da *AngularJS*$_G$.

4.3.4 Classe FrontSvc

FrontSvc
+ FrontSvc() + createRows(data:Array, columns:Number)

Tabella 39: Classe FrontSvc

Descrizione

Questa classe contiene i metodi per le operazioni di supporto alla classe Controllers::FrontCtrl.

Utilizzo

Sarà utilizzata dalla classe Controllers::FrontCtrl per alcune operazioni di supporto.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:FrontCtrl Relazione entrante. La classe FrontCtrl utilizzerà le funzionalità della classe FrontSvc.

Attributi

Assenti.

Metodi

`+ FrontSvc():FrontSvc`
Il metodo costruisce la classe FrontSvc.

`+ createRows(data:Array, columns:Number):Array`
Divide i dati grezzi in righe e colonne .

- `data:Array`
  Rappresenta i dati grezzi dei grafici nella pagina.
- `columns:Number`
  Rappresenta il numero di colonne in cui verranno disposti i grafici, e pertanto il numero massimo di grafici per riga.

4.3.5 Classe MapSvc

MapSvc

`+ MapSvc(ColorsSvc:Object)`
`+ cnvLatLong(x:Array)`
`+ setPathMode(mode:String)`
`+ setColors(colors:Array)`
`+ createPolyline(pathLine:Array, colors:String, map:Object)`
`+ createMarker(point:Array, map:Object)`
`+ buildPath(path:Array, color:String, map:Object, polylines:Array, method:String)`
`+ buildLegend(map:Object, position:Array, id:String)`
`+ updateMovie(markers:Array, newData:Array, map:Object)`

Tabella 40: Classe MapSvc

Descrizione

Questa classe contiene i metodi per le operazioni di supporto alla classe Controllers::MapChartCtrl.

Utilizzo

Sarà utilizzata dalla classe Controllers::MapChartCtrl per alcune operazioni di supporto.

Relazioni con altre classi

- Norris::NorrisApp::Services:ColorsSvc Relazione uscente. La classe MapSvc utilizza le funzionalità della classe ColorError per la gestione degli errori;
- Norris::NorrisApp::Controllers:MapChartCtrl Relazione entrante. La classe MapChartCtrl utilizzerà le funzionalità della classe MapSvc.

Attributi

Assenti.

Metodi

`+ MapSvc(ColorsSvc:Object):MapSvc`
Crea la classe MapSvc.

- **`ColorsSvc:Object`**
  Rappresenta un riferimento alla classe ColorsSvc.

`+ cnvLatLong(x:Array):Object`
Converte un array di due punti in un oggetto LatLng da usare come punto in *Google Maps API*$_G$.

- **`x:Array`**
  Rappresenta l'array da convertire.

`+ setPathMode(mode:String):Object`
Converte una stringa in un oggetto per poter essere utilizzato come parametro nel calcolo dei percorsi.

- **`mode:String`**
  Rappresenta il metodo con il quale calcolare il percorso.

`+ setColors(colors:Array):Array`
Metodo di utilità che converte i colori in formato esadecimale.

- **`colors:Array`**
  Rappresenta i colori dei vari percorsi.

`+ createPolyline(pathLine:Array, colors:String, map:Object):Object`
A partire da una serie di punti il metodo crea una linea colorata sulla mappa.

- **`pathLine:Array`**
  Rappresenta i punti per i quali passa la linea.

- `colors:String`
  Il colore della linea.
- `map:Object`
  È il riferimento alla mappa sulla quale inserire la linea.

`+ createMarker(point:Array, map:Object):Object`
Il metodo crea un segnalino sulla mappa a partire dalle coordinate di un punto.

- `point:Array`
  Rappresenta le coordinate sulle quali inserire il segnalino.
- `map:Object`
  Riferimento alla mappa sulla quale inserire il punto.

`+ buildPath(path:Array, color:String, map:Object, polylines:Array, method:String):`
Il metodo, tramite una chiamata al servizio Navigazione di Google, a partire da due punti genera un percorso stradale tra i punti, e li disegna sulla mappa.

- `path:Array`
  Rappresenta i punti di inizio e fine del percorso.
- `color:String`
  Rappresenta il colore del percorso da tracciare.
- `map:Object`
  Rappresenta il riferimento alla mappa sulla quale disegnare il percorso.
- `polylines:Array`
  È l'array del modello dati nel quale inserire i valori dei percorsi.
- `method:String`
  Rappresenta il metodo di calcolo da utilizzare per i percorsi.

`+ buildLegend(map:Object, position:Array, id:String):void`
Il metodo posiziona la legenda dentro la mappa.

- `map:Object`
  Rappresenta il riferimento alla mappa.
- `position:Array`
  Il punto nel quale inserire la legenda.
- `id:String`
  È L'identificativo HTML dell'elemento contenente la legenda.

`+ updateMovie(markers:Array, newData:Array, map:Object):void`
Il metodo aggiorna i punti presenti sulla mappa con il metodo stream, ne aggiunge nel caso ne esistano di nuovi, rimuove quelli non più presenti e aggiorna quelli che hanno cambiato posizione.

- `markers:Array`
  Array dei segnalini presenti nella mappa.
- `newData:Array`
  Dati aggiornati dei punti da rappresentare.
- `map:Object`
  Rappresenta il riferimento alla mappa .

4.3.6 Classe SocketsSvc

SocketsSvc
- sockets:Array
+ SocketsSvc(scope:Object) + open(id:integer, namespace:String) + on(id:Number, eventName:String, callback:function) + emit(id:Number, eventName:String, data:Object, callback:function)

Tabella 41: Classe SocketsSvc

Descrizione

Questa classe contiene i metodi necessari alla gestione dei servizi di *Socket.io*$_G$ dal lato del *front-end*$_G$.

Utilizzo

Sarà utilizzata allo scopo di gestire le connessioni con il *socket*$_G$ lato *server*$_G$ dalle classi del *package*$_G$ Controllers.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:BarLineChartCtrl Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe SocketsSvc;
- Norris::NorrisApp::Controllers:MapChartCtrl Relazione entrante. La classe MapChartCtrl utilizzerà le funzionalità della classe SocketSvc;
- Norris::NorrisApp::Controllers:TableCtrl Relazione entrante. La classe TableCtrl utilizzerà le funzionalità della classe SocketsSvc.

Attributi

`- sockets:Array`
Rappresenta l'array dei *socket*$_G$, uno per ciascun grafico della pagina.

Metodi

`+ SocketsSvc(scope:Object):SocketsSvc`
Costruisce la classe SocketsSvc.

- `scope:Object`
  Questo parametro rappresenta l'oggetto che permette la comunicazione tra le viste ed il controller, rendendo possibile l'accesso al modello mantenendolo sincronizzato .

`+ open(id:integer, namespace:String):void`
Apre una connessione al *namespace*$_G$ specificato.

- `id:integer`
  Rappresenta il codice identificativo del grafico del quale aprire un *socket*$_G$.
- `namespace:String`
  Rappresenta il *namespace*$_G$ sul quale aprire la connessione.

`+ on(id:Number, eventName:String, callback:function):void`
Attiva una callback che ascolta un determinato evento.

- `id:Number`
  Rappresenta il codice identificativo del grafico del quale aprire un *socket*$_G$.
- `eventName:String`
  Rappresenta l'evento da mantenere osservato.
- `callback:function`
  Rappresenta la funzione da chiamare al verificarsi dell'evento.

`+ emit(id:Number, eventName:String, data:Object, callback:function):void`
Emette un evento sul *socket*$_G$ specificato.

- `id:Number`
  Rappresenta il codice identificativo del grafico del quale aprire un *socket*$_G$.
- `eventName:String`
  Rappresenta l'evento da emettere.
- `data:Object`
  Rappresenta i dati da emettere.
- `callback:function`
  Rappresenta la funzione da chiamare al una volta emesso l'evento .

4.3.7 Classe TableSvc

TableSvc

`+ TableSvc(ColorsSvc:Object)`
`+ fillData(headers:Array, inData:Array, colors:Array, border:Array)`
`+ inPlaceUpd(inData:Array, outData:Array)`
`+ streamUpd(inData:Array, outData:Array, limit:Number)`

Tabella 42: Classe TableSvc

Descrizione

Questa classe contiene i metodi per le operazioni di supporto alla classe Controllers::TableCtrl.

Utilizzo

Sarà utilizzata dalla classe Controllers::TableCtrl per alcune operazioni di supporto.

Relazioni con altre classi

- Norris::NorrisApp::Services:ColorsSvc Relazione uscente. La classe TableSvc utilizza le funzionalità della classe ColorsSvc per la gestione delle funzionalità relative ai colori;
- Norris::NorrisApp::Controllers:TableCtrl Relazione entrante. La classe TableCtrl utilizzerà le funzionalità della classe TableSvc.

Attributi

Assenti.

Metodi

`+ TableSvc(ColorsSvc:Object):TableSvc`
Il metodo costruisce la classe TableSvc.

- **`ColorsSvc:Object`**
  Questo parametro rappresenta un riferimento al servizio ColorsSvc .

`+ fillData(headers:Array, inData:Array, colors:Array, border:Array):Array`
Il metodo genera, a partire dai dati grezzi forniti dal *back-end*$_G$, i dati per il modello.

- **`headers:Array`**
  Rappresenta le intestazioni delle colonne della tabella.
- **`inData:Array`**
  Rappresenta l'array dei dati in arrivo.
- **`colors:Array`**
  Rappresenta l'array dei colori dei dati.
- **`border:Array`**
  Rappresenta lo stato dei bordi delle varie celle della tabella.

`+ inPlaceUpd(inData:Array, outData:Array):void`
Il metodo aggiorna i dati del modello con modalità in place.

- **`inData:Array`**
  Rappresenta l'array dei nuovi dati.
- **`outData:Array`**
  Rappresenta l'array dei dati aggiornati.

`+ streamUpd(inData:Array, outData:Array, limit:Number):void`
Il metodo aggiorna i dati del modello con metodo stream e controlla che non venga superato il limite massimo di valori.

- **`inData:Array`**
  Rappresenta l'array dei dati in arrivo.

- `outData:Array`
  Rappresenta l'array dei dati aggiornati.
- `limit:Number`
  Rappresenta il limite massimo di righe di dati da salvare.

4.4 Componente Norris::NorrisApp::Views

4.4.1 Classe BarChartView

BarChartView

Tabella 43: Classe BarChartView

Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Bar Chart*$_G$ nel *front-end*$_G$.

Utilizzo

Sarà utilizzata da *AngularJS*$_G$ per visualizzare un grafico di tipo *Bar Chart*$_G$ all'interno della pagina finale.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:BarLineChartCtrl Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe BarChartView.

Attributi

Assenti.

Metodi

Assenti.

4.4.2 Classe Index

Index

Tabella 44: Classe Index

Descrizione

Questa classe importa tutti gli script e i file necessari al *front-end*$_G$. .

Utilizzo

Sarà inviata ai browser in seguito alla loro prima richiesta e si occuperà di importare gli script necessari al funzionamento del *front-end*$_G$..

Attributi

Assenti.

Metodi

Assenti.

4.4.3 Classe LineChartView

LineChartView

Tabella 45: Classe LineChartView

Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Line Chart*$_G$ nel *front-end*$_G$.

Utilizzo

Sarà utilizzata da *AngularJS*$_G$ per visualizzare un grafico di tipo*Line Chart*$_G$ all'interno della pagina finale .

Relazioni con altre classi

- Norris::NorrisApp::Controllers:BarLineChartCtrl Relazione entrante. La classe BarLineChartCtrl utilizzerà le funzionalità della classe LineChartView.

Attributi

Assenti.

Metodi

Assenti.

4.4.4 Classe MapChartView

MapChartView

Tabella 46: Classe MapChartView

Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Map Chart*$_G$ dal lato *front-end*$_G$.

Utilizzo

Sarà utilizzata da *AngularJS*$_G$ visualizzare un grafico di tipo *Map Chart*$_G$ all'interno della pagina finale .

Relazioni con altre classi

- Norris::NorrisApp::Controllers:MapChartCtrl Relazione entrante. La classe MapChartCtrl utilizzerà le funzionalità della classe MapChartView.

Attributi

Assenti.

Metodi

Assenti.

4.4.5 Classe TableView

TableView

Tabella 47: Classe TableView

Descrizione

Questa classe garantirà un punto di visualizzazione per i grafici di tipo *Table*$_G$ dal lato *front-end*$_G$.

Utilizzo

Sarà utilizzata da *AngularJS*$_G$ per visualizzare un grafico di tipo *Table*$_G$ all'interno della pagina finale.

Relazioni con altre classi

- Norris::NorrisApp::Controllers:TableCtrl Relazione entrante. La classe TableCtrl utilizzerà le funzionalità della classe TableView.

Attributi

Assenti.

Metodi

Assenti.

5 Tracciamento

5.1 Tracciamento requisiti-classi

Requisiti	Classi
RAF1	Norris::Lib::PresentationLayer::Norris
RAF1.1	Norris::Lib::Utils::SocketService
RAF1.2	Norris::Lib::BusinessLayer::SocketController
RAF1.2.1	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::SocketController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::TableModel
RAF1.2.2	Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::SocketController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::TableModel
RAF1.2.3	Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::SocketController Norris::Lib::DataLayer::MapChartModel
RAF1.3	Norris::Lib::PresentationLayer::Page
RAF2	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF2.1	Norris::Lib::BusinessLayer::PageController Norris::Lib::DataLayer::PageModel

Requisiti	Classi
RAF3	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF3.1	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::PageModel
RAF3.2	Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::PageModel
RAF3.3	Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel
RAF3.4	Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF3.5	Norris::NorrisApp::Controllers::BarLineChartCtrl Norris::NorrisApp::Controllers::FrontCtrl Norris::NorrisApp::Controllers::MapChartCtrl Norris::NorrisApp::Controllers::TableCtrl Norris::NorrisApp::Model::BarChartMdl Norris::NorrisApp::Model::LineChartMdl Norris::NorrisApp::Model::MapChartMdl Norris::NorrisApp::Model::TableMdl Norris::NorrisApp::Services::BarLineSvc Norris::NorrisApp::Services::ColorsSvc Norris::NorrisApp::Services::FirstConnectSvc Norris::NorrisApp::Services::FrontSvc Norris::NorrisApp::Services::MapSvc Norris::NorrisApp::Services::SocketsSvc Norris::NorrisApp::Services::TableSvc
RAF3.5.1	Norris::NorrisApp::Views::BarChartView Norris::NorrisApp::Views::Index Norris::NorrisApp::Views::LineChartView Norris::NorrisApp::Views::MapChartView Norris::NorrisApp::Views::TableView

Requisiti	Classi
RAF4	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF4.1	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF4.1.1	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF4.1.2	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF4.1.3	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF4.1.4	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::Table
RAF4.1.5	Norris::Lib::PresentationLayer::Table
RAF4.1.6	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart

Requisiti	Classi
RAF4.1.7	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart
RAF4.1.8	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart
RAF4.1.9	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart
RAF4.1.10	Norris::Lib::PresentationLayer::BarChart
RAF4.1.11	Norris::Lib::PresentationLayer::MapChart
RAF4.1.12	Norris::Lib::PresentationLayer::MapChart
RAF4.1.13	Norris::Lib::PresentationLayer::Table
RAF4.1.14	Norris::Lib::PresentationLayer::Table
RAF4.1.15	Norris::Lib::PresentationLayer::Table
RAF4.1.16	Norris::Lib::PresentationLayer::Table
RAF4.1.17	Norris::Lib::PresentationLayer::Table
RAF4.2	Norris::Lib::BusinessLayer::BarChartController Norris::Lib::BusinessLayer::LineChartController Norris::Lib::BusinessLayer::MapChartController Norris::Lib::BusinessLayer::PageController Norris::Lib::BusinessLayer::TableController Norris::Lib::DataLayer::BarChartModel Norris::Lib::DataLayer::LineChartModel Norris::Lib::DataLayer::MapChartModel Norris::Lib::DataLayer::PageModel Norris::Lib::DataLayer::TableModel
RAF4.2.1	Norris::Lib::PresentationLayer::BarChart Norris::Lib::PresentationLayer::LineChart Norris::Lib::PresentationLayer::MapChart Norris::Lib::PresentationLayer::PageRouter Norris::Lib::PresentationLayer::Table
RAV3	Norris::Lib::Utils::SocketService

Tabella 48: Tracciamento Requisiti - Classi

5.2 Tracciamento classi-requisiti

Classi	Requisiti
Norris::Lib::BusinessLayer::ActiveResourcesController	
Norris::Lib::BusinessLayer::BarChartController	RAF1.2.1 RAF2 RAF3.1 RAF4 RAF4.1 RAF4.2
Norris::Lib::BusinessLayer::DataConsistency	
Norris::Lib::BusinessLayer::LineChartController	RAF1.2.1 RAF1.2.2 RAF2 RAF3.2 RAF4 RAF4.1 RAF4.2
Norris::Lib::BusinessLayer::MapChartController	RAF1.2.1 RAF1.2.3 RAF2 RAF3.3 RAF4 RAF4.1 RAF4.2
Norris::Lib::BusinessLayer::PageController	RAF2 RAF2.1 RAF3.1 RAF3.2 RAF3.3 RAF3.4 RAF4 RAF4.1 RAF4.2
Norris::Lib::BusinessLayer::SocketController	RAF1.2 RAF1.2.1 RAF1.2.2 RAF1.2.3

Classi	Requisiti
Norris::Lib::BusinessLayer::TableController	RAF1.2.1 RAF1.2.2 RAF2 RAF3.4 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer::ActiveResources	
Norris::Lib::DataLayer::BarChartModel	RAF1.2.1 RAF2 RAF3.1 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer::LineChartModel	RAF1.2.1 RAF1.2.2 RAF2 RAF3.2 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer::MapChartModel	RAF1.2.1 RAF1.2.3 RAF2 RAF3.3 RAF4 RAF4.1 RAF4.2
Norris::Lib::DataLayer::PageModel	RAF2 RAF2.1 RAF3.1 RAF3.2 RAF3.3 RAF3.4 RAF4 RAF4.1 RAF4.2

Classi	Requisiti
Norris::Lib::DataLayer::TableModel	RAF1.2.1 RAF1.2.2 RAF2 RAF3.4 RAF4 RAF4.1 RAF4.2
Norris::Lib::PresentationLayer::BarChart	RAF3 RAF4.1.1 RAF4.1.10 RAF4.1.2 RAF4.1.3 RAF4.1.4 RAF4.1.6 RAF4.1.7 RAF4.1.8 RAF4.1.9 RAF4.2.1
Norris::Lib::PresentationLayer::LineChart	RAF3 RAF4.1.1 RAF4.1.2 RAF4.1.3 RAF4.1.4 RAF4.1.6 RAF4.1.7 RAF4.1.8 RAF4.1.9 RAF4.2.1
Norris::Lib::PresentationLayer::MapChart	RAF3 RAF4.1.1 RAF4.1.11 RAF4.1.12 RAF4.1.2 RAF4.1.3 RAF4.1.4 RAF4.1.6 RAF4.1.7 RAF4.1.8 RAF4.2.1
Norris::Lib::PresentationLayer::Norris	RAF1
Norris::Lib::PresentationLayer::Page	RAF1.3

Classi	Requisiti
Norris::Lib::PresentationLayer::PageRouter	RAF4.2.1
Norris::Lib::PresentationLayer::Table	RAF3 RAF4.1.1 RAF4.1.13 RAF4.1.14 RAF4.1.15 RAF4.1.16 RAF4.1.17 RAF4.1.2 RAF4.1.3 RAF4.1.4 RAF4.1.5 RAF4.2.1
Norris::Lib::Utils::ColorManager	
Norris::Lib::Utils::NorrisError	
Norris::Lib::Utils::ProgressiveID	
Norris::Lib::Utils::SocketService	RAF1.1 RAV3
Norris::NorrisApp::Controllers::BarLineChartCtrl	RAF3.5
Norris::NorrisApp::Controllers::FrontCtrl	RAF3.5
Norris::NorrisApp::Controllers::MapChartCtrl	RAF3.5
Norris::NorrisApp::Controllers::TableCtrl	RAF3.5
Norris::NorrisApp::Model::BarChartMdl	RAF3.5
Norris::NorrisApp::Model::FrontMdl	
Norris::NorrisApp::Model::LineChartMdl	RAF3.5
Norris::NorrisApp::Model::MapChartMdl	RAF3.5
Norris::NorrisApp::Model::TableMdl	RAF3.5
Norris::NorrisApp::Services::BarLineSvc	RAF3.5
Norris::NorrisApp::Services::ColorsSvc	RAF3.5
Norris::NorrisApp::Services::FirstConnectSvc	RAF3.5
Norris::NorrisApp::Services::FrontSvc	RAF3.5

Classi	Requisiti
Norris::NorrisApp::Services::MapSvc	RAF3.5
Norris::NorrisApp::Services::SocketsSvc	RAF3.5
Norris::NorrisApp::Services::TableSvc	RAF3.5
Norris::NorrisApp::Views::BarChartView	RAF3.5.1
Norris::NorrisApp::Views::Index	RAF3.5.1
Norris::NorrisApp::Views::LineChartView	RAF3.5.1
Norris::NorrisApp::Views::MapChartView	RAF3.5.1
Norris::NorrisApp::Views::TableView	RAF3.5.1

Tabella 49: Tracciamento Classi - Requisiti

5.3 Tracciamento metodi-test

Metodo	Test
Norris::Lib::BusinessLayer::ActiveResourcesController::storeGraph()	TU47
Norris::Lib::BusinessLayer::ActiveResourcesController::retrieveGraph()	TU42
Norris::Lib::BusinessLayer::ActiveResourcesController::retrievePage()	TU42
Norris::Lib::BusinessLayer::ActiveResourcesController::storePage()	TU41
Norris::Lib::BusinessLayer::BarChartController::createBarChart()	TU5
Norris::Lib::BusinessLayer::BarChartController::getChartInfo()	TU1
Norris::Lib::BusinessLayer::BarChartController::updateInPlace()	TU11
Norris::Lib::BusinessLayer::DataConsistency::checkTemplate()	
Norris::Lib::BusinessLayer::DataConsistency::checkOrientation()	TU48
Norris::Lib::BusinessLayer::DataConsistency::checkLegendPosition()	TU49
Norris::Lib::BusinessLayer::DataConsistency::checkMapLegendPosition()	TU50
Norris::Lib::BusinessLayer::DataConsistency::checkMapZoom()	TU51
Norris::Lib::BusinessLayer::DataConsistency::pathMode()	TU52

Metodo	Test
Norris::Lib::BusinessLayer::DataConsistency::checkOrderBy()	TU53
Norris::Lib::BusinessLayer::DataConsistency::checkInsertPosition()	TU54
Norris::Lib::BusinessLayer::DataConsistency::checkDisplayedLines()	TU55
Norris::Lib::BusinessLayer::DataConsistency::checkColors()	TU56
Norris::Lib::BusinessLayer::DataConsistency::checkHex()	TU57
Norris::Lib::BusinessLayer::DataConsistency::checkColorArray()	TU58
Norris::Lib::BusinessLayer::DataConsistency::checkColorMatrix()	TU59
Norris::Lib::BusinessLayer::DataConsistency::checkSeries()	TU60
Norris::Lib::BusinessLayer::DataConsistency::checkValueType()	TU61
Norris::Lib::BusinessLayer::DataConsistency::checkDecimals()	TU62
Norris::Lib::BusinessLayer::DataConsistency::checkTableFormat()	TU63
Norris::Lib::BusinessLayer::DataConsistency::checkBounds()	TU64
Norris::Lib::BusinessLayer::DataConsistency::checkAllColors()	
Norris::Lib::BusinessLayer::DataConsistency::jsonConsistencyCheck()	
Norris::Lib::BusinessLayer::DataConsistency::inPlaceTableOptionsConsistency()	
Norris::Lib::BusinessLayer::DataConsistency::streamTableOptionsConsistency()	
Norris::Lib::BusinessLayer::DataConsistency::labelConsistency()	
Norris::Lib::BusinessLayer::DataConsistency::seriesConsistency()	
Norris::Lib::BusinessLayer::LineChartController::createLineChart()	TU5
Norris::Lib::BusinessLayer::LineChartController::updateStream()	TU11
Norris::Lib::BusinessLayer::LineChartController::getChartInfo()	TU1
Norris::Lib::BusinessLayer::LineChartController::updateInPlace()	TU11
Norris::Lib::BusinessLayer::MapChartController::createMapChart()	TU5
Norris::Lib::BusinessLayer::MapChartController::updateMovie()	TU11
Norris::Lib::BusinessLayer::MapChartController::getChartInfo()	TU1
Norris::Lib::BusinessLayer::MapChartController::updateInPlace()	

Metodo	Test
Norris::Lib::BusinessLayer::MapChartController::pointsConsistency()	
Norris::Lib::BusinessLayer::PageController::createPage()	TU3 TU7 TU10 TU20 TU27
Norris::Lib::BusinessLayer::PageController::addGraphToPage()	TU9 TU13 TU37
Norris::Lib::BusinessLayer::PageController::getPageInfo()	TU1
Norris::Lib::BusinessLayer::PageController::createPageOptions()	TU45
Norris::Lib::BusinessLayer::SocketController::setSocket()	TU1
Norris::Lib::BusinessLayer::SocketController::sendUpdate()	TU30
Norris::Lib::BusinessLayer::SocketController::socketNamespace()	TU1
Norris::Lib::BusinessLayer::TableController::createTable()	TU5 TU16
Norris::Lib::BusinessLayer::TableController::getChartInfo()	TU1
Norris::Lib::BusinessLayer::TableController::updateStream()	TU11 TU14
Norris::Lib::BusinessLayer::TableController::updateInPlace()	TU11
Norris::Lib::BusinessLayer::TableController::buildTempl()	
Norris::Lib::BusinessLayer::TableController::fillDefaultOpts()	
Norris::Lib::BusinessLayer::TableController::fillDevOpts()	
Norris::Lib::BusinessLayer::TableController::fillDefaultColorOpts()	
Norris::Lib::BusinessLayer::TableController::fillDevBgColorOpts()	
Norris::Lib::BusinessLayer::TableController::fillDevFontColorOpts()	
Norris::Lib::BusinessLayer::TableController::fillDevUpdateOpts()	
Norris::Lib::BusinessLayer::TableController::dataModelPush()	
Norris::Lib::BusinessLayer::TableController::fillClientColors()	

Metodo	Test
Norris::Lib::BusinessLayer::TableController::removeRow()	
Norris::Lib::BusinessLayer::TableController::getColors()	
Norris::Lib::DataLayer::ActiveResources::ActiveResources()	TU46
Norris::Lib::DataLayer::BarChartModel::BarChartModel()	
Norris::Lib::DataLayer::LineChartModel::LineChartModel()	
Norris::Lib::DataLayer::MapChartModel::MapChartModel()	
Norris::Lib::DataLayer::PageModel::PageModel()	
Norris::Lib::DataLayer::TableModel::TableModel()	
Norris::Lib::PresentationLayer::BarChart::BarChart()	
Norris::Lib::PresentationLayer::BarChart::getChartInfo()	TU1
Norris::Lib::PresentationLayer::BarChart::updateInPlace()	TU11
Norris::Lib::PresentationLayer::LineChart::LineChart()	
Norris::Lib::PresentationLayer::LineChart::getChartInfo()	TU1
Norris::Lib::PresentationLayer::LineChart::updateInPlace()	TU11
Norris::Lib::PresentationLayer::LineChart::updateStream()	TU11
Norris::Lib::PresentationLayer::MapChart::MapChart()	
Norris::Lib::PresentationLayer::MapChart::getChartInfo()	
Norris::Lib::PresentationLayer::MapChart::updateInPlace()	
Norris::Lib::PresentationLayer::MapChart::updateMovie()	
Norris::Lib::PresentationLayer::Norris::Norris()	TU43
Norris::Lib::PresentationLayer::Page::Page()	
Norris::Lib::PresentationLayer::Page::getPageInfo()	TU1
Norris::Lib::PresentationLayer::Page::addGraph()	TU9
Norris::Lib::PresentationLayer::PageRouter::PageRouter()	TU2 TU4 TU44

Metodo	Test
Norris::Lib::PresentationLayer::Table::Table()	
Norris::Lib::PresentationLayer::Table::getChartInfo()	TU1
Norris::Lib::PresentationLayer::Table::updateInPlace()	TU11
Norris::Lib::PresentationLayer::Table::updateStream()	TU11 TU15
Norris::Lib::Utils::ColorManager::ColorGenerator()	TU39
Norris::Lib::Utils::ColorManager::hexColorParse()	
Norris::Lib::Utils::NorrisError::NorrisError()	TU6 TU29
Norris::Lib::Utils::NorrisError::toString()	
Norris::Lib::Utils::ProgressiveID::ProgressiveID()	TU32
Norris::Lib::Utils::SocketService::setSocket()	TU1
Norris::Lib::Utils::SocketService::connectionManager()	TU40
Norris::Lib::Utils::SocketService::sendUpdate()	TU30
Norris::Lib::Utils::SocketService::getSocketNamespace()	TU1
Norris::NorrisApp::Controllers::BarLineChartCtrl::BarLineChartCtrl()	
Norris::NorrisApp::Controllers::FrontCtrl::FrontCtrl()	
Norris::NorrisApp::Controllers::MapChartCtrl::MapChartCtrl()	
Norris::NorrisApp::Controllers::TableCtrl::TableCtrl()	
Norris::NorrisApp::Services::BarLineSvc::BarLineSvc()	
Norris::NorrisApp::Services::BarLineSvc::fillLineData()	
Norris::NorrisApp::Services::BarLineSvc::setColors()	
Norris::NorrisApp::Services::BarLineSvc::setOpts()	
Norris::NorrisApp::Services::ColorsSvc::ColorsSvc()	
Norris::NorrisApp::Services::ColorsSvc::componentToHex()	
Norris::NorrisApp::Services::ColorsSvc::rgbToHex()	

Metodo	Test
Norris::NorrisApp::Services::FirstConnectSvc::FirstConnectSvc()	TU76
Norris::NorrisApp::Services::FrontSvc::FrontSvc()	
Norris::NorrisApp::Services::FrontSvc::createRows()	TU70
Norris::NorrisApp::Services::MapSvc::MapSvc()	
Norris::NorrisApp::Services::MapSvc::cnvLatLong()	TU71
Norris::NorrisApp::Services::MapSvc::setPathMode()	
Norris::NorrisApp::Services::MapSvc::setColors()	
Norris::NorrisApp::Services::MapSvc::createPolyline()	TU72
Norris::NorrisApp::Services::MapSvc::createMarker()	TU73
Norris::NorrisApp::Services::MapSvc::buildPath()	TU74
Norris::NorrisApp::Services::MapSvc::buildLegend()	TU75
Norris::NorrisApp::Services::MapSvc::updateMovie()	TU77
Norris::NorrisApp::Services::SocketsSvc::SocketsSvc()	
Norris::NorrisApp::Services::SocketsSvc::open()	TU68
Norris::NorrisApp::Services::SocketsSvc::on()	TU69
Norris::NorrisApp::Services::SocketsSvc::emit()	
Norris::NorrisApp::Services::TableSvc::TableSvc()	
Norris::NorrisApp::Services::TableSvc::fillData()	TU65
Norris::NorrisApp::Services::TableSvc::inPlaceUpd()	TU66
Norris::NorrisApp::Services::TableSvc::streamUpd()	TU67

Tabella 50: Tracciamento metodi-test di unità