A-VIEW API Document:

For Extending A-VIEW

Table of Contents

Client Side APIs

Collaboration

AutoPropertyName

Package com.amrita.edu.collaboration
Class public class AutoPropertyName
Inheritance AutoPropertyName → Object

Provides the possible values for the AutoPropertyName. Use the static constants NUMERIC or DATE This is a singleton class. Should not instantiated by the clients. It will throw error if the instantiation is tried.

Public Properties

	Property	Defined By
	type : String [read-only]	AutoPropertyName

Public Methods

	Method	Defined By
	AutoPropertyName(lock:ConstructorLock, type:String) Constructor.	AutoPropertyName

Public Constants

	Constant	Defined By
	DATE : AutoPropertyName [static] When this is used, the Property name is automatically generated with the current time in milli seconds for each addValue invocation on CollaborationObject	AutoPropertyName
	NUMERIC : AutoPropertyName [static] When this is used, the Property name is automatically generated starting with 1 and incremented by 1 for each addValue invocation on CollaborationObject	AutoPropertyName

Property Detail

type property
`type:String` [read-only]

Implementation
`public function get type():String`

Constructor Detail

AutoPropertyName() Constructor
`public function AutoPropertyName(lock:ConstructorLock, type:String)`

Constructor. For the internal use only. Instantiating would throw an error.

Parameters
`lock:ConstructorLock`

`type:String`

Constant Detail

DATE Constant
`public static const DATE:AutoPropertyName`

When this is used, the Property name is automatically generated with the current time in milli seconds for each addValue invocation on CollaborationObject

NUMERIC Constant
`public static const NUMERIC:AutoPropertyName`

When this is used, the Property name is automatically generated starting with 1 and incremented by 1 for each addValue invocation on CollaborationObject

---

CollaborationFactory

Package com.amrita.edu.collaboration
Class public class CollaborationFactory
Inheritance CollaborationFactory → Object

Public Methods

	Method	Defined By
	CollaborationFactory()	CollaborationFactory
	getCollaborationService(connection:MediaServerConnection):CollaborationService [static] Function to create a new collaboration service.	CollaborationFactory

Constructor Detail

CollaborationFactory() Constructor

```
public function CollaborationFactory()
```

Method Detail

getCollaborationService() method

```
public static function getCollaborationService(connection:MediaServerConnection):
CollaborationService
```

Function to create a new collaboration service.

Parameters

`connection:MediaServerConnection` — of type MediaServerConnection

Returns

`CollaborationService` — CollaborationService

CollaborationObject

Package com.amrita.edu.collaboration
Class public class CollaborationObject
Inheritance CollaborationObject → Object

Provides the connection to the remote CollaborationObject and helps to update the collaboration data such as chat messages or whiteboard drawings.

This also provides functionality to receive notifications when the collaboration data is modified by any of the connected clients collaborationService should be used to aquire this object.

See also modules.common.collaboration.CollaborationS

Public Properties

	Property	Defined By
	syncEventCount : uint [read-only] Return the count that the Collaboration Object sync hadler get excecuted.	CollaborationObject

	Method	Defined By
	CollaborationObject(lock:ConstructorLock) The constructor.	CollaborationObject
	addValue(propertyValue:Object):void Adds a new value on the remote Collaboration Object This would trigger the onSync and onChange notifications on all connected clients.	CollaborationObject
	destroy():void Function to close the collaboration object.	CollaborationObject
	flush():void Refer the lock() method's document.	CollaborationObject
	getData():Object Return the Collaboration object's data object.	CollaborationObject
	lock():void If the module wants to update many Collaboration properties at once, and wants all of these changes to be notified to other modules in a single change event, the mechanism of locking can be used.	CollaborationObject
	onSend(functionName:String, parameters:Object):void	

	Function used to invoke a remote method through the NetConnection object	CollaborationObject
	removeAllValues():void Removes all of Property values from the remote CollaborationObject This will invoke the onClear event on the connected clients.	CollaborationObject
	removeOnChange():void Unregisters the handler function for change notifications.	CollaborationObject
	removeOnChangeProperty(propertyName:String):void Unregisters the handler function for change notifications for a given property.	CollaborationObject
	removeOnClear():void Unregisters the handler function for clear notifications.	CollaborationObject
	removeOnDeleteProperty(propertyName:String):void Unregisters the handler function for delete notifications for a given property.	CollaborationObject
	removeOnSend(functionName:String):void Delete collaboration object	CollaborationObject
	removeOnSync():void Unregisters the handler function for data notifications.	CollaborationObject
	removeValue(propertyName:String):void Removes the property on the remote Collaboration object This would trigger the onDeleteProperty notifications on all connected clients.	CollaborationObject
	resetConnection(connection:NetC onnection):void	CollaborationObject
	send(functionName:String, parameters:Object):void collaboration object.send	CollaborationObject

	setOnChange(callback:Function): void Used to register for getting notifications when a new Property is added to the collaboration data or any exisiting Property is modified by any of the connected clients.	CollaborationObject
	setOnChangeProperty(propertyName:String, callback:Function):void Used to register for getting notifications when a given Property is modified by any of the connected clients.	CollaborationObject
	setOnClear(callback:Function):voi d Used to register for getting notifications when the collaboration data is cleared by any of the connected clients.	CollaborationObject
	setOnDeleteProperty(propertyName:String, callback:Function):void Used to register for getting notifications when a given Property is deleted by any of the connected clients.	CollaborationObject
	setOnSend(functionName:String, callback:Function):void Set collaboration object	CollaborationObject
	setOnSync(callback:Function):voi d Used to register for getting notifications when the collaboration data is modified by any of the connected clients.	CollaborationObject
	setValue(propertyName:String, propertyValue:Object):void Sets the value of a property on the remote Collaboration object.	CollaborationObject
	unlock():void Refer the lock() method's document.	CollaborationObject

Public Methods

Property Detail

syncEventCount property

`syncEventCount:uint` [read-only]

Return the count that the Collaboration Object sync hadler get excecuted.

Implementation

```
public function get syncEventCount():uint
```

Constructor Detail

CollaborationObject() Constructor

```
public function CollaborationObject(lock:ConstructorLock)
```

The constructor. Should not be used. Please use CollaborationService.connectCollaborationObject instead. Usage of this constructor would throw an error.

Parameters

`lock:ConstructorLock`

Method Detail

addValue() method

```
public function addValue(propertyValue:Object):void
```

Adds a new value on the remote Collaboration Object This would trigger the onSync and onChange notifications on all connected clients.

This method should be used only for the CollaborationObjects which are setup with AutoPropertyName Based on the setup, either a date based or a numeric automatic Property name is generated on the server and is attached to the Property value.

This is done by creating a wrapper object and the automatic Propertyname is added to the object as autoPropertyName attribute and the propertyValue is added as propertyValue attribute.

Parameters

`propertyValue:Object`

Example

The following code gives an example use case

```
var collabObj:CollaborationObject = collaborationService.connectCollaborationObject
("Chat",AutoPropertyName.DATE);
collabObj.setOnChange(changeHandler);

//This method is adding the new chat message to the remote CollaboraitonObject public
function sendMessageServer(message:String):void
{
collabObj.addValue(message);
}

//The change handler is getting the message out of the wrapper object by accessing the
propertyValue attribute.


public function changeHandler(value:Object):void
{
//Do something with the new or modified value chatComp.messageList.addItem
(value.propertyValue); //Adding the new chat message to the list
}
```

destroy() method

`public function destroy():void`

Function to close the collaboration object.

flush() method

`public function flush():void`

Refer the lock() method's document. This method commits the pending changes to the Collaboration object and causes other partcipants to be notified.

getData() method

`public function getData():Object`

Return the Collaboration object's data object. This function can be used to check values of different properties in the Collaboration object.

Returns
`Object`

lock() method

`public function lock():void`

If the module wants to update many Collaboration properties at once, and wants all of these changes to be notified to other modules in a single change event, the mechanism of locking can be used. This mechanism has four steps. 1. Lock: The first step is to lock the collaboration object by calling lock() method, which prevents any more updates to it 2. Update: Make the necessary changes to the Collaboration object 3.

Flush: The second step is to commit the changes, at this
point all these changes are sent to all the other subscribers 4. Unlock: Unlock the Collaboration object so that others can update it as well.

onSend() method

`public function onSend(functionName:String, parameters:Object):void`

Function used to invoke a remote method through the NetConnection object

Parameters

`functionName:String` — of type String

`parameters:Object` — of type Object

removeAllValues() method

`public function removeAllValues():void`

Removes all of Property values from the remote CollaborationObject This will invoke the onClear event on the connected clients. Ex: It would be useful when the client wants to wipe out white board or clear entire chat history

removeOnChange() method

`public function removeOnChange():void`

Unregisters the handler function for change notifications.

removeOnChangeProperty() method

`public function removeOnChangeProperty(propertyName:String):void`

Unregisters the handler function for change notifications for a given property.

Parameters

`propertyName:String` — the name of the parameter for which the notifications are removed

removeOnClear() method

`public function removeOnClear():void`

Unregisters the handler function for clear notifications.

removeOnDeleteProperty() method

`public function removeOnDeleteProperty(propertyName:String):void`

Unregisters the handler function for delete notifications for a given property.

Parameters

`propertyName:String` — the name of the parameter for which the notifications are removed

removeOnSend() method

`public function removeOnSend(functionName:String):void`

Delete collaboration object

Parameters

`functionName:String` — of type String

removeOnSync() method

`public function removeOnSync():void`

Unregisters the handler function for data notifications.

removeValue() method

`public function removeValue(propertyName:String):void`

Removes the property on the remote Collaboration object This would trigger the onDeleteProperty notifications on all connected clients.

Parameters

`propertyName:String` — name of the property to be deleted such as a whiteboard shape or specific chat message

resetConnection() method

`public function resetConnection(connection:NetConnection):void`

Parameters

`connection:NetConnection`

send() method

`public function send(functionName:String, parameters:Object):void`

collaboration object.send

Parameters

`functionName:String` — of type String

`parameters:Object` — of type Object

setOnChange() method

`public function setOnChange(callback:Function):void`

Used to register for getting notifications when a new Property is added to the collaboration data or any exisiting Property is modified by any of the connected clients. For Ex: This handler is useful to track when a new chat message is added or new white board shape is added.

Parameters

`callback:Function` — the handler function. The handler function should be a public function accepting a parameter of type Object as the new value of the property, and a parameter of type String as the name of the property The Object is the new Parameter Value or the modified Parameter value.

Example
The following code gives an example use case

```
var collabObj:CollaborationObject = collaborationService.connectCollaborationObject
("Chat",AutoPropertyName.DATE);
collabObj.setOnChange(changeHandler);

public function changeHandler(value:Object,propertyName:String):void
{
//Do something with the new or modified value


chatComp.messageList.addItem(value.propertyValue); //Adding the new chat message to the
list}
```

setOnChangeProperty() method

`public function setOnChangeProperty(propertyName:String, callback:Function):void`

Used to register for getting notifications when a given Property is modified by any of the connected clients. For Ex: In a document sharing application, this handler is useful to track when a page is changed, or rotated or the pointer is moved etc

Parameters

`propertyName:String` — the name of the parameter for which the notifications are desired `callback:Function` — the handler function. The handler function should be a public function accepting a parameter of type Object The Object is the modified Parameter value.

Example

The following code gives an example use case

```
var collabObj:CollaborationObject = collaborationService.connectCollaborationObject
("DocumentSharing"); collabObj.setOnChangeProperty(pageNumber,pageNumberHandler);
collabObj.setOnChangeProperty(pointerPosition,pointerPositionHandler);

public function pageNumberHandler(newPageNumber:Object,propertyName:String):void
{
//Do something with the modified value docComp.setNewPage(newPageNumber);
//Setting the new page
}

public function pointerPositionHandler(newPosition:Object,propertyName:String):void
{
//Do something with the modified value docComp.pointer.moveTo(newPosition); //Moving the
pointer to new position
}
```

setOnClear() method

```
public function setOnClear(callback:Function):void
```

Used to register for getting notifications when the collaboration data is cleared by any of the connected clients. Ex: This event is triggered when the chat history is completely cleared or when the whiteboard page is cleared

Parameters

`callback:Function` — the handler function. The handler function should be a public function which accepts no parameters

Example

The following code gives an example use case

```
var collabObj:CollaborationObject = collaborationService.connectCollaborationObject
("Chat",AutoPropertyName.DATE);
collabObj.setOnClear(clearHandler);


public function clearHandler():void
{
//Do some cleanup.
chatComp.messageList.removeAll(); //Removing all the historical chat messages from the
ist}
```

setOnDeleteProperty() method

`public function setOnDeleteProperty(propertyName:String, callback:Function):void`

Used to register for getting notifications when a given Property is deleted by any of the connected clients.

Parameters

`propertyName:String` — the name of the parameter for which the notifications are desired

`callback:Function` — the handler function. The handler function should be a public function accepting propertyName:String parameter

setOnSend() method

`public function setOnSend(functionName:String, callback:Function):void`

Set collaboration object

Parameters

`functionName:String` — of type String.

`callback:Function` — of type Function.

setOnSync() method

`public function setOnSync(callback:Function):void`

Used to register for getting notifications when the collaboration data is modified by any of the connected clients.

Parameters

`callback:Function` — the handler function. The handler function should be a public function accepting a parameter of type Object The Object is the actual complete collaboration object which exists on the server and gets modified by the other clients.

Example
The following code gives an example use case

```
var collabObj:CollaborationObject = collaborationService.connectCollaborationObject
("test"); collabObj.setOnSync(collaborationHandler);


public function collaborationHandler(data:Object):void
{
//Do something with data ..
this.currentValue.text = data.currentValue;
//currentValue is the Parameter name.data.currentValue gi ves the corresponding parameter
value.}
```

setValue() method

`public function setValue(propertyName:String, propertyValue:Object):void`

Sets the value of a property on the remote Collaboration object. This would trigger the onSync, onChange and onChangeProperty notifications on all connected clients. This can be used to set the values of predetermined parameters such as document page number, document rotation, pointer position etc

Parameters

`propertyName:String` — name of the property to be modified, such as page number

`propertyValue:Object` — new value of the property to be set.

unlock() method

`public function unlock():void`

Refer the lock() method's document. This method removes the lock on the Collaboration object, allowing further updates on to it.

CollaborationService

Package com.amrita.edu.collaboration
Class public class CollaborationService
Inheritance CollaborationService → Object

The CollaborationService manages the CollaborationObjects in an Application. The clients should call the connectCollaborationObject with a name. All the client instances which will be collaborating on perticular funcationality such as Chat or Whiteboard must pass the same name. Whin in a application calling the connectCollaborationObject more than once with a same name would throw an error.

Apart from providing connection to the remote CollaborationObject, this class also allows the client to disconnect with remote CollaborationObject through closeCollaborationObject method

	Method	Defined By
	CollaborationService(connection:MediaServerConnection, lock:ConstructorLock)	CollaborationService
	closeCollaborationObject(name:String):void Closes the connection to the remote CollaborationObject.	CollaborationService
	connectCollaborationObject(name:String, auto:AutoPropertyName = null):CollaborationObject Provides the connection to the remote CollaborationObject.	CollaborationService
	isConnected():Boolean Function to check whether the connection is present.	CollaborationService

Public Methods

Constructor Detail

CollaborationService() Constructor

```
public function CollaborationService(connection:MediaServerConnection, lock:
ConstructorLock)
```

Parameters

`connection:MediaServerConnection`

`lock:ConstructorLock`

Method Detail

closeCollaborationObject() method

```
public function closeCollaborationObject(name:String):void
```

Closes the connection to the remote CollaborationObject. The data in the remote object is not affected and also the connections from other clients to the CollaborationObject are not affected

Parameters

`name:String` —- The name of the CollaborationObject to which the connection has to be closed

Example

The following code gives some examples use cases

```
collaborationService.closeCollaborationObject("test");

collaborationService.closeCollaborationObject("chat");

collaborationService.closeCollaborationObject("whiteboard");
```

connectCollaborationObject() method

```
public function connectCollaborationObject(name:String, auto:AutoPropertyName = null)
:CollaborationObject
```

Provides the connection to the remote CollaborationObject. Clients must call this method first to beign the collaboration

Parameters

`name:String` — - The name of the CollaborationObject. All the clients collaborating using a perticular functionality like Chat or Whiteboard should pass the same name to see each other's changes.

The following validations are performed on the name parameter.

- name must be used only once in a given application. Calling this method more than once using the same name in an application would generate error
- name cannot contain any of the ~ % ; : , > ? ? # chars, backspace, quotes or spaces

`auto:AutoPropertyName` (default = `null`) — -

Returns

`CollaborationObject`

See also modules.common.collaboration.AutoPropertyName

Example

The following code gives some examples use cases

```
var collabObj:CollaborationObject = collaborationService.connectCollaborationObject
("test");

var collabObj:CollaborationObject = collaborationService.connectCollaborationObject
("chat",AutoPropertyName.DATE);

var collabObj:CollaborationObject = collaborationService.connectCollaborationObject
("whiteboard",AutoPropertyName.NUMERIC);
```

isConnected() method

`public function isConnected():Boolean`

Function to check whether the connection is present.

Returns
`Boolean`

Common service

MediaServerConnection

Package edu.amrita.aview.common.service
Class public class MediaServerConnection
Inheritance MediaServerConnection → flash.events.EventDispatcher

This class wraps the connection to Media server and provides additional functionalities such as connection testing, selecting appropriate port, reconnections, keep alives etc.

Public Properties

	Property	Defined By
	appName : String [read-only] return String, Application name on Media server to which the connection is established	MediaServerConnection
	connectingClient : Object [read-only]	MediaServerConnection
	connectionParams : IList [read-only]	MediaServerConnection
	connectionRetrys : int [read-only]	MediaServerConnection
	connectionURL : String [read-only]	MediaServerConnection
	fmsPort : Number [read-only]	MediaServerConnection
	instanceName : String [read-only]	MediaServerConnection

	isConnectionRejected : Boolean [read-only] Indicates if the connection is rejected by the server.	MediaServerConnection
	isDuplicateLogin : Boolean Function to check it is a duplicate login or not	MediaServerConnection
	netConnection : NetConnection [read-only]	MediaServerConnection
	netStatusCode : String [read-only]	MediaServerConnection
	serverIP : String [read-only] return String, Ip address of the Media server	MediaServerConnection

Public Methods

	Method	**Defined By**
	MediaServerConnection(serverIP:String, appName:String, instanceName:String, connectionParams:IList, client:Object) Constructor: Is used to provide all the initialization parameters.	MediaServerConnection
	addClientMethod(methodName:String, method:Function):void This method allows for adding more call back methods to the client object after the construction of MediaServerConnection The client object which is passed to the constructor should be declared as dynamic for it to be modified with this method.	MediaServerConnection
	close(isNormalTerminiation:Boolean = true):void This funciton closes the connection to Media server and performs cleanups	MediaServerConnection
	initialize():void Initialize method must be called first before calling any other methods.	MediaServerConnection
	isConnected():Boolean Function to check connection is present or not	MediaServerConnection

	resetConnectionRetrys():void	MediaServerConnection
	Function to reset the connection retry count to zero.	MediaServerConnection

Public Constants

	Constant	Defined By
	CONNECTION_RETRY_WAIT_TIME_MS : int = 3000 [static] Wait time between video reconnections	MediaServerConnection
	MAX_CONNECTION_RETRYS : int = 30 [static] Maximum number of retries performed to re establish video connection, incase the connection gets closed.	MediaServerConnection

Property Detail

appName property

`appName:String` [read-only]

return String, Application name on Media server to which the connection is established

Implementation
`public function get appName():String`

connectingClient property

`connectingClient:Object` [read-only]

Implementation
`public function get connectingClient():Object`

connectionParams property

`connectionParams:IList` [read-only]

Implementation
`public function get connectionParams():IList`

connectionRetrys property

`connectionRetrys:int` [read-only]

Implementation
```
public function get connectionRetrys():int
```

connectionURL property

`connectionURL:String` [read-only]

Implementation
```
public function get connectionURL():String
```

fmsPort property

`fmsPort:Number` [read-only]

Implementation
```
public function get fmsPort():Number
```

instanceName property

`instanceName:String` [read-only]

Implementation
```
public function get instanceName():String
```

isConnectionRejected property

`isConnectionRejected:Boolean` [read-only]

Indicates if the connection is rejected by the server. This may happen if the same user is trying to connect the server again (duplicate connection). Or any other server side error. Refer to the server side logs for more details. If this is true, then connection is not retried

Implementation
```
public function get isConnectionRejected():Boolean
```

isDuplicateLogin property

`isDuplicateLogin:Boolean`

Function to check it is a duplicate login or not

Implementation
```
public function get isDuplicateLogin():Boolean
public function set isDuplicateLogin(value:Boolean):void
```

netConnection property

`netConnection:NetConnection` [read-only]

Implementation

```
public function get netConnection():NetConnection
```

netStatusCode property

`netStatusCode:String` [read-only]

Implementation

```
public function get netStatusCode():String
```

serverIP property

`serverIP:String` [read-only]

return String, Ip address of the Media server

Implementation

```
public function get serverIP():String
```

Constructor Detail

MediaServerConnection() Constructor

```
public function MediaServerConnection(serverIP:String, appName:String, instanceName:
String, connectionParams:IList, client:Object)
```

Constructor: Is used to provide all the initialization parameters. After calling the constructor, the initialize method has to be called for actual connection creation.

Parameters

`serverIP:String` — Ip address of the Media server

`appName:String` — Application name to which the connection should be made

`instanceName:String` — Instance/Romm name to which the connection should be made. If there is no room, null should be passed.

`connectionParams:IList` — List of connection parameters in the same order as mentioned in the server

`client:Object` — Referece to the calling client object, to which server call backs are sent. This client object should implement all the expected call back methods from the server.

Method Detail

addClientMethod() method

`public function addClientMethod(methodName:String, method:Function):void`

This method allows for adding more call back methods to the client object after the construction of MediaServerConnection The client object which is passed to the constructor should be declared as dynamic for it to be modified with this method.

Parameters

`methodName:String` — Name of the call back method to be added or replaced

`method:Function` — Function object which is associated with this method call

close() method

`public function close(isNormalTerminiation:Boolean = true):void`

This funciton closes the connection to Media server and performs cleanups

Parameters

`isNormalTerminiation:Boolean` (default = `true`)

initialize() method

`public function initialize():void`

Initialize method must be called first before calling any other methods. This method first tests the connection, chooses appropriate port to connect (either 1935 or 80) and then only makes the actual connection. During the life of this connection, it would thorugh MediaServerStatusEvent event with various codes So the calling code is expected to handle this event and check for all the relevant event codes. Please refer MediaServerStatusEvent's documentation for more details.

isConnected() method

`public function isConnected():Boolean`

Function to check connection is present or not

Returns
`Boolean`

resetConnectionRetrys() method

`public function resetConnectionRetrys():void`

Function to reset the connection retry count to zero.

Constant Detail

CONNECTION_RETRY_WAIT_TIME_MS Constant

```
public static const CONNECTION_RETRY_WAIT_TIME_MS:int = 3000
```

Wait time between video reconnections

MAX_CONNECTION_RETRYS Constant

```
public static const MAX_CONNECTION_RETRYS:int = 30
```

Maximum number of retries performed to re establish video connection, incase the connection gets closed.

Common Service Events

MediaServerStatusEvent

Package edu.amrita.aview.common.service.events
Class public class MediaServerStatusEvent
Inheritance MediaServerStatusEvent → flash.events.Event

Event class used for notifying the change of status in the Media server connection

Public Properties

	Property	Defined By
	code : String [read-only]	MediaServerStatusEvent

Public Methods

	Method	Defined By
	MediaServerStatusEvent(type:String, bubbles:Boolean = false, cancelable:Boolean = false, code:String = null) Constructor.	MediaServerStatusEvent

Public Constants

Constant	Defined By
CODE_CONNECTION_TEST_FAILED : String = ConnectionTestFailed [static] This event code is used to indicate that the connection test is failed.	MediaServerStatusEvent
CODE_CONNECTION_TEST_SUCCESS : String = ConnectionTestSuccess [static] This event code is used to indicate that connection test is successful and actual connection is going to be attempted	MediaServerStatusEvent
CODE_COULD_NOT_RECONNECT : String = CouldNotReconnect [static] This event code is used to indicate that connection could not be established in the retry attempts	MediaServerStatusEvent
CODE_NET_STATUS_CHANGE : String = NetConnection.Connect.NetworkChange [static] This event code is used to indicate that connection status is in transition	MediaServerStatusEvent
CODE_NET_STATUS_CLOSED : String = NetConnection.Connect.Closed [static] Indicates that the connection is closed, the connection would be retried automatically for MediaServerConnection.MAX_CONNECTION_RETRYS,	MediaServerStatusEvent
CODE_NET_STATUS_FAILED : String = NetConnection.Connect.Failed [static] This event code is used to indicate that the connection is failed during a connection attempt	MediaServerStatusEvent
CODE_NET_STATUS_REJECTED : String = NetConnection.Connect.Rejected [static] This event code is used to indicate that the connection is rejected by the server	MediaServerStatusEvent
CODE_NET_STATUS_SUCCESS : String = NetConnection.Connect.Success [static] This event code is used to indicate that the connection is successful	MediaServerStatusEvent

	CODE_NET_STATUS_UNKNOWN : String = UNKNOWN [static] This event code is used to indicate an unhandled status code	MediaServerStatusEvent
	TYPE_CONNECTION_STATUS : String = connectionStatus [static] Type of the event, value is "connectionStatus"	MediaServerStatusEvent

Property Detail

code property

`code:String` [read-only]

Implementation

```
public function get code():String
```

Constructor Detail

MediaServerStatusEvent() Constructor

```
public function MediaServerStatusEvent(type:String, bubbles:Boolean = false, cancelable:
Boolean = false, code:String = null)
```

Constructor.

Parameters

`type:String`

`bubbles:Boolean` (default = `false`)

`cancelable:Boolean` (default = `false`)

`code:String` (default = `null`) — The connection status code.

Constant Detail

CODE_CONNECTION_TEST_FAILED Constant

```
public static const CODE_CONNECTION_TEST_FAILED:String = ConnectionTestFailed
```

This event code is used to indicate that the connection test is failed.

CODE_CONNECTION_TEST_SUCCESS Constant

```
public static const CODE_CONNECTION_TEST_SUCCESS:String = ConnectionTestSuccess
```

This event code is used to indicate that connection test is successful and actual connection is going to be attempted

CODE_COULD_NOT_RECONNECT Constant

```
public static const CODE_COULD_NOT_RECONNECT:String = CouldNotReconnect
```

This event code is used to indicate that connection could not be established in the retry attempts

CODE_NET_STATUS_CHANGE Constant

```
public static const CODE_NET_STATUS_CHANGE:String = NetConnection.Connect.NetworkChange
```

This event code is used to indicate that connection status is in transition

CODE_NET_STATUS_CLOSED Constant

```
public static const CODE_NET_STATUS_CLOSED:String = NetConnection.Connect.Closed
```

Indicates that the connection is closed, the connection would be retried automatically for MediaServerConnection.MAX_CONNECTION_RETRYS,

CODE_NET_STATUS_FAILED Constant

```
public static const CODE_NET_STATUS_FAILED:String = NetConnection.Connect.Failed
```

This event code is used to indicate that the connection is failed during a connection attempt

CODE_NET_STATUS_REJECTED Constant

```
public static const CODE_NET_STATUS_REJECTED:String = NetConnection.Connect.Rejected
```

This event code is used to indicate that the connection is rejected by the server

CODE_NET_STATUS_SUCCESS Constant

```
public static const CODE_NET_STATUS_SUCCESS:String = NetConnection.Connect.Success
```

This event code is used to indicate that the connection is successful

CODE_NET_STATUS_UNKNOWN Constant

```
public static const CODE_NET_STATUS_UNKNOWN:String = UNKNOWN
```

This event code is used to indicate an unhandled status code

TYPE_CONNECTION_STATUS Constant

```
public static const TYPE_CONNECTION_STATUS:String = connectionStatus
```

Type of the event, value is "connectionStatus"

Core Evaluation Helper

QbCategoryHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QbCategoryHelper
Inheritance QbCategoryHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

QbCategoryHelper class is used to call the remote java methods

Public Methods

	Method	Defined By
	QbCategoryHelper()	QbCategory Helper
	createQbCategory(qbCategoryVO:QbCategoryVO, userId:Number, onResult:Function, onFault:Function = null):void createQbCategory Calls the remote server to create a category in Question Bank	QbCategory Helper
	deleteQbCategory(qbCategoryId:Number, userId:Number, onResult:Function, onFault:Function = null):void deleteQbCategory Calls the remote server to delete a category in Question Bank	QbCategory Helper
	getAllActiveQbCategoriesForUser(userId:Number, onResult:Function, onFault:Function = null):void getAllActiveQbCategoriesForUser Calls the remote server to retrieve all active(not deleted) categories for user in Question Bank	QbCategory Helper
	updateQbCategory(qbCategoryVO:QbCategoryVO, userId:Number, onResult:Function, onFault:Function = null):void updateQbCategory Calls the remote server to update a category in Question Bank	QbCategory Helper

Constructor Detail

QbCategoryHelper() Constructor

```
public function QbCategoryHelper()
```

Method Detail

createQbCategory() method

```
public function createQbCategory(qbCategoryVO:QbCategoryVO, userId:Number,
onResult:Function, onFault:Function = null):void
```

createQbCategory Calls the remote server to create a category in Question Bank

Parameters

`qbCategoryVO:QbCategoryVO` — type of QbCategoryVO

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

deleteQbCategory() method

```
public function deleteQbCategory(qbCategoryId:Number, userId:Number, onResult:Function,
onFault:Function = null):void
```

deleteQbCategory Calls the remote server to delete a category in Question Bank

Parameters

`qbCategoryId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQbCategoriesForUser() method

```
public function getAllActiveQbCategoriesForUser(userId:Number, onResult:Function,
onFault:Function = null):void
```

getAllActiveQbCategoriesForUser Calls the remote server to retrieve all active(not deleted) categories for user in Question Bank

Parameters

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

updateQbCategory() method

```
public function updateQbCategory(qbCategoryVO:QbCategoryVO, userId:Number, onResult:
Function, onFault:Function = null):void
```

updateQbCategory Calls the remote server to update a category in Question Bank

Parameters

`qbCategoryVO:QbCategoryVO` — type of QbCategoryVO

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QbDifficultyLevelHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QbDifficultyLevelHelper
Inheritance QbDifficultyLevelHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

	Method	Defined By
	QbDifficultyLevelHelper()	QbDifficultyLevelHelper
	getAllActiveDifficultyLevels(onResult:Function, onFault:Function = null):void getAllActiveDifficultyLevels Calls the remote server to retreive all active difficulty levels for a question	QbDifficultyLevelHelper

Constructor Detail

QbDifficultyLevelHelper() Constructor

```
public function QbDifficultyLevelHelper()
```

Method Detail

getAllActiveDifficultyLevels() method

```
public function getAllActiveDifficultyLevels(onResult:Function, onFault:Function = null)
:void
```

getAllActiveDifficultyLevels Calls the remote server to retreive all active difficulty levels for a question

Parameters

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QbQuestionHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QbQuestionHelper
Inheritance QbQuestionHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

	Method	Defined By
	QbQuestionHelper()	QbQuestion Helper
	createQbQuestion(qbQuestionVO:QbQuestionVO, answers:ArrayCollection, userId:Number, onResult:Function, onFault:Function = null):void createQbQuestion This method creates question for a subcategory in Question Bank	QbQuestion Helper
	createQbQuestionForPolling(qbQuestionVO:QbQuestionVO, answers:ArrayCollection, userId:Number, onResult:Function, onFault:Function = null):void	

	createQbQuestionForPolling Calls the remote server to create question of polling type for a subcategory in Question Bank	QbQuestion Helper
	deleteQbQuestions(questions:ArrayCollection, userId:Number, onResult:Function, onFault:Function = null):void deleteQbQuestions Calls the remote server to delete questions for a subcategory in Question Bank	QbQuestion Helper
	getAllActiveQbQuestionsForSubcategory(subcategoryId:Number, onResult:Function, onFault:Function = null):void getAllActiveQbQuestionsForSubcategory Calls the remote server to retrieve all active questions for sub category in Question Bank	QbQuestion Helper
	getQbQuestions(categoryId:Number, subcategoryId:Number, typeId:Number, levelId:Number, quesText:String, onResult:Function, onFault:Function = null):void getQbQuestions Calls the remote server to retrieve questions for a subcategory in Question Bank	QbQuestion Helper
	getQbQuestionsForPolling(userId:Number, onResult:Function, onFault:Function = null):void getQbQuestionsForPolling Calls the remote server to retrieve questions of polling type for a subcategory in Question Bank	QbQuestion Helper
	updateQbQuestion(qbQuestionVO:QbQuestionVO, answers:ArrayCollection, userId:Number, onResult:Function, onFault:Function = null):void updateQbQuestion Calls the remote server to update a question for a subcategory in Question Bank	QbQuestion Helper

Constructor Detail

QbQuestionHelper() Constructor

```
public function QbQuestionHelper()
```

Method Detail

createQbQuestion() method

```
public function createQbQuestion(qbQuestionVO:QbQuestionVO, answers:ArrayCollection,
userId:Number, onResult:Function, onFault:Function = null):void
```

createQbQuestion This method creates question for a subcategory in Question Bank

Parameters

`qbQuestionVO:QbQuestionVO` — type of QbQuestionVO

`answers:ArrayCollection` — type of ArrayCollection

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

createQbQuestionForPolling() method

```
public function createQbQuestionForPolling(qbQuestionVO:QbQuestionVO, answers:
ArrayCollection, userId:Number, onResult:Function, onFault:Function = null):void
```

createQbQuestionForPolling Calls the remote server to create question of polling type for a subcategory in Question Bank

Parameters

`qbQuestionVO:QbQuestionVO` — type of QbQuestionVO

`answers:ArrayCollection` — type of ArrayCollection

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

deleteQbQuestions() method

```
public function deleteQbQuestions(questions:ArrayCollection, userId:Number, onResult:
Function, onFault:Function = null):void
```

deleteQbQuestions Calls the remote server to delete questions for a subcategory in Question Bank

Parameters

`questions:ArrayCollection` — type of ArrayCollection

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQbQuestionsForSubcategory() method

```
public function getAllActiveQbQuestionsForSubcategory(subcategoryId:Number, onResult:
Function, onFault:Function = null):void
```

getAllActiveQbQuestionsForSubcategory Calls the remote server to retrieve all active questions for sub category in Question Bank

Parameters

`subcategoryId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQbQuestions() method

```
public function getQbQuestions(categoryId:Number, subcategoryId:Number, typeId:Number,
levelId:Number, quesText:String, onResult:Function, onFault:Function = null):void
```

getQbQuestions Calls the remote server to retrieve questions for a subcategory in Question Bank

Parameters

`categoryId:Number` — type of Number

`subcategoryId:Number` — type of Number

`typeId:Number` — type of Number

`levelId:Number` — type of Number

`quesText:String` — type of String

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQbQuestionsForPolling() method

```
public function getQbQuestionsForPolling(userId:Number, onResult:Function, onFault:
Function = null):void
```

getQbQuestionsForPolling Calls the remote server to retrieve questions of polling type for a subcategory in Question Bank

Parameters

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

updateQbQuestion() method

```
public function updateQbQuestion(qbQuestionVO:QbQuestionVO, answers:ArrayCollection,
userId:Number, onResult:Function, onFault:Function = null):void
```

updateQbQuestion Calls the remote server to update a question for a subcategory in Question Bank

Parameters

`qbQuestionVO:QbQuestionVO` — type of QbQuestionVO

`answers:ArrayCollection` — type of ArrayCollection

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QbQuestionTypeHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QbQuestionTypeHelper
Inheritance QbQuestionTypeHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

	Method	Defined By
	QbQuestionTypeHelper()	QbQuestionTypeHelper
	getAllActiveQbQuestionTypes(onResult:Function, onFault:Function = null):void getAllActiveQbQuestionTypes Calls the remote server to get all active question types for a question	QbQuestionTypeHelper
	getQbQuestionTypeByName(questionTypeName:String, onResult:Function, onFault:Function = null):void getQbQuestionTypeByName Calls the remote server to get question type details for a given question type name	QbQuestionTypeHelper

Constructor Detail

QbQuestionTypeHelper() Constructor

```
public function QbQuestionTypeHelper()
```

Method Detail

getAllActiveQbQuestionTypes() method

```
public function getAllActiveQbQuestionTypes(onResult:Function, onFault:Function = null):
void
```

getAllActiveQbQuestionTypes Calls the remote server to get all active question types for a question

Parameters

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQbQuestionTypeByName() method

```
public function getQbQuestionTypeByName(questionTypeName:String, onResult:Function,
onFault:Function = null):void
```

getQbQuestionTypeByName Calls the remote server to get question type details for a given question type name

Parameters

`questionTypeName:String` — type of String

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QbSubcategoryHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QbSubcategoryHelper
Inheritance QbSubcategoryHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

	Method	Defined By
	QbSubcategoryHelper()	QbSubcategoryHelper
	createQbSubcategory(subcategoryVO:QbSubcategoryVO, userId:Number, onResult:Function, onFault:Function = null):void createQbSubcategory Calls the remote server to create a sub category in Question Bank	QbSubcategoryHelper
	deleteQbSubcategory(subcategoryId:Number, userId:Number, onResult:Function, onFault:Function = null):void deleteQbSubcategory Calls the remote server to delete a sub category in Question Bank	QbSubcategoryHelper
	getAllActiveQbSubcategoriesForUser(userId:Number, onResult:Function, onFault:Function = null):void getAllActiveQbSubcategoriesForUser Calls the remote server to get all active sub categories for a user in Question Bank	QbSubcategoryHelper
	getAllActiveQbSubcategoriesSummaryForCategory(categoryId:Number, onResult:Function, onFault:Function = null):void getAllActiveQbSubcategoriesSummaryForCategory Calls the remote server to get all active(not deleted) sub	QbSubcategoryHelper

	categories in Question Bank	
	updateQbSubcategory(subcategoryVO:QbSubcategoryVO, userId:Number, onResult:Function, onFault:Function = null):void updateQbSubcategory Calls the remote server to update a sub category in Question Bank	QbSubcategoryHelper

Constructor Detail

QbSubcategoryHelper() Constructor

`public function QbSubcategoryHelper()`

Method Detail

createQbSubcategory() method

```
public function createQbSubcategory(subcategoryVO:QbSubcategoryVO, userId:Number,
onResult:Function, onFault:Function = null):void
```

createQbSubcategory Calls the remote server to create a sub category in Question Bank

Parameters

`subcategoryVO:QbSubcategoryVO` — type of QbSubcategoryVO

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

deleteQbSubcategory() method

```
public function deleteQbSubcategory(subcategoryId:Number, userId:Number, onResult:
Function, onFault:Function = null):void
```

deleteQbSubcategory Calls the remote server to delete a sub category in Question Bank

Parameters

`subcategoryId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQbSubcategoriesForUser() method

```
public function getAllActiveQbSubcategoriesForUser(userId:Number, onResult:Function,
onFault:Function = null):void
```

getAllActiveQbSubcategoriesForUser Calls the remote server to get all active sub categories for a user in Question Bank

Parameters

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQbSubcategoriesSummaryForCategory() method

```
public function getAllActiveQbSubcategoriesSummaryForCategory(categoryId:Number,
onResult:Function, onFault:Function = null):void
```

getAllActiveQbSubcategoriesSummaryForCategory Calls the remote server to get all active (not deleted) sub categories in Question Bank

Parameters

`categoryId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

updateQbSubcategory() method

```
public function updateQbSubcategory(subcategoryVO:QbSubcategoryVO, userId:Number,
onResult:Function, onFault:Function = null):void
```

updateQbSubcategory Calls the remote server to update a sub category in Question Bank

Parameters

`subcategoryVO:QbSubcategoryVO` — type of QbSubcategoryVO

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QuestionPaperHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QuestionPaperHelper
Inheritance QuestionPaperHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

	Method	Defined By
	QuestionPaperHelper()	QuestionPaperHelper
	createPollingQuestionPaper(pollingQuestionId:ArrayCollection, classId:Number, courseId:Number, userId:Number, onResult:Function, onFault:Function = null):void createPollingQuestionPaper Calls the remote server to create a polling question paper	QuestionPaperHelper
	createQuestionPaper(questionPaperVO:QuestionPaperVO, creatorId:Number, onResult:Function, onFault:Function = null):void createQuestionPaper Calls the remote server to create a question paper	QuestionPaperHelper
	deleteQuestionPaper(questionPaperId:Number, userId:Number, onResult:Function, onFault:Function = null):void deleteQuestionPaper Calls the remote server to delete a question paper	QuestionPaperHelper
	genericFaultHandler(event:FaultEvent):void [override] genericFaultHandler Default Fault Handler for handling the faults which are not caught by any methods specified above	QuestionPaperHelper
	getAllActiveQuestionPapers(onResult:Function, onFault:Function = null):void getAllActiveQuestionPapers Calls the remote server to get active question papers	QuestionPaperHelper

		getAllActiveQuestionPapersForUser(userID:Number, onResult:Function, onFault:Function = null):void getAllActiveQuestionPapersForUser Calls the remote server to get active question papers for a user	QuestionPaperHelper
		getAllActiveQuestionPapersForUserInLiveClass(userID:Number, onResult:Function, isComplete:String = null, onFault:Function = null):void getAllActiveQuestionPapersForUserInLiveClass Call the remote server to get active question papers for user in live class	QuestionPaperHelper
		getFaultMessage(event:FaultEvent):String [override] getFaultMessage Gets the fault message for Default Fault Handler	QuestionPaperHelper
		getQuestionPaperComplete(questionPaperId:Number, onResult:Function, onFault:Function = null):void getQuestionPaperComplete Calls the remote server to get all question papers which are complete	QuestionPaperHelper
		getQuestionPaperIfQuestionsExist(onResult:Function, onFault:Function = null):void getQuestionPaperIfQuestionsExist Calls the remote server to check if a question exists in a question paper	QuestionPaperHelper
		questionPaperPreview(questionPaperId:Number, userId:Number, onResult:Function, onFault:Function = null):void questionPaperPreview Calls the remote server to preview question paper	QuestionPaperHelper
		saveQuestionPaper(questionPaperVO:QuestionPaperVO, userId:Number, onResult:Function, onFault:Function = null):void saveQuestionPaper Calls the remote server to save a question paper	QuestionPaperHelper
		updateQuestionPaper(questionPaperVO:QuestionPaperVO, updaterId:Number, onResult:Function, onFault:Function = null):void updateQuestionPaper Calls the remote server to update a question paper	QuestionPaperHelper
		validateQuestionPaper(questionPaperId:Number, userId:Number, onResult:Function, onFault:Function = null):void	QuestionPaperHelper

	validateQuestionPaper Calls the remote server to validate a question paper	

Constructor Detail

QuestionPaperHelper() Constructor

```
public function QuestionPaperHelper()
```

Method Detail

createPollingQuestionPaper() method

```
public function createPollingQuestionPaper(pollingQuestionId:ArrayCollection,
classId:Number, courseId:Number, userId:Number, onResult:Function, onFault:Function =
null):void
```

createPollingQuestionPaper Calls the remote server to create a polling question

paper Parameters

`pollingQuestionId:ArrayCollection` — type of ArrayCollection

`classId:Number` — type of Number

`onFault:Function` (default = `null`) — type of Function

createQuestionPaper() method

```
public function createQuestionPaper(questionPaperVO:QuestionPaperVO, creatorId:Number,
onResult:Function, onFault:Function = null):void
```

createQuestionPaper Calls the remote server to create a question paper

Parameters

`questionPaperVO:QuestionPaperVO` — type of QuestionPaperVO

`creatorId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function (default = null)` — type of Function

deleteQuestionPaper() method

```
public function deleteQuestionPaper(questionPaperId:Number, userId:Number, onResult:
Function, onFault:Function = null):void
```

deleteQuestionPaper Calls the remote server to delete a question paper

Parameters

`questionPaperId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

genericFaultHandler() method

```
override public function genericFaultHandler(event:FaultEvent):void
```

genericFaultHandler Default Fault Handler for handling the faults which are not caught by any methods specified above

Parameters

`event:FaultEvent` — of type FaultEvent

getAllActiveQuestionPapers() method

```
public function getAllActiveQuestionPapers(onResult:Function, onFault:Function = null):
void
```

getAllActiveQuestionPapers Calls the remote server to get active question papers

Parameters

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQuestionPapersForUser() method

```
public function getAllActiveQuestionPapersForUser(userID:Number, onResult:Function,
onFault:Function = null):void
```

getAllActiveQuestionPapersForUser Calls the remote server to get active question papers for a user

Parameters

`userID:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQuestionPapersForUserInLiveClass() method

```
public function getAllActiveQuestionPapersForUserInLiveClass(userID:Number, onResult:
Function, isComplete:String = null, onFault:Function = null):void
```

getAllActiveQuestionPapersForUserInLiveClass Call the remote server to get active question

papers for user in live class Parameters

`userID:Number` — type of Number

`onResult:Function` — type of Function

`isComplete:String` (default = `null`) — type of String `onFault:Function` (default = `null`) — type of

Function

getFaultMessage() method

```
override public function getFaultMessage(event:FaultEvent):String
```

getFaultMessage Gets the fault message for Default Fault Handler

Parameters

`event:FaultEvent` — of type FaultEvent

Returns
`String` — String

getQuestionPaperComplete() method

```
public function getQuestionPaperComplete(questionPaperId:Number, onResult:Function,
onFault:Function = null):void
```

getQuestionPaperComplete Calls the remote server to get all question papers which are complete

Parameters

`questionPaperId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQuestionPaperIfQuestionsExist() method

```
public function getQuestionPaperIfQuestionsExist(onResult:Function, onFault:Function =
null):void
```

getQuestionPaperIfQuestionsExist Calls the remote server to check if a question exists in a question paper

Parameters

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

questionPaperPreview() method

```
public function questionPaperPreview(questionPaperId:Number, userId:Number, onResult:
Function, onFault:Function = null):void
```

questionPaperPreview Calls the remote server to preview question paper

Parameters

`questionPaperId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

saveQuestionPaper() method

```
public function saveQuestionPaper(questionPaperVO:QuestionPaperVO, userId:Number,
onResult:Function, onFault:Function = null):void
```

saveQuestionPaper Calls the remote server to save a question

paper Parameters

`questionPaperVO:QuestionPaperVO` — type of QuestionPaperVO

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

updateQuestionPaper() method

```
public function updateQuestionPaper(questionPaperVO:QuestionPaperVO, updaterId:Number,
onResult:Function, onFault:Function = null):void
```

updateQuestionPaper Calls the remote server to update a question paper

Parameters

`questionPaperVO:QuestionPaperVO` — type of QuestionPaperVO

`updaterId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

validateQuestionPaper() method

```
public function validateQuestionPaper(questionPaperId:Number, userId:Number, onResult:
Function, onFault:Function = null):void
```

validateQuestionPaper Calls the remote server to validate a question paper

Parameters

`questionPaperId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QuestionPaperQuestionHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QuestionPaperQuestionHelper
Inheritance QuestionPaperQuestionHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

	Method	Defined By
	QuestionPaperQuestionHelper()	QuestionPaperQuestionHelper
	deleteQpqQuestions(questions:ArrayCollection, userId:Number, onResult:Function, onFault:Function = null):void	

	deleteQpqQuestions Calls the remote server to delete question paper questions	QuestionPaperQuestionHelper
	getAllActiveQuestionPaperQuestionsForQP(questionPaperId:Number, userId:Number, onResult:Function, onFault:Function = null):void getAllActiveQuestionPaperQuestionsForQP Calls the remote server to get question paper questions for a question paper	QuestionPaperQuestionHelper
	getAllActiveSpecificQuestionsForQuestionPaper(onResult:Function, onFault:Function = null):void getAllActiveSpecificQuestionsForQuestionPaper Calls the remote server to get specific questions	QuestionPaperQuestionHelper

Constructor Detail

QuestionPaperQuestionHelper() Constructor

`public function QuestionPaperQuestionHelper()`

Method Detail

deleteQpqQuestions() method

```
public function deleteQpqQuestions(questions:ArrayCollection, userId:Number, onResult:
Function, onFault:Function = null):void
```

deleteQpqQuestions Calls the remote server to delete question paper questions

Parameters

`questions:ArrayCollection` — type of ArrayCollection

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQuestionPaperQuestionsForQP() method

```
public function getAllActiveQuestionPaperQuestionsForQP(questionPaperId:Number, userId:
Number, onResult:Function, onFault:Function = null):void
```

getAllActiveQuestionPaperQuestionsForQP Calls the remote server to get question paper questions for a question paper

Parameters

`questionPaperId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveSpecificQuestionsForQuestionPaper() method

```
public function getAllActiveSpecificQuestionsForQuestionPaper(onResult:Function, onFault:
Function = null):void
```

getAllActiveSpecificQuestionsForQuestionPaper Calls the remote server to get specific questions

Parameters

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QuizHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QuizHelper
Inheritance QuizHelper ➝ edu.amrita.aview.core.shared.helper.AbstractHelper

QuizHelper class is used to call the remote java methods

Public Methods

	Method	Defined By
	QuizHelper()	QuizHelper
	createQuiz(quizVO:QuizVO, userId:Number, onResult:Function, onFault:Function = null):void createQuiz Calls the remote server to create a quiz	QuizHelper
	deleteQuiz(quizId:Number, userId:Number, onResult:Function, onFault:Function = null):void deleteQuiz Calls the remote server to delete a quiz	QuizHelper

	genericFaultHandler(event:FaultEvent):void [override] genericFaultHandler Default Fault Handler for handling the faults which are not caught by any methods specified above	QuizHelper
	getAllActiveQuizzesForQuestionPaper(questionId:Number, userId:Number, onResult:Function, onFault:Function = null):void getAllActiveQuizzesForQuestionPaper Calls the remote method to get active quizzes for question paper	QuizHelper
	getAllActiveQuizzesForStudent(userId:Number, onResult:Function, onFault:Function = null):void getAllActiveQuizzesForStudent Calls remote server to get active quizzes for a student	QuizHelper
	getAllActiveQuizzesForUser(userId:Number, onResult:Function, onFault:Function = null):void getAllActiveQuizzesForUser Calls the remote server to get active quizzes for user	QuizHelper
	getAllActiveQuizzesOffLineForStudent(userId:Number, onResult:Function, onFault:Function = null):void getAllActiveQuizzesOffLineForStudent Calls the remote server to get active quizzes(offline)	QuizHelper
	getCategoryBasedResult(quizId:Number, onResult:Function, onFault:Function = null):void getCategoryBasedResult Calls the remote server to get quiz result on basis of a category	QuizHelper
	getQuestionPaperResultForChart(quizId:Number, onResult:Function, onFault:Function = null):void getQuestionPaperResultForChart Calls the remote server to get quiz response on basis of question paper	QuizHelper
	getQuizById(quizId:Number, onResult:Function, onFault:Function = null):void getQuizById Calls the remote method to get quiz details for a quiz id	QuizHelper
	getQuizResultByLocation(quizId:Number, onResult:Function, onFault:Function = null):void getQuizResultByLocation Calls the remote server to get the quiz result on basis of location i.e local or remote	QuizHelper

	getQuizResultByQuestion(quizId:Number, onResult:Function, onFault:Function = null):void getQuizResultByQuestion Calls the remote server to get the result of students for a quiz on basis of quiz question .	QuizHelper
	getQuizResultByQuestionPaper(quizId:Number, onResult:Function, onFault:Function = null):void getQuizResultByQuestionPaper Calls the remote server to get the question paper result	QuizHelper
	getQuizResultForStudent(quizId:Number, userId:Number, onResult:Function, onFault:Function = null):void getQuizResultForStudent Calls the remote server to get quiz result for a student i.e the response of a student which consists of total score , correct answer , user id etc	QuizHelper
	sendLiveQuizAsSMS(quizId:Number, onFault:Function = null):void sendLiveQuizAsSMS Calls the remote server to send the live quiz as SMS	QuizHelper
	updateQuiz(quizVO:QuizVO, userId:Number, onResult:Function, onFault:Function = null):void updateQuiz Calls the remote server to update a quiz	QuizHelper

Constructor Detail

QuizHelper() Constructor

`public function QuizHelper()`

Method Detail

createQuiz() method

```
public function createQuiz(quizVO:QuizVO, userId:Number, onResult:Function, onFault:
Function = null):void
```

createQuiz Calls the remote server to create a quiz

Parameters

`quizVO:QuizVO` — type of QuizVO

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

deleteQuiz() method

```
public function deleteQuiz(quizId:Number, userId:Number, onResult:Function, onFault:
Function = null):void
```

deleteQuiz Calls the remote server to delete a quiz

Parameters

`quizId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

genericFaultHandler() method

```
override public function genericFaultHandler(event:FaultEvent):void
```

genericFaultHandler Default Fault Handler for handling the faults which are not caught by any methods specified above

Parameters

`event:FaultEvent` — of type FaultEvent

getAllActiveQuizzesForQuestionPaper() method

```
public function getAllActiveQuizzesForQuestionPaper(questionId:Number, userId:Number,
onResult:Function, onFault:Function = null):void
```

getAllActiveQuizzesForQuestionPaper Calls the remote method to get active quizzes for question paper

Parameters

`questionId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQuizzesForStudent() method

```
public function getAllActiveQuizzesForStudent(userId:Number, onResult:Function,
onFault:Function = null):void
```

getAllActiveQuizzesForStudent Calls remote server to get active quizzes for a student

Parameters

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQuizzesForUser() method

```
public function getAllActiveQuizzesForUser(userId:Number, onResult:Function, onFault:
Function = null):void
```

getAllActiveQuizzesForUser Calls the remote server to get active quizzes for user

Parameters

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getAllActiveQuizzesOffLineForStudent() method

```
public function getAllActiveQuizzesOffLineForStudent(userId:Number, onResult:Function,
onFault:Function = null):void
```

getAllActiveQuizzesOffLineForStudent Calls the remote server to get active quizzes(offline)

Parameters

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getCategoryBasedResult() method

```
public function getCategoryBasedResult(quizId:Number, onResult:Function, onFault:Function
= null):void
```

getCategoryBasedResult Calls the remote server to get quiz result on basis of a category

Parameters

`quizId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQuestionPaperResultForChart() method

```
public function getQuestionPaperResultForChart(quizId:Number, onResult:Function, onFault:
Function = null):void
```

getQuestionPaperResultForChart Calls the remote server to get quiz response on basis of question paper

Parameters

`quizId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQuizById() method

```
public function getQuizById(quizId:Number, onResult:Function, onFault:Function = null):
void
```

getQuizById Calls the remote method to get quiz details for a quiz id

Parameters

`quizId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQuizResultByLocation() method

```
public function getQuizResultByLocation(quizId:Number, onResult:Function, onFault:Function
= null):void
```

getQuizResultByLocation Calls the remote server to get the quiz result on basis of location i.e local or remote

Parameters

`quizId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQuizResultByQuestion() method

```
public function getQuizResultByQuestion(quizId:Number, onResult:Function, onFault:Function
= null):void
```

getQuizResultByQuestion Calls the remote server to get the result of students for a quiz on basis of quiz question .

Parameters

`quizId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQuizResultByQuestionPaper() method

```
public function getQuizResultByQuestionPaper(quizId:Number, onResult:Function, onFault:
Function = null):void
```

getQuizResultByQuestionPaper Calls the remote server to get the question paper result

Parameters

`quizId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQuizResultForStudent() method

```
public function getQuizResultForStudent(quizId:Number, userId:Number, onResult:Function,
onFault:Function = null):void
```

getQuizResultForStudent Calls the remote server to get quiz result for a student i.e the response of a student which consists of total score , correct answer , user id etc

Parameters

`quizId:Number` — type of Number

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

sendLiveQuizAsSMS() method

`public function sendLiveQuizAsSMS(quizId:Number, onFault:Function = null):void`

sendLiveQuizAsSMS Calls the remote server to send the live quiz as SMS

Parameters

`quizId:Number` — type of Number

`onFault:Function` (default = `null`) — type of Function

updateQuiz() method

```
public function updateQuiz(quizVO:QuizVO, userId:Number, onResult:Function, onFault:
Function = null):void
```

updateQuiz Calls the remote server to update a quiz

Parameters

`quizVO:QuizVO` — type of QuizVO

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QuizQuestionHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QuizQuestionHelper
Inheritance QuizQuestionHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

	Method	Defined By
	QuizQuestionHelper()	QuizQuestionHelper
	getPollingQuizForStudent(userId:Number, onResult:Function, onFault:Function = null):void getPollingQuizForStudent Calls the remote server to get polling quiz	QuizQuestionHelper

	getQuizQuestionsForQuiz(quizId:Number, onResult:Function, onFault:Function = null):void getQuizQuestionsForQuiz Calls the remote server to get quiz questions for quiz	QuizQuestionHelper

Constructor Detail

QuizQuestionHelper() Constructor

`public function QuizQuestionHelper()`

Method Detail

getPollingQuizForStudent() method

```
public function getPollingQuizForStudent(userId:Number, onResult:Function, onFault:
Function = null):void
```

getPollingQuizForStudent Calls the remote server to get polling quiz

Parameters

`userId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getQuizQuestionsForQuiz() method

```
public function getQuizQuestionsForQuiz(quizId:Number, onResult:Function, onFault:
Function = null):void
```

getQuizQuestionsForQuiz Calls the remote server to get quiz questions for quiz

Parameters

`quizId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

QuizQuestionResponseHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QuizQuestionResponseHelper
Inheritance QuizQuestionResponseHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

Method	Defined By
QuizQuestionResponseHelper()	QuizQuestionResponseHelper
getResultForPollingQuiz(quizId:Number, qbQuestionId:Number, onResult:Function, onFault:Function = null):void getResultForPollingQuiz Calls the remote server to get response of polling quiz	QuizQuestionResponseHelper
getStudentAnswerSheet(quizId:Number, userName:String, onResult:Function, onFault:Function = null):void getStudentAnswerSheet Calls the remote server to get answer sheet(quiz response) for a student	QuizQuestionResponseHelper
getStudentAnswerSheetResultHandler(event:ResultEvent):void getStudentAnswerSheetResultHandler Result Handler for getting answer sheet(quiz response) for a student	QuizQuestionResponseHelper

Constructor Detail

QuizQuestionResponseHelper() Constructor

```
public function QuizQuestionResponseHelper()
```

Method Detail

getResultForPollingQuiz() method

```
public function getResultForPollingQuiz(quizId:Number, qbQuestionId:Number, onResult:
Function, onFault:Function = null):void
```

getResultForPollingQuiz Calls the remote server to get response of polling quiz

Parameters

`quizId:Number` — type of Number

`qbQuestionId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getStudentAnswerSheet() method

```
public function getStudentAnswerSheet(quizId:Number, userName:String, onResult:Function,
onFault:Function = null):void
```

getStudentAnswerSheet Calls the remote server to get answer sheet(quiz response) for a student

Parameters

`quizId:Number` — type of Number

`userName:String` — type of String

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

getStudentAnswerSheetResultHandler() method

```
public function getStudentAnswerSheetResultHandler(event:ResultEvent):void
```

getStudentAnswerSheetResultHandler Result Handler for getting answer sheet(quiz response) for a student

Parameters

`event:ResultEvent` — of type ResultEvent

QuizResponseHelper

Package edu.amrita.aview.core.evaluation.helper
Class public class QuizResponseHelper
Inheritance QuizResponseHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Public Methods

	Method	Defined By
	QuizResponseHelper()	QuizResponse Helper
	createQuizResponse(quizResponseVo:QuizResponseVO, quizQuestionIdArr:Array, quizAnswerChoiceIdArr:Array, creatorId:Number, onResult:Function, onFault:Function = null):void createQuizResponse Calls the remote server to create a quiz response	QuizResponse Helper

Constructor Detail

QuizResponseHelper() Constructor

```
public function QuizResponseHelper()
```

Method Detail

createQuizResponse() method

```
public function createQuizResponse(quizResponseVo:QuizResponseVO, quizQuestionIdArr:Array,
quizAnswerChoiceIdArr:Array, creatorId:Number, onResult:Function, onFault:Function = null)
:void
```

createQuizResponse Calls the remote server to create a quiz

response Parameters

`quizResponseVo:QuizResponseVO` — type of QuizResponseVO

`quizQuestionIdArr:Array` — type of Array

`quizAnswerChoiceIdArr:Array` — type of Array

`creatorId:Number` — type of Number

`onResult:Function` — type of Function

`onFault:Function` (default = `null`) — type of Function

Core Shared Audit

Action

Package edu.amrita.aview.core.shared.audit
Class public class Action
Inheritance Action → Object

Loads all the ActionVOs from database. Initialized and invoked during application startup

Public Methods

Method	Defined By
getActions():void Gets all the actions from the database/java layer.	Action
getActionsResultHandlers(actions:ArrayCollection):void Receives the ActionVOs from the result handler and populates them in the AuditContext as map The rest of the application accesses these ActionVOs while auditing various actions	Action

Method Detail

getActions() method

`public function getActions():void`

Gets all the actions from the database/java layer. Usually called during the application initialization

getActionsResultHandlers() method

`public function getActionsResultHandlers(actions:ArrayCollection):void`

Receives the ActionVOs from the result handler and populates them in the AuditContext as map The rest of the application accesses these ActionVOs while auditing various actions

Parameters

`actions:ArrayCollection` — coming in from result handler of actionHelper.getActions

---

AuditConstants

Package edu.amrita.aview.core.shared.audit
Class public class AuditConstants
Inheritance AuditConstants → Object

Constants file for the Audit modules.

Public Methods

Method		Defined By
	AuditConstants()	AuditCon stants

Public Constants

Constant		Defined By
	changePassword : String = Change Password [static]	AuditConstants
	chatClear : String = ChatClear [static]	AuditConstants
	chatMessage : String = ChatMessage [static]	AuditConstants
	closeFullScreenVideoPresenter : String = CloseFullScreenVideoPresenter [static]	AuditConstants
	closeFullScreenVideoSelectedViewer : String = CloseFullScreenVideoSelectedViewer [static]	AuditConstants
	closeFullScreenVideoViewedViewer : String = CloseFullScreenVideoViewedViewer [static]	AuditConstants
	closeViewed : String = CloseViewed [static]	AuditConstants
	connectionClose : String = ConnectionClose [static]	AuditConstants
	connectionFail : String = ConnectionFail [static]	AuditConstants
	connectionReject : String = ConnectionReject [static]	AuditConstants
	connectionSuccess : String = ConnectionSuccess	

	[static]	AuditConstants
	desktopSharingEnd : String = DesktopSharingEnd [static]	AuditConstants
	desktopSharingStart : String = DesktopSharingStart [static]	AuditConstants
	documentAllowDownload : String = DocumentAllowDownload [static]	AuditConstants
	documentAnnotationTools : String = DocumentAnnotationTools [static]	AuditConstants
	documentAnnotationToolSelection : String = DocumentAnnotationToolSelection [static]	AuditConstants
	documentDenyDownload : String = DocumentDenyDownload [static]	AuditConstants
	documentDownloadLocal : String = DocumentDownloadLocal [static]	AuditConstants
	documentNavigation : String = DocumentNavigation [static]	AuditConstants
	documentPointer : String = DocumentPointer [static]	AuditConstants
	documentRefresh : String = DocumentRefresh [static]	AuditConstants
	documentRemoveAnnotationTools : String = DocumentRemoveAnnotationTools	

	[static]	AuditConstants
	documentRotate : String = DocumentRotate [static]	AuditConstants
	documentThumbnail : String = DocumentThumbnail [static]	AuditConstants
	documentUnload : String = DocumentUnload [static]	AuditConstants
	documentUpload : String = DocumentUpload [static]	AuditConstants
	documentView : String = DocumentView [static]	AuditConstants
	documentZoomIn : String = DocumentZoomIn [static]	AuditConstants
	documentZoomOut : String = DocumentZoomOut [static]	AuditConstants
	documentZoomReset : String = DocumentZoomReset [static]	AuditConstants
	editingEnd : String = EditingEnd [static]	AuditConstants
	editingStart : String = EditingStart [static]	AuditConstants
	faceRecognitionNotRegistered : String = FaceRecognitionNotRegistered [static]	AuditConstants

	faceRecognitionRegister : String = FaceRecognitionRegister [static]	AuditConstants
	faceRecognitionRemove : String = FaceRecognitionRemove [static]	AuditConstants
	feedBack : String = Feedback [static]	AuditConstants
	filterUsers : String = FilterUsers [static]	AuditConstants
	freeTalk : String = FreeTalk [static]	AuditConstants
	fullScreenVideoPresenter : String = FullScreenVideoPresenter [static]	AuditConstants
	fullScreenVideoSelectedViewer : String = FullScreenVideoSelectedViewer [static]	AuditConstants
	fullScreenVideoViewedViewer : String = FullScreenVideoViewedViewer [static]	AuditConstants
	handraised : String = Handraised [static]	AuditConstants
	handraiseReleased : String = HandraiseReleased [static]	AuditConstants
	helpUsage : String = HelpUsage [static]	AuditConstants
	hidePanel : String = HidePanel [static]	AuditConstants

	interacting : String = Interacting [static]	AuditConstants
	interactionEnded : String = InteractionEnded [static]	AuditConstants
	keyboardShortcut : String = KeyboardShortcut [static]	AuditConstants
	playbackEnd : String = PlaybackEnd [static]	AuditConstants
	playbackStart : String = PlaybackStart [static]	AuditConstants
	popIn2D : String = PopIn2D [static]	AuditConstants
	popInChat : String = PopInChat [static]	AuditConstants
	popInDocument : String = PopInDocument [static]	AuditConstants
	popInVideoSharing : String = PopInVideoSharing [static]	AuditConstants
	popInWhiteBoard : String = PopInWhiteBoard [static]	AuditConstants
	popOut2D : String = PopOut2D [static]	AuditConstants
	popOutChat : String = PopOutChat	

	[static]	AuditConstants
	popOutDocument : String = PopOutDocument [static]	AuditConstants
	popOutVideoSharing : String = PopOutVideoSharing [static]	AuditConstants
	popOutWhiteBoard : String = PopOutWhiteBoard [static]	AuditConstants
	prefMultiUserInteration : String = PrefMultiUserInteration [static]	AuditConstants
	prefUninterruptedDesktopSharing : String = PrefUninterruptedDesktopSharing [static]	AuditConstants
	prefUserListSorting : String = PrefUserListSorting [static]	AuditConstants
	presenterRequest : String = PresenterRequest [static]	AuditConstants
	presenterVideoToggle : String = PresenterVideoToggle [static]	AuditConstants
	pretestingCameraTab : String = Pre-CheckCameraTab [static]	AuditConstants
	pretestingCameraTestResult : String = Pre-CheckCameraTest [static]	AuditConstants
	pretestingLaunch : String = Pre-CheckLaunch [static]	AuditConstants

	pretestingMikeTab : String = Pre-CheckMikeTab [static]	AuditConstants
	pretestingMikeTestResult : String = Pre-CheckMikeTest [static]	AuditConstants
	pretestingSave : String = Pre-CheckSave [static]	AuditConstants
	pretestingSpeakerTab : String = Pre-CheckSpeakerTab [static]	AuditConstants
	pretestingSpeakerTestResult : String = Pre-CheckSpeakerTest [static]	AuditConstants
	privateChatMessage : String = PrivateChatMessage [static]	AuditConstants
	pushToTalk : String = PushToTalk [static]	AuditConstants
	questionAnswer : String = QuestionAnswer [static]	AuditConstants
	questionAsk : String = QuestionAsk [static]	AuditConstants
	questionDelete : String = QuestionDelete [static]	AuditConstants
	questionVote : String = QuestionVote [static]	AuditConstants
	recordingEnd : String = RecordingEnd [static]	AuditConstants

	recordingStart : String = RecordingStart [static]	AuditConstants
	releaseAllHandRaises : String = ReleaseAllHandRaises [static]	AuditConstants
	showPanel : String = ShowPanel [static]	AuditConstants
	threeDSharingLoad : String = 3DSharingLoad [static]	AuditConstants
	threeDSharingStop : String = 3DSharingStop [static]	AuditConstants
	threeDSharingUpload : String = 3DSharingUpload [static]	AuditConstants
	twoDSharingLoad : String = 2DSharingLoad [static]	AuditConstants
	twoDSharingPause : String = 2DSharingPause [static]	AuditConstants
	twoDSharingPlay : String = 2DSharingPlay [static]	AuditConstants
	twoDSharingSeek : String = 2DSharingSeek [static]	AuditConstants
	twoDSharingStop : String = 2DSharingStop [static]	AuditConstants
	twoDSharingUpLoad : String = 2DSharingUpload	

	[static]	AuditConstants
	userRole : String = UserRole [static]	AuditConstants
	videoBitrateSelection : String = VideoBitrateSelection [static]	AuditConstants
	videoPublishEnd : String = VideoPublishEnd [static]	AuditConstants
	videoPublishStart : String = VideoPublishStart [static]	AuditConstants
	videoRefresh : String = VideoRefresh [static]	AuditConstants
	videoSharingLoad : String = VideoSharingLoad [static]	AuditConstants
	videoSharingPause : String = VideoSharingPause [static]	AuditConstants
	videoSharingPlay : String = VideoSharingPlay [static]	AuditConstants
	videoSharingSeek : String = VideoSharingSeek [static]	AuditConstants
	videoSharingStop : String = VideoSharingStop [static]	AuditConstants
	videoSharingUpload : String = VideoSharingUpload [static]	AuditConstants
	videoWallClose : String =	

	VideoWallClose [static]	AuditConstants
	videoWallOpen : String = VideoWallOpen [static]	AuditConstants
	videoWallPopIn : String = VideoWallPopIn [static]	AuditConstants
	videoWallPopOut : String = VideoWallPopOut [static]	AuditConstants
	viewed : String = Viewed [static]	AuditConstants
	viewerVideoToggle : String = ViewerVideoToggle [static]	AuditConstants
	whiteBoardClear : String = WhiteBoardClear [static]	AuditConstants
	whiteBoardCollaboration : String = WhiteBoardCollaboration [static]	AuditConstants
	whiteBoardHide : String = WhiteBoardHide [static]	AuditConstants
	whiteBoardLineColor : String = WhiteBoardLineColor [static]	AuditConstants
	whiteBoardLineThickness : String = WhiteBoardLineThickness [static]	AuditConstants
	whiteBoardPageChange : String = WhiteBoardPageChange [static]	AuditConstants

	whiteBoardPointer : String = WhiteBoardPointer [static]	AuditConstants
	whiteBoardRestore : String = WhiteBoardRestore [static]	AuditConstants
	whiteBoardTool : String = WhiteBoardTool [static]	AuditConstants

Constructor Detail

AuditConstants() Constructor

```
public function AuditConstants()
```

Constant Detail

changePassword Constant

```
public static const changePassword:String = Change Password
```

chatClear Constant

```
public static const chatClear:String = ChatClear
```

chatMessage Constant

```
public static const chatMessage:String = ChatMessage
```

closeFullScreenVideoPresenter Constant

```
public static const closeFullScreenVideoPresenter:String = CloseFullScreenVideoPresenter
```

```
public static const closeFullScreenVideoSelectedViewer:String =
CloseFullScreenVideoSelectedViewer
```

closeFullScreenVideoViewedViewer Constant

```
public static const closeFullScreenVideoViewedViewer:String =
CloseFullScreenVideoViewedViewer
```

closeViewed Constant

```
public static const closeViewed:String = CloseViewed
```

connectionClose Constant

```
public static const connectionClose:String = ConnectionClose
```

connectionFail Constant
```
public static const connectionFail:String = ConnectionFail
```

connectionReject Constant
```
public static const connectionReject:String = ConnectionReject
```

connectionSuccess Constant
```
public static const connectionSuccess:String = ConnectionSuccess
```

desktopSharingEnd Constant
```
public static const desktopSharingEnd:String = DesktopSharingEnd
```

desktopSharingStart Constant
```
public static const desktopSharingStart:String = DesktopSharingStart
```

documentAllowDownload Constant
```
public static const documentAllowDownload:String = DocumentAllowDownload
```

documentAnnotationTools Constant
```
public static const documentAnnotationTools:String = DocumentAnnotationTools
```

documentAnnotationToolSelection Constant
```
public static const documentAnnotationToolSelection:String =
DocumentAnnotationToolSelection
```

documentDenyDownload Constant
```
public static const documentDenyDownload:String = DocumentDenyDownload
```

documentDownloadLocal Constant
```
public static const documentDownloadLocal:String = DocumentDownloadLocal
```

documentNavigation Constant
```
public static const documentNavigation:String = DocumentNavigation
```

documentPointer Constant
```
public static const documentPointer:String = DocumentPointer
```

documentRefresh Constant

```
public static const documentRefresh:String = DocumentRefresh
```

documentRemoveAnnotationTools Constant

```
public static const documentRemoveAnnotationTools:String = DocumentRemoveAnnotationTools
```

documentRotate Constant

```
public static const documentRotate:String = DocumentRotate
```

documentThumbnail Constant

```
public static const documentThumbnail:String = DocumentThumbnail
```

documentUnload Constant

```
public static const documentUnload:String = DocumentUnload
```

documentUpload Constant

```
public static const documentUpload:String = DocumentUpload
```

documentView Constant

```
public static const documentView:String = DocumentView
```

documentZoomIn Constant

```
public static const documentZoomIn:String = DocumentZoomIn
```

documentZoomOut Constant

```
public static const documentZoomOut:String = DocumentZoomOut
```

documentZoomReset Constant

```
public static const documentZoomReset:String = DocumentZoomReset
```

editingEnd Constant

```
public static const editingEnd:String = EditingEnd
```

editingStart Constant

```
public static const editingStart:String = EditingStart
```

faceRecognitionNotRegistered Constant

```
public static const faceRecognitionNotRegistered:String = FaceRecognitionNotRegistered
```

faceRecognitionRegister Constant

```
public static const faceRecognitionRegister:String = FaceRecognitionRegister
```

faceRecognitionRemove Constant

```
public static const faceRecognitionRemove:String = FaceRecognitionRemove
```

feedBack Constant

```
public static const feedBack:String = Feedback
```

filterUsers Constant

```
public static const filterUsers:String = FilterUsers
```

freeTalk Constant

```
public static const freeTalk:String = FreeTalk
```

fullScreenVideoPresenter Constant

```
public static const fullScreenVideoPresenter:String = FullScreenVideoPresenter
```

fullScreenVideoSelectedViewer Constant

```
public static const fullScreenVideoSelectedViewer:String = FullScreenVideoSelectedViewer
```

fullScreenVideoViewedViewer Constant

```
public static const fullScreenVideoViewedViewer:String = FullScreenVideoViewedViewer
```

handraised Constant

```
public static const handraised:String = Handraised
```

handraiseReleased Constant

```
public static const handraiseReleased:String = HandraiseReleased
```

helpUsage Constant

```
public static const helpUsage:String = HelpUsage
```

hidePanel Constant

```
public static const hidePanel:String = HidePanel
```

interacting Constant

```
public static const interacting:String = Interacting
```

interactionEnded Constant

```
public static const interactionEnded:String = InteractionEnded
```

keyboardShortcut Constant

```
public static const keyboardShortcut:String = KeyboardShortcut
```

playbackEnd Constant

```
public static const playbackEnd:String = PlaybackEnd
```

playbackStart Constant

```
public static const playbackStart:String = PlaybackStart
```

popIn2D Constant

```
public static const popIn2D:String = PopIn2D
```

popInChat Constant

```
public static const popInChat:String = PopInChat
```

popInDocument Constant

```
public static const popInDocument:String = PopInDocument
```

popInVideoSharing Constant

```
public static const popInVideoSharing:String = PopInVideoSharing
```

popInWhiteBoard Constant

```
public static const popInWhiteBoard:String = PopInWhiteBoard
```

popOut2D Constant

```
public static const popOut2D:String = PopOut2D
```

popOutChat Constant

```
public static const popOutChat:String = PopOutChat
```

popOutDocument Constant

```
public static const popOutDocument:String = PopOutDocument
```

popOutVideoSharing Constant

```
public static const popOutVideoSharing:String = PopOutVideoSharing
```

popOutWhiteBoard Constant

```
public static const popOutWhiteBoard:String = PopOutWhiteBoard
```

prefMultiUserInteration Constant

```
public static const prefMultiUserInteration:String = PrefMultiUserInteration
```

prefUninterruptedDesktopSharing Constant

```
public static const prefUninterruptedDesktopSharing:String =
PrefUninterruptedDesktopSharing
```

prefUserListSorting Constant

```
public static const prefUserListSorting:String = PrefUserListSorting
```

presenterRequest Constant

```
public static const presenterRequest:String = PresenterRequest
```

presenterVideoToggle Constant

```
public static const presenterVideoToggle:String = PresenterVideoToggle
```

pretestingCameraTab Constant

```
public static const pretestingCameraTab:String = Pre-CheckCameraTab
```

pretestingCameraTestResult Constant

```
public static const pretestingCameraTestResult:String = Pre-CheckCameraTest
```

pretestingLaunch Constant

```
public static const pretestingLaunch:String = Pre-CheckLaunch
```

pretestingMikeTab Constant

```
public static const pretestingMikeTab:String = Pre-CheckMikeTab
```

pretestingMikeTestResult Constant

```
public static const pretestingMikeTestResult:String = Pre-CheckMikeTest
```

pretestingSave Constant

```
public static const pretestingSave:String = Pre-CheckSave
```

pretestingSpeakerTab Constant

```
public static const pretestingSpeakerTab:String = Pre-CheckSpeakerTab
```

pretestingSpeakerTestResult Constant

```
public static const pretestingSpeakerTestResult:String = Pre-CheckSpeakerTest
```

privateChatMessage Constant

```
public static const privateChatMessage:String = PrivateChatMessage
```

pushToTalk Constant

```
public static const pushToTalk:String = PushToTalk
```

questionAnswer Constant

```
public static const questionAnswer:String = QuestionAnswer
```

questionAsk Constant

```
public static const questionAsk:String = QuestionAsk
```

questionDelete Constant

```
public static const questionDelete:String = QuestionDelete
```

questionVote Constant

```
public static const questionVote:String = QuestionVote
```

recordingEnd Constant

```
public static const recordingEnd:String = RecordingEnd
```

recordingStart Constant

```
public static const recordingStart:String = RecordingStart
```

releaseAllHandRaises Constant

```
public static const releaseAllHandRaises:String = ReleaseAllHandRaises
```

showPanel Constant

```
public static const showPanel:String = ShowPanel
```

threeDSharingLoad Constant

```
public static const threeDSharingLoad:String = 3DSharingLoad
```

threeDSharingStop Constant

```
public static const threeDSharingStop:String = 3DSharingStop
```

threeDSharingUpload Constant

```
public static const threeDSharingUpload:String = 3DSharingUpload
```

twoDSharingLoad Constant

```
public static const twoDSharingLoad:String = 2DSharingLoad
```

twoDSharingPause Constant

```
public static const twoDSharingPause:String = 2DSharingPause
```

twoDSharingPlay Constant

```
public static const twoDSharingPlay:String = 2DSharingPlay
```

twoDSharingSeek Constant

```
public static const twoDSharingSeek:String = 2DSharingSeek
```

twoDSharingStop Constant

```
public static const twoDSharingStop:String = 2DSharingStop
```

twoDSharingUpLoad Constant

```
public static const twoDSharingUpLoad:String = 2DSharingUpload
```

userRole Constant

```
public static const userRole:String = UserRole
```

videoBitrateSelection Constant

```
public static const videoBitrateSelection:String = VideoBitrateSelection
```

videoPublishEnd Constant

```
public static const videoPublishEnd:String = VideoPublishEnd
```

videoPublishStart Constant

```
public static const videoPublishStart:String = VideoPublishStart
```

videoRefresh Constant

```
public static const videoRefresh:String = VideoRefresh
```

videoSharingLoad Constant

```
public static const videoSharingLoad:String = VideoSharingLoad
```

videoSharingPause Constant

```
public static const videoSharingPause:String = VideoSharingPause
```

videoSharingPlay Constant

```
public static const videoSharingPlay:String = VideoSharingPlay
```

videoSharingSeek Constant

```
public static const videoSharingSeek:String = VideoSharingSeek
```

videoSharingStop Constant

```
public static const videoSharingStop:String = VideoSharingStop
```

videoSharingUpload Constant

```
public static const videoSharingUpload:String = VideoSharingUpload
```

videoWallClose Constant

```
public static const videoWallClose:String = VideoWallClose
```

videoWallOpen Constant

```
public static const videoWallOpen:String = VideoWallOpen
```

videoWallPopIn Constant
```
public static const videoWallPopIn:String = VideoWallPopIn
```

videoWallPopOut Constant
```
public static const videoWallPopOut:String = VideoWallPopOut
```

viewed Constant
```
public static const viewed:String = Viewed
```

viewerVideoToggle Constant
```
public static const viewerVideoToggle:String = ViewerVideoToggle
```

whiteBoardClear Constant
```
public static const whiteBoardClear:String = WhiteBoardClear
```

whiteBoardCollaboration Constant
```
public static const whiteBoardCollaboration:String = WhiteBoardCollaboration
```

whiteBoardHide Constant
```
public static const whiteBoardHide:String = WhiteBoardHide
```

whiteBoardLineColor Constant
```
public static const whiteBoardLineColor:String = WhiteBoardLineColor
```

whiteBoardLineThickness Constant
```
public static const whiteBoardLineThickness:String = WhiteBoardLineThickness
```

whiteBoardPageChange Constant
```
public static const whiteBoardPageChange:String = WhiteBoardPageChange
```

whiteBoardPointer Constant
```
public static const whiteBoardPointer:String = WhiteBoardPointer
```

whiteBoardRestore Constant
```
public static const whiteBoardRestore:String = WhiteBoardRestore
```

whiteBoardTool Constant

```
public static const whiteBoardTool:String = WhiteBoardTool
```

AuditContext

Package edu.amrita.aview.core.shared.audit
Class public class AuditContext
Inheritance AuditContext → Object

This class holds all the state information regarding Auditing. This has all the classes which communicate the audit information to server This class prepares the auditing service during the initialization process for the modules to use This class also holds relvant auditing context information such as UserLoginVO, AuditLectureVO and AuditUserSettingVO so that actions logged in the right context. This will be accessed by entire AVC code

Public Properties

	Property	Defined By
	action : Action = null [static] Helps in populating the ActionVOs	AuditContext
	actionsHash : Object = null [static] Action name->ActionVO map.	AuditContext
	auditLectureVO : AuditLectureVO = null [static] Holds the Lecture entry/exit auditing information.	AuditContext
	lecture : AuditLecture = null [static] Helps in auditing the lecture entry/exit	AuditContext
	login : AuditUserLogin = null [static] Helps in auditing the user Login	AuditContext
	userAction : AuditUserAction = null [static] Helps in auditing the actions of all the modules	AuditContext
	userLoginVO : UserLoginVO = null [static] Holds the user Login audit information.	AuditContext

Public Methods

	Method	Defined By
	init():void [static] Initializes all the Auditing helper classes, which interface with database helper classes for storing the auditing information in database Also fetches the ActionVOs from database and populates them in actionsHash	AuditContext
	populateActionIds(actionsAC:ArrayCollection):void [static] Populates the actionsHash from the list of ActionVOs.	AuditContext
	resetLectureAudit():void [static] Invoked during the lecture exit.	AuditContext

Property Detail

action property

```
public static var action:Action = null
```

Helps in populating the ActionVOs

actionsHash property

```
public static var actionsHash:Object = null
```

Action name->ActionVO map. Used by various modules to get hold of the ActionID for a given Action

auditLectureVO property

```
public static var auditLectureVO:AuditLectureVO = null
```

Holds the Lecture entry/exit auditing information. Will be used as a reference to further auditing. Populated during the lecture entry and updated during lecture exit.

lecture property

```
public static var lecture:AuditLecture = null
```

Helps in auditing the lecture entry/exit

login property

```
public static var login:AuditUserLogin = null
```

Helps in auditing the user Login

userAction property

```
public static var userAction:AuditUserAction = null
```

Helps in auditing the actions of all the modules

userLoginVO property

```
public static var userLoginVO:UserLoginVO = null
```

Holds the user Login audit information. Will be used as a reference to further auditing. Populated during the login process.

Method Detail

init() method

```
public static function init():void
```

Initializes all the Auditing helper classes, which interface with database helper classes for storing the auditing information in database Also fetches the ActionVOs from database and populates them in actionsHash

populateActionIds() method

```
public static function populateActionIds(actionsAC:ArrayCollection):void
```

Populates the actionsHash from the list of ActionVOs. So that the actions can be accessed by actionName easily for rest of the modules

Parameters

`actionsAC:ArrayCollection` — coming in from result handler of actionHelper.getActions

resetLectureAudit() method

```
public static function resetLectureAudit():void
```

Invoked during the lecture exit. It clears out the lecture auditing information. Lecture auditing information is re populated during the next lecture entry.

AuditLecture

Package edu.amrita.aview.core.shared.audit
Class public class AuditLecture
Inheritance AuditLecture ➝ Object

Helps in populating the AuditLectureVO object with the lecture audit information and interfaces with database helper for storing in database.

Public Methods

	Method	Defined By
	auditLectureEntry():void Creates the AuditLectureVO objects based on the lecture details and calls the auditLectureHelper for storing in the database	AuditLecture
	auditLectureExit():void Calls the AuditLectureHelper for updating the earlier auditLectureVO, with the modified date as the Lecture Exit time.	AuditLecture
	createAuditLectureResultHandler(auditLectureVO:AuditLectureVO):void Result handler of the createAuditLecture.	AuditLecture
	updateAuditLectureByIdResultHandler(auditLectureVO:AuditLectureVO):void Result handler of the updateAuditLectureById.	AuditLecture

Method Detail

auditLectureEntry() method

```
public function auditLectureEntry():void
```

Creates the AuditLectureVO objects based on the lecture details and calls the auditLectureHelper for storing in the database

auditLectureExit() method

```
public function auditLectureExit():void
```

Calls the AuditLectureHelper for updating the earlier auditLectureVO, with the modified date as the Lecture Exit time. This method is called during lecture exit. The modified date is populated by the server upon receiving this call.

createAuditLectureResultHandler() method

`public function createAuditLectureResultHandler(auditLectureVO:AuditLectureVO):void`

Result handler of the createAuditLecture.

Parameters

`auditLectureVO:AuditLectureVO` — - The created AuditLectureVO, returned from the server, now has the database Id also populated

updateAuditLectureByIdResultHandler() method

`public function updateAuditLectureByIdResultHandler(auditLectureVO:AuditLectureVO):void`

Result handler of the updateAuditLectureById.

Parameters

`auditLectureVO:AuditLectureVO` — - The updated AuditLectureVO returned from the server

AuditUserAction

Package edu.amrita.aview.core.shared.audit
Class public class AuditUserAction
Inheritance AuditUserAction → Object

Helps the application modules in auditing actions. Each action has a corresponding in this class, which translates the specific action details to generic action details and calls the UserActionHelper class which communicates and stores the action in the database.

Public Methods

	Method	Defined By
	changePrefUninterruptedDesktopSharingEventLog(status:String):void Audits the "PrefUninterruptedDesktopSharing" action, when the presenter/moderator changes the Uninterrupted DesktopSharing preference.	AuditUserAction
	connectionCloseEventLog(module:String, portFMS:String):void Audits the "ConnectionClose" action for a module	AuditUserAction
	connectionFailEventLog(module:String, portFMS:String):void	

	Audits the "ConnectionFail" action for a module	AuditUserAction
	connectionRejectEventLog(module:String, portFMS:String):void Audits the "ConnectionReject" action for a module	AuditUserAction
	connectionSuccessEventLog(module:String, portFMS:String, connectionRetrys:String):void Audits the "ConnectionSuccess" action for a module	AuditUserAction
	createAction(actionName:String, attributeValue1:String, attributeValue2:String, attributeValue3:String):void Pakcages the action into the UserActionVO and calls the userActionHelper for storing in the databse	AuditUserAction
	desktopSharingEndEventLog(sharingMode:String, applicationName:String):void Audits the "DesktopSharingEnd" action, when the presenter stops sharing his/her desktop	AuditUserAction
	desktopSharingStartEventLog(sharingMode:String, applicationName:String):void Audits the "DesktopSharingStart" action, when the presenter starts sharing his/her desktop	AuditUserAction
	insertPendingActionsWhileExitingClassroom():void The user actions are not sent to db directly.	AuditUserAction
	keyBoardShortcutEventLog(shortCut:String, module:String):void Audits the "KeyboardShortcut" action, when the user uses any short cut in any module	AuditUserAction
	pretestingLaunchEventLog(launchFrom:String):void Audits the "Pre-CheckLaunch" action, when the user launches the pretesting launch window	AuditUserAction
	userInteractionEndedEventLog(userName:String, interactionCount:int):void Audits the "InteractionEnded" action, when the user/presenter ends the interaction	AuditUserAction
	videoWallPopInEventLog():void Audits the "VideoWallPopIn" action, when the user Pops in/closes the video wall tab	AuditUserAction

	videoWallPopOutEventLog():void Audits the "VideoWallPopOut" action, when the user Pops out the video wall tab	AuditUserAction

Method Detail

changePrefUninterruptedDesktopSharingEventLog() method

`public function changePrefUninterruptedDesktopSharingEventLog(status:String):void`

Audits the "PrefUninterruptedDesktopSharing" action, when the presenter/moderator changes the

Uninterrupted DesktopSharing preference.

Parameters

`status:String` — - UninterruptedDesktopSharing Off/On

connectionCloseEventLog() method

`public function connectionCloseEventLog(module:String, portFMS:String):void`

Audits the "ConnectionClose" action for a module

Parameters

`module:String` — - A-VIEW Module name

`portFMS:String` — - FMS port to which the connection being made

connectionFailEventLog() method

`public function connectionFailEventLog(module:String, portFMS:String):void`

Audits the "ConnectionFail" action for a module

Parameters

`module:String` — - A-VIEW Module name

`portFMS:String` — - FMS port to which the connection being made

connectionRejectEventLog() method

`public function connectionRejectEventLog(module:String, portFMS:String):void`

Audits the "ConnectionReject" action for a module

Parameters

`module:String` — - A-VIEW Module name

`portFMS:String` — - FMS port to which the connection being made

connectionSuccessEventLog() method

```
public function connectionSuccessEventLog(module:String, portFMS:String, connectionRetrys:
String):void
```

Audits the "ConnectionSuccess" action for a module

Parameters

`module:String` — - A-VIEW Module name

`portFMS:String` — - FMS port to which the connection being made

`connectionRetrys:String` — - Current number of connection retrys

createAction() method

```
public function createAction(actionName:String, attributeValue1:String, attributeValue2:
String, attributeValue3:String):void
```

Pakcages the action into the UserActionVO and calls the userActionHelper for storing in the databse

Parameters

`actionName:String` — - Name of the Action being audited

`attributeValue1:String` — - Associated details related to the action

`attributeValue2:String` — - Associated details related to the action

`attributeValue3:String` — - Associated details related to the action

desktopSharingEndEventLog() method

```
public function desktopSharingEndEventLog(sharingMode:String, applicationName:String)
:void
```

Audits the "DesktopSharingEnd" action, when the presenter stops sharing his/her desktop

Parameters

`sharingMode:String` — - Whether the mode is Application sharing or Desktop sharing

`applicationName:String` — - Total number of interactions including the current one

desktopSharingStartEventLog() method

```
public function desktopSharingStartEventLog(sharingMode:String, applicationName:String):
void
```

Audits the "DesktopSharingStart" action, when the presenter starts sharing his/her desktop

Parameters

`sharingMode:String` — - Whether the mode is Application sharing or Desktop sharing

`applicationName:String` — - Total number of interactions including the current one

insertPendingActionsWhileExitingClassroom() method

```
public function insertPendingActionsWhileExitingClassroom():void
```

The user actions are not sent to db directly. They are batched and sent to db server in the intervals of 5 mins. So when the user is exiting the classroom, if there are any pending actions, then this method send them to db server

keyBoardShortcutEventLog() method

```
public function keyBoardShortcutEventLog(shortCut:String, module:String):void
```

Audits the "KeyboardShortcut" action, when the user uses any short cut in any module

Parameters

`shortCut:String` — used

`module:String` — in which it's used

pretestingLaunchEventLog() method

```
public function pretestingLaunchEventLog(launchFrom:String):void
```

Audits the "Pre-CheckLaunch" action, when the user launches the pretesting launch window

Parameters

`launchFrom:String` — - 'Settings' or 'Prompt'

userInteractionEndedEventLog() method

`public function userInteractionEndedEventLog(userName:String, interactionCount:int):void`

Audits the "InteractionEnded" action, when the user/presenter ends the interaction Parameters

`userName:String` — of the viewer who's interaction just ended

`interactionCount:int` — - Total number of interactions including the current one

videoWallPopInEventLog() method

`public function videoWallPopInEventLog():void`

Audits the "VideoWallPopIn" action, when the user Pops in/closes the video wall tab

videoWallPopOutEventLog() method

`public function videoWallPopOutEventLog():void`

Audits the "VideoWallPopOut" action, when the user Pops out the video wall tab

---

AuditUserLogin

Package edu.amrita.aview.core.shared.audit
Class public class AuditUserLogin
Inheritance AuditUserLogin → Object

This class helps in auditing the user login details during login and also update the same audit details during logout.

Public Properties

	Property	Defined By
	updateUserLoginCalled : Boolean = false Flag to prevent update being called multiple times from various logout/close functions	AuditUser Login

Public Methods

	Method	Defined By
	auditLogin(version:String):void Populates the UserLoginVO with various details and passes it the helper class for storing in database	AuditUser Login

	createUserLoginResultHandler(userLoginVO:UserLoginVO):void Result handler for the database storage call	AuditUser Login
	updateAuditUserLogin(userLoginVO:UserLoginVO):void Update the UserLoginVO on the server.	AuditUser Login
	updateUserLoginResultHandler(userLoginVO:UserLoginVO):void Result handler for the database update call	AuditUser Login

Property Detail

updateUserLoginCalled property

`public var updateUserLoginCalled:Boolean = false`

Flag to prevent update being called multiple times from various logout/close functions

Method Detail

auditLogin() method

`public function auditLogin(version:String):void`

Populates the UserLoginVO with various details and passes it the helper class for storing in database

Parameters

`version:String` — - Version of the A-VIEW client software

createUserLoginResultHandler() method

`public function createUserLoginResultHandler(userLoginVO:UserLoginVO):void`

Result handler for the database storage call

Parameters

`userLoginVO:UserLoginVO` — - Returned UserLoginVO from the server. Has the database id populated

updateAuditUserLogin() method

`public function updateAuditUserLogin(userLoginVO:UserLoginVO):void`

Update the UserLoginVO on the server. Called during logout/closing the application to update the logout time. Parameters

`userLoginVO:UserLoginVO` — - UserLoginVO which was returned from the server after the login

updateUserLoginResultHandler() method

```
public function updateUserLoginResultHandler(userLoginVO:UserLoginVO):void
```

Result handler for the database update call

Parameters

	Method	Defined By
	addToConevrssionQue(folderPath:String, fileName:String, isAnimated:String, onResult:Function, onFault:Function):void For converting files in remote location	ContentService
	checkFileExistance(filePath:String, onResult:Function, onFault:Function):void Checks to see if the remote file is already existing or not on the Content server.	ContentService
	copyFiles(sourcePath:String, destinationPath:String, onResult:Function, onFault:Function):void For process of copying remote folder/files to other locations.	ContentService
	createFolder(folderPath:String, onResult:Function, onFault:Function):void For creating folder(s) in the remote server	ContentService
	deleteFile(filePath:String, onResult:Function, onFault:Function):void For deleting the folder/files recursively under the specified path	ContentService
	download(filePath:String, onResult:Function, onFault:Function):void For downloading a file from remote location to Application or Personal Computer	ContentService
	getFileList(rootFolder:String, onResult:Function, onFault:Function):void For process of getting the list of files/folders from Remote location in xml format	ContentService

	getSharedFileList(url:String, libraryXML:XML, onResult:Function, onFault:Function):void For process of getting the file lists from shared library	ContentService
	textureUpload(folderPath:String, onResult:Function, onFault:Function):void	ContentService
	updateCollada(filePath:String, content:Array, onResult:Function, onFault:Function):void	ContentService
	uploadFile(folderPath:String, fileReference:FileReference, onResult:Function, onFault:Function):void For uploading file to remote location	ContentService

`userLoginVO:UserLoginVO` — - Returned UserLoginVO from the server after updating the logout time

---

Core Shared Service Content

ContentService

Package edu.amrita.aview.core.shared.service.content
Class public class ContentService
Inheritance ContentService → mx.rpc.http.mxml.HTTPService

This class provides easy to use methods to handle all content managment requests to the remote Contentserver

Public Methods

Method Detail

addToConevrssionQue() method

```
public function addToConevrssionQue(folderPath:String, fileName:String, isAnimated:String,
onResult:Function, onFault:Function):void
```

For converting files in remote location

Parameters

`folderPath:String` — Remote folder path to which the file is

uploaded `fileName:String` — containing the name of the file

`isAnimated:String` — result handler function should accept ResultEvent `onResult:Function` — fault handler function should accept FaultEvent parameter

`onFault:Function`

checkFileExistance() method

```
public function checkFileExistance(filePath:String, onResult:Function, onFault:Function):
void
```

Checks to see if the remote file is already existing or not on the Content server. If the file exists, it will send the message in ResultEvent as "Error: File already exists" and if not "Success: File does not exist"

Parameters

`filePath:String`

`onResult:Function` — result handler function should accept ResultEvent parameter

`onFault:Function` — fault handler function should accept FaultEvent parameter

copyFiles() method

```
public function copyFiles(sourcePath:String, destinationPath:String, onResult:Function,
onFault:Function):void
```

For process of copying remote folder/files to other locations. This method recursively copies all the folders/files to the new destination path

Parameters

`sourcePath:String` — from which the files/folder would copied

`destinationPath:String` — to which the files/folder would copied. It would create any directories which are needed to be created. It does not delete files/folders from destination

`onResult:Function` — result handler function should accept ResultEvent parameter

`onFault:Function` — fault handler function should accept FaultEvent parameter

createFolder() method

```
public function createFolder(folderPath:String, onResult:Function, onFault:Function):void
```

For creating folder(s) in the remote server

Parameters

`folderPath:String` — The complete path including the folder to be created. If more than folder is new in the path, this function would create all of them

`onResult:Function` — Result handler function should accept ResultEvent parameter

`onFault:Function` — Fault handler function should accept FaultEvent parameter

deleteFile() method

```
public function deleteFile(filePath:String, onResult:Function, onFault:Function):void
```

For deleting the folder/files recursively under the specified path

Parameters

`filePath:String` — Remote folder/file path which will be deleted recursively.

`onResult:Function` — result handler function should accept ResultEvent

`onFault:Function` — fault handler function should accept FaultEvent parameter

download() method

```
public function download(filePath:String, onResult:Function, onFault:Function):void
```

For downloading a file from remote location to Application or Personal Computer

Parameters

`filePath:String` — the full path of the file starting from the doc root of the Content server `onResult:Function` — result handler function should accept ResultEvent `onFault:Function` — fault handler function should accept FaultEvent

getFileList() method

```
public function getFileList(rootFolder:String, onResult:Function, onFault:Function):void
```

For process of getting the list of files/folders from Remote location in xml format

Parameters

`rootFolder:String` — The remote root directory which is used to get the file list `onResult:Function` — result handler function should accept ResultEvent `onFault:Function` — fault handler function should accept FaultEvent

getSharedFileList() method

```
public function getSharedFileList(url:String, libraryXML:XML, onResult:Function, onFault
:Function):void
```

For process of getting the file lists from shared library

Parameters

`url:String` — of String `libraryXML:XML` — of

XML

`onResult:Function` — result handler function should accept ResultEvent `onFault:Function` — fault

handler function should accept FaultEvent parameter

textureUpload() method

```
public function textureUpload(folderPath:String, onResult:Function, onFault:Function):void
```

Parameters

`folderPath:String` — of type String

`onResult:Function` — result handler function should accept ResultEvent parameter

`onFault:Function` — fault handler function should accept FaultEvent parameter

updateCollada() method

```
public function updateCollada(filePath:String, content:Array, onResult:Function, onFault
:Function):void
```

Parameters

`filePath:String` — of type String

`content:Array` — of type Content

`onResult:Function` — result handler function should accept ResultEvent `onFault:Function` — fault

handler function should accept FaultEvent parameter

uploadFile() method

```
public function uploadFile(folderPath:String, fileReference:FileReference, onResult:
Function, onFault:Function):void
```

For uploading file to remote location

Parameters

`folderPath:String` — Remote folder path to which the file is uploaded

`fileReference:FileReference` — containing the File to be uploaded `onResult:Function`

— result handler function should accept ResultEvent `onFault:Function` — fault handler

function should accept FaultEvent parameter

Shared audit helper

ActionHelper

Package edu.amrita.aview.core.shared.audit.helper
Class public class ActionHelper
Inheritance ActionHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Helper class for getting all the Actions from database. Called at the time of login to pre fetch all the Actions

Public Methods

	Method	Defined By
	ActionHelper() Constructor.	ActionHelper
	getActions(onResult:Function, onFault:Function = null):void Fetches all the actions from Database from the action table.	ActionHelper

Constructor Detail

ActionHelper() Constructor
`public function ActionHelper()`

Constructor. Initializes the remote objects and registers the result and fault handlers

Method Detail

getActions() method
`public function getActions(onResult:Function, onFault:Function = null):void`

Fetches all the actions from Database from the action table.

Parameters

`onResult:Function` — The reference to the calling component/class. Used by this class to call back in the result/fault handlers. The reference is assigned to AsyncToken and then accessed by the result/fault handers to call back on this object.

`onFault:Function` (default = `null`)

AuditLectureHelper

Package edu.amrita.aview.core.shared.audit.helper
Class public class AuditLectureHelper
Inheritance AuditLectureHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Helper class for recording user's entry into an A-VIEW session. Stores/Retrieves the contents of the AuditLectureVO to the database

Public Methods

Method	Defined By
AuditLectureHelper() constructor Initializes the remote objects and registers the result and fault handlers	AuditLectureHelper
createAuditLecture(auditLectureVO:AuditLectureVO, onResult:Function, onFault:Function = null):void Creates the Lecture entry audit record in audit_lecture table.	AuditLectureHelper
updateAuditLectureById(auditLectureId:Number, onResult:Function, onFault:Function = null):void Updates the Lecture entry audit record, which was created during session entry, when the user exits a session.	AuditLectureHelper

Constructor Detail

AuditLectureHelper() Constructor
`public function AuditLectureHelper()`

constructor Initializes the remote objects and registers the result and fault handlers

Method Detail

createAuditLecture() method

```
public function createAuditLecture(auditLectureVO:AuditLectureVO, onResult:Function,
onFault:Function = null):void
```

Creates the Lecture entry audit record in audit_lecture table. This method is called when the user enters the session.

Parameters

`auditLectureVO:AuditLectureVO` — The reference to the calling component/class. Used by this class to call back in the result/fault handlers. The reference is assigned to AsyncToken and then accessed by the result/fault handers to call back on this object.

`onResult:Function` — The AuditLectureVO containing the data to be inserted.

`onFault:Function` (default = `null`)

updateAuditLectureById() method

```
public function updateAuditLectureById(auditLectureId:Number, onResult:Function, onFault
:Function = null):void
```

Updates the Lecture entry audit record, which was created during session entry, when the user exits a session.

Parameters

`auditLectureId:Number` — The reference to the calling component/class. Used by this class to call back in the result/fault handlers. The reference is assigned to AsyncToken and then accessed by the result/fault handers to call back on this object.

`onResult:Function` — The Primary key reference to the lecture entry audit record, which will be updated.

`onFault:Function` (default = `null`)

AuditUserLoginHelper

Package edu.amrita.aview.core.shared.audit.helper
Class public class AuditUserLoginHelper
Inheritance AuditUserLoginHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Helper class recording user login/logout details into the database. Stores/Retrieves the contents of UserLoginVO

Public Methods

	Method	Defined By
	AuditUserLoginHelper() constructor Initializes the remote objects and registers the result and fault handlers	AuditUserLoginHelper
	createUserLogin(userLoginVO:UserLoginVO, onResult:Function, onFault:Function = null):void Called after successful login into A-VIEW.	AuditUserLoginHelper
	updateUserLogin(userLoginVO:UserLoginVO, onResult:Function, onFault:Function = null):void Called during logout.	AuditUserLoginHelper

Constructor Detail

AuditUserLoginHelper() Constructor

`public function AuditUserLoginHelper()`

constructor Initializes the remote objects and registers the result and fault handlers

Method Detail

createUserLogin() method

```
public function createUserLogin(userLoginVO:UserLoginVO, onResult:Function, onFault:
Function = null):void
```

Called after successful login into A-VIEW. The created date and updated date of the record will be same as the login time. The updated date gets updated during logout, while created date stays same, indicating the login time

Parameters

`userLoginVO:UserLoginVO` — The reference to the calling component/class. Used by this class to call back in the result/fault handlers. The reference is assigned to AsyncToken and then accessed by the result/fault handers to call back on this object.

`onResult:Function` — UserLoginVOs details, which are stored in the database.

`onFault:Function` (default = `null`)

updateUserLogin() method

`public function updateUserLogin(userLoginVO:UserLoginVO, onResult:Function, onFault: Function = null):void`

Called during logout. It updates the login database entry with the logout time (updateddate)

Parameters

`userLoginVO:UserLoginVO` — The reference to the calling component/class. Used by this class to call back in the result/fault handlers. The reference is assigned to AsyncToken and then accessed by the result/fault handers to call back on this object.

`onResult:Function` — The UserLoginVO object would be updated with the log out time on the server.

`onFault:Function` (default = `null`)

UserActionHelper

Package edu.amrita.aview.core.shared.audit.helper
Class public class UserActionHelper
Inheritance UserActionHelper → edu.amrita.aview.core.shared.helper.AbstractHelper

Helper class for storing the User's actions across all the modules and inside and outside session into the database For performance reasons, the actions are batched and sent to database once every 5 minutes.

Public Methods

	Method	Defined By
	UserActionHelper() constructor Initializes the remote objects and registers the result and fault handlers	UserAction Helper
	createUserAction(userAction:UserActionVO):void This method is called by AuditUserAction for storing the action into the database.	UserAction Helper
	databaseInsert():void Sends all the pending UserActionVOs to database and emptys the list	UserAction Helper

Constructor Detail

UserActionHelper() Constructor

```
public function UserActionHelper()
```

constructor Initializes the remote objects and registers the result and fault handlers

Method Detail

createUserAction() method

```
public function createUserAction(userAction:UserActionVO):void
```

This method is called by AuditUserAction for storing the action into the database. For performance reasons, this method does not actually send Action to database, it simply stores in a local list. Every 5 minutes, the local list is sent over to database.

Parameters

`userAction:UserActionVO` — UserActionVO contains the details of the action

databaseInsert() method

```
public function databaseInsert():void
```

Sends all the pending UserActionVOs to database and emptys the list

---

Server Side APIs

EXCEPTION

Class AViewException

```
 java.lang.Object
 └ java.lang.Throwable
       └ java.lang.Exception └
            com.amrita.edu.AViewException
```

All Implemented Interfaces:
Serializable

```
public class AViewException
extends Exception
```

See Also:

Serialized Form

Constructor Summary

`AViewException(String cause)`

Method Summary

Methods inherited from class java.lang. Throwable
`fillInStackTrace, getCause, getLocalizedMessage, getMessage, getStackTrace, initCause, printStackTrace, printStackTrace, printStackTrace, setStackTrace, toString`

Methods inherited from class java.lang. Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

AViewException

```
public AViewException(String cause)
```

HELPER

CLASS HELPER

```
java.lang.Object
com.amrita.edu.helper.ClassHelper
```

```
public class ClassHelper
extends Object
```

Constructor Summary

`ClassHelper()`

Method Summary

`static javax.ws.rs.core.Response`	`classdelete(Long adminId, Long classId)` This method is used to delete class
`static javax.ws.rs.core.Response`	`createClass(Long adminId, String classDetails, String classServerDetails)` This method is used to create class
`static javax.ws.rs.core.Response`	`searchClass(Long adminId, String className, Long courseId, Long instituteId)` This method is used to search class
`static javax.ws.rs.core.Response`	`updateClass(Long adminId, String classDetails,String classServerDetails)` This method is used to update class

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ClassHelper

```
public ClassHelper()
```

Method Detail

classdelete

```
public static javax.ws.rs.core.Response classdelete(Long
                                                     adminId, Long
                                                     classId)
                                              throws AViewException
```

This method is used to delete class

Parameters:
: `adminId -`
: `classId -`

Returns:
: Response

Throws:
: `AViewException`

createClass

```
public static javax.ws.rs.core.Response createClass(Long adminId,
                                                    String classDetails, String
                                                    classServerDetails)
                                             throws AViewException
```

This method is used to create class

Parameters:
: `adminId -`
: `classDetails -`
: `classServerDetails -`

Returns:
: Response

Throws:
: `AViewException`

searchClass

```
public static javax.ws.rs.core.Response searchClass(Long
                                                    adminId, String
                                                    className, Long
                                                    courseId, Long
                                                    instituteId)
                                             throws AViewException
```

This method is used to search class

Parameters:
: `adminId -`
: `className -`
: `courseId -`
: `instituteId -`

Returns:
: Response

Throws:
: `AViewException`

updateClass

```
public static javax.ws.rs.core.Response updateClass(Long adminId,
                                                    String classDetails, String
                                                    classServerDetails)
```

```
throws AViewException,
       org.codehaus.jettison.json.JSONException
```

This method is used to update class

Parameters:

`adminId` -

`classDetails` -

`classServerDetails` -

Returns:

Response

Throws:

`AViewException` `org.codehaus.jettison.json.JSONException`

Class Registration Helper

```
java.lang.Object
  └com.amrita.edu.helper.ClassRegistrationHelper
```

```
public class ClassRegistrationHelper
extends Object
```

Constructor Summary

`ClassRegistrationHelper()`

Method Summary

`static javax.ws.rs.core.Response`	`createClassRegistration(Long userId, Long classId, String isModerator, Long adminId)` This method is used to create class registration
`static javax.ws.rs.core.Response`	`deleteClassRegister(Long adminId, Long classRegisterId)` This method is used to delete classRegistration
`static javax.ws.rs.core.Response`	`searchClassRegister(Long adminId, Long classId, Long userId)` This method is used to search classRegistration
`static javax.ws.rs.core.Response`	`updateClassRegister(Long adminId, Long classRegisterId, String isModerator)` This method is used to update class registration

Methods inherited from class java.lang.Object

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

Constructor Detail

ClassRegistrationHelper

```
public ClassRegistrationHelper()
```

Method Detail

createClassRegistration

```
public static javax.ws.rs.core.Response createClassRegistration(Long
                                                                 userId, Long
                                                                 classId,
                                                                 String
                                                                 isModerator, Long
                                                                 adminId)
                                                          throws AViewException
```

This method is used to create class registration

Parameters:
- `userId -`
- `classId -`
- `isModerator -`
- `adminId -`

Returns:
Response

Throws:
`AViewException`

deleteClassRegister

```
public static javax.ws.rs.core.Response deleteClassRegister(Long adminId,
                                                             Long classRegisterId)
                                                      throws AViewException
```

This method is used to delete classRegistration

Parameters:

```
adminId -
classRegisterId -
```

CourseHelper

```
java.lang.Object
  com.amrita.edu.helper.CourseHelper
```

```
public class CourseHelper
extends Object
```

Constructor Summary
CourseHelper()

Method Summary	
static javax.ws.rs.core.Response	searchCourses(Long adminId, String courseName, Long instituteId) This method is used to search course

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CourseHelper

```
public CourseHelper()
```

Method Detail

searchCourses

```
public static javax.ws.rs.core.Response searchCourses(Long
                                                        adminId, String
                                                        courseName, Long
```

instituteId)

throws AViewException

This method is used to search course

Parameters:
: `adminId -`
: `courseName -`
: `instituteId -`

Returns:
: Response

Throws:
: AViewException

Class DocumentUploadHelper

```
java.lang.Object
  com.amrita.edu.helper.DocumentUploadHelper
```

```
public class DocumentUploadHelper
extends Object
```

Constructor Summary
`DocumentUploadHelper()`

Method Summary	
`javax.ws.rs.core.Response`	`uploadFile(LongmoderatorId,org.springframework.web.multipMultipartFilemultipartFile, Long classRegisterId, String isAnimatedDocument)` This method is used to Upload file

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DocumentUploadHelper

```
public DocumentUploadHelper()
```

Method Detail

uploadFile

```
public javax.ws.rs.core.Response uploadFile(Long moderatorId,
                                            org.springframework.web.multipart.MultipartFile
                                            multipartFile, Long classRegisterId,
                                            String isAnimatedDocument)
                                     throws AViewException
```

This method is used to Upload file

Parameters:

`moderatorId` -

`multipartFile` -

`classRegisterId` -

`isAnimatedDocument` -

Returns:

Response

Throws:

`AViewException`

Class InstituteHelper

```
java.lang.Object
  com.amrita.edu.helper.InstituteHelper
```

```
public class InstituteHelper
extends Object
```

Constructor Summary

InstituteHelper()

Method Summary

Static javax.ws.rs.core.Response	searchInstitute(Long adminId, String instituteName) This method is used to search institute

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

InstituteHelper

```
public InstituteHelper()
```

Method Detail

searchInstitute

```
public static javax.ws.rs.core.Response searchInstitute(Long adminId,
                                                        String instituteName)
                                                 throws AViewException
```

This method is used to search institute

Parameters:

`adminId` -

`instituteName` -

Returns:

Response

Throws:

`AViewException`

Class LectureHelper

```
java.lang.Object
  com.amrita.edu.helper.LectureHelper
```

```
public class LectureHelper
extends Object
```

Constructor Summary
LectureHelper()

Method Summary	
static javax.ws.rs.core.Response	createLecture(Long adminId, String lectureDetails) This method is used to create lecture
static javax.ws.rs.core.Response	deleteLecture(Long adminId, Long lectureId) This method is used to delete lecture
static javax.ws.rs.core.Response	lectureSearch(Long adminId, Long classId) This method is used to search lecture
static javax.ws.rs.core.Response	updateLecture(Long adminId, String lectureDetails) This method is used to update lecture

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

LectureHelper

```
public LectureHelper()
```

Method Detail

createLecture

```
public static javax.ws.rs.core.Response createLecture(Long adminId,
                                                      String lectureDetails)
                                               throws AViewException
```

This method is used to create lecture

Parameters:
: `adminId` -
: `lectureDetails` - as JSON

Returns:
: Response

Throws:
: `AViewException`

deleteLecture

```
public static javax.ws.rs.core.Response deleteLecture(Long
                                                      adminId, Long
                                                      lectureId)
                                               throws AViewException
```

This method is used to delete lecture

Parameters:
: `adminId` -
: `lectureId` -

Returns:
: Response

Throws:
: `AViewException`

lectureSearch

```
public static javax.ws.rs.core.Response lectureSearch(Long
                                                      adminId, Long
                                                      classId)
                                               throws AViewException
```

This method is used to search lecture

Parameters:

`adminId -`

`classId -`

Returns:

Response

Throws:

`AViewException`

updateLecture

```
public static javax.ws.rs.core.Response updateLecture(Long adminId,
                                                       String lectureDetails)
                                                throws AViewException
```

This method is used to update lecture

Parameters:

`adminId -`

`lectureDetails -`

Returns:

Response

Throws:

`AViewException`

Class UserHelper

```
java.lang.Object └
  com.amrita.edu.helper.UserHelper
```

```
public class UserHelper
extends Object
```

Constructor Summary
`UserHelper()`

Method Summary

`static javax.ws.rs.core.Response`	`changePassword(Long userId, String newPassword)` This method is used to change password
`static javax.ws.rs.core.Response`	`createUser(Long adminId, String userDetails)` This method is used to create user.
`static javax.ws.rs.core.Response`	`searchUser(Long adminId, String role, String userName, String fname, String lname, String city, Long instituteId)` This method is used to search user.
`static javax.ws.rs.core.Response`	`updateUser(Long adminId, String userDetails)` This method is used to update user
`static javax.ws.rs.core.Response`	`userDelete(Long adminId, Long userId)` This method is used to delete user

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

UserHelper

```
public UserHelper()
```

Method Detail

changePassword

```
public static javax.ws.rs.core.Response changePassword(Long userId,
                                                       String newPassword)
                                                throws AViewException
```

This method is used to change password

Parameters:

`userId -`

`newPassword -`

Returns:
    Response
Throws:
    AViewException

createUser

```
public static javax.ws.rs.core.Response createUser(Long adminId, String
                                                    userDetails)
                                             throws AViewException
```

This method is used to create user.

Parameters:
    `adminId -`
    `userDetails -`

Returns:
    Response
Throws:
    AViewException

searchUser

```
public static javax.ws.rs.core.Response searchUser(Long
                                                    adminId, String
                                                    role, String
                                                    userName, String
                                                    fname, String
                                                    lname, String
                                                    city,
                                                    Long instituteId)
                                             throws AViewException
```

This method is used to search user.

Parameters:
    `adminId -`
    `userName`
    `- fname -`
    `lname -`
    `role -`
    `city -`
    `instituteId -`

Returns:
    Response

Throws:
  AViewException

updateUser

```
public static javax.ws.rs.core.Response updateUser(Long adminId, String
                                                    userDetails)
                                             throws AViewException
```

This method is used to update user

Parameters:
  `adminId -`
  `userDetails -`
Returns:
  response
Throws:
  AViewException

userDelete

```
public static javax.ws.rs.core.Response userDelete(Long
                                                    adminId, Long
                                                    userId)
                                             throws AViewException
```

This method is used to delete user

Parameters:
  `adminId -`
  `userId -`
Returns:
  Response
Throws:
  AViewException

Utils

Class JSONParserUtils

```
java.lang.Object └
 com.amrita.edu.utils.JSONParserUtils
```

```
public class JSONParserUtils
 extends Object
```

Field Summary

String	error

Constructor Summary

JSONParserUtils()

Method Summary

Static Object	readJSONAsObject(String strToRead, Class<?> classToConvert)
static String	writeObjectAsJSON(Object genericObject)

Methods inherited from class java.lang.Object

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

Field Detail

error

```
public String error
```

Constructor Detail

JSONParserUtils

```
public JSONParserUtils()
```

Method Detail	
readJSONAsObject	`public static Object readJSONAsObject(String strToRead, Class<?> classToConvert)`
writeObjectAsJSON	`public static String writeObjectAsJSON(Object genericObject)`