```
package br.com.optimedia.autor.view
{
    import br.com.optimedia.assets.NotificationConstants;
    import br.com.optimedia.assets.vo.MediaVO;
    import br.com.optimedia.assets.vo.PresentationVO;
    import br.com.optimedia.assets.vo.SlideVO;
    import br.com.optimedia.autor.AutorFacade;
    import br.com.optimedia.autor.model.SlideManagerProxy;
    import br.com.optimedia.autor.model.SubjectManagerProxy;
    import br.com.optimedia.autor.view.components.PresentationEditor;
    import br.com.optimedia.autor.view.components.SetOrderPopUp;
    import br.com.optimedia.player.PlayerFacade;

    import flash.events.Event;
    import flash.events.MouseEvent;

    import mx.collections.ArrayCollection;
    import mx.controls.Alert;
    import mx.events.CloseEvent;
    import mx.managers.PopUpManager;

    import org.puremvc.as3.multicore.interfaces.INotification;
    import org.puremvc.as3.multicore.patterns.mediator.Mediator;

    public class PresentationEditorMediator extends Mediator
    {
        public static const NAME:String = 'PresentationEditorMediator';

        private var slideManagerProxy:SlideManagerProxy;
        private var subjectManagerProxy:SubjectManagerProxy;

        public function PresentationEditorMediator(viewComponent:Object=null)
        {
            super(NAME+viewComponent.uid, viewComponent);
        }

        override public function onRegister():void
        {
            trace(NAME+".onRegister()");
            slideManagerProxy = facade.retrieveProxy( SlideManagerProxy.NAME ) as
SlideManagerProxy;
            subjectManagerProxy = facade.retrieveProxy( SubjectManagerProxy.NAME ) as
SubjectManagerProxy;
            view.backBtn.addEventListener(MouseEvent.CLICK, backBtnClick);
            view.newSlideBtn.addEventListener(MouseEvent.CLICK, addSlide);
            view.slideEditBtn.addEventListener(MouseEvent.CLICK, editSlide);
            view.slideRemoveBtn.addEventListener(MouseEvent.CLICK, deleteSlide);
            view.setOrderBtn.addEventListener(MouseEvent.CLICK, setOrder);
            view.saveSlideBtn.addEventListener(MouseEvent.CLICK, saveSlide);
        }

        override public function onRemove():void {
            trace(NAME+".onRemove()");
        }
```

```
        public function get view():PresentationEditor
        {
            return viewComponent as PresentationEditor;
        }

        override public function listNotificationInterests():Array
        {
            return [NotificationConstants.BEGIN_PRESENTATION_EDIT,
                    NotificationConstants.GET_SLIDES_OK,
                    NotificationConstants.UNLOCK_PRESENTATION_OK,
                    NotificationConstants.SAVE_SLIDE_RESULT,
                    NotificationConstants.DO_LINK_EVENT,
                    NotificationConstants.ADD_NEW_SLIDE_RESULT,
                    NotificationConstants.DELETE_SLIDE_RESULT,
                    NotificationConstants.SET_SLIDE_ORDER_RESULT,
                    NotificationConstants.ENABLE_SLIDE_EDITION,
                    NotificationConstants.DISABLE_SLIDE_EDITION];
        }

        override public function handleNotification(note:INotification):void
        {
            switch (note.getName())
            {
                case NotificationConstants.BEGIN_PRESENTATION_EDIT:
                    view.presentationVO = note.getBody() as PresentationVO;
                    slideManagerProxy.getSlides( view.presentationVO.presentation_id );
                    view.playerModule.visible = false;
                    view.repositoryPanel.linkBtn.visible = false;
                    view.playerModule.setSlide( ArrayCollection(view.presentationVO.slidesArray
).getItemAt(0) as SlideVO );
                    view.newSlideBtn.enabled = false;
                    view.slideEditBtn.enabled = false;
                    view.slideRemoveBtn.enabled = false;
                    view.setOrderBtn.enabled = false;
                    view.repositoryPanel.deleteBtn.enabled = false;
                    break;
                case NotificationConstants.ENABLE_SLIDE_EDITION:
                    view.presentationVO.locked_by = AutorFacade(facade).userID;
                    view.newSlideBtn.enabled = true;
                    view.slideEditBtn.enabled = true;
                    view.slideRemoveBtn.enabled = true;
                    view.setOrderBtn.enabled = true;
                    view.repositoryPanel.deleteBtn.enabled = true;
                    break;
                case NotificationConstants.DISABLE_SLIDE_EDITION:
                    view.presentationVO.locked_by = note.getBody() as int;
                    view.newSlideBtn.enabled = false;
                    view.slideEditBtn.enabled = false;
                    view.slideRemoveBtn.enabled = false;
                    view.setOrderBtn.enabled = false;
                    view.repositoryPanel.deleteBtn.enabled = false;
                    if( AutorFacade(facade).userRole == AutorFacade.IS_OBSERVER ) {
                        view.repositoryPanel.addBtn.enabled = false;
```

```
                    }
                    else {
                        view.repositoryPanel.addBtn.enabled = true;
                    }
                    break;
                case NotificationConstants.GET_SLIDES_OK:
                    view.presentationVO.slidesArray = note.getBody() as Array;
                    PlayerFacade.getInstance().sendNotification(
NotificationConstants.GET_PRESENTATION_FOR_PLAYER, note.getBody());
                    view.playerModule.visible = true;
                    if( view.presentationVO.slidesArray.length <= 1 ) {
                        view.slideRemoveBtn.enabled = false;
                    }
                    else {
                        view.slideRemoveBtn.enabled = view.newSlideBtn.enabled;
                    }
                    break;
                case NotificationConstants.UNLOCK_PRESENTATION_OK:
                    sendNotification( NotificationConstants.BACK_TO_SUBJECT_MANAGER );
                    break;
                case NotificationConstants.SAVE_SLIDE_RESULT:
                    var currentSlide:SlideVO = view.playerModule.slideVO;
                    var index:int = view.playerModule.slidesArray.getItemIndex( currentSlide );
                    view.playerModule.slidesArray.setItemAt(note.getBody() as SlideVO, index);
                    view.playerModule.setSlide( note.getBody() as SlideVO );
                    view.viewStack.selectedIndex--;
                    break;
                case NotificationConstants.DO_LINK_EVENT:
                    if( view.bodyTextArea.selection.beginIndex !=
view.bodyTextArea.selection.endIndex ) {
                        view.bodyTextArea.selection.url = "event:"+MediaVO(note.getBody()).
media_id;
                        view.bodyTextArea.selection.textDecoration = "underline";
                    }
                    else {
                        Alert.show("É necessário selecionar um texto antes.", "Atenção");
                    }
                    break;
                case NotificationConstants.ADD_NEW_SLIDE_RESULT:
                    var slide:SlideVO = note.getBody() as SlideVO;
                    var slidesArray:ArrayCollection = view.presentationVO.slidesArray;
                    slidesArray.addItemAt(slide, slide.page_order-1);
                    view.playerModule.setSlide( slide );
                    slideManagerProxy.getSlides( view.presentationVO.presentation_id );
                    break;
                case NotificationConstants.DELETE_SLIDE_RESULT:
                    var newSlideArray:ArrayCollection = new ArrayCollection( note.getBody() as
Array );
                    var pageIndex:int = view.currentSlide.page_order-1;
                    var newSlide:SlideVO;
                    if( pageIndex == newSlideArray.length ) {
                        newSlide = newSlideArray.getItemAt( pageIndex-1 ) as SlideVO;
                    }
                    else {
```

```
                    newSlide = newSlideArray.getItemAt( pageIndex ) as SlideVO;
                }
                view.presentationVO.slidesArray = newSlideArray;
                view.playerModule.slidesArray = newSlideArray;
                view.playerModule.setSlide( newSlide );
                break;
            case NotificationConstants.SET_SLIDE_ORDER_RESULT:
                var newSlideArray1:ArrayCollection = new ArrayCollection( note.getBody() as
Array );
                var pageIndex1:int = view.currentSlide.page_order-1;
                var newSlide1:SlideVO = newSlideArray1.getItemAt( pageIndex1 ) as SlideVO;
                view.presentationVO.slidesArray = newSlideArray1;
                view.playerModule.slidesArray = newSlideArray1;
                view.playerModule.setSlide( newSlide1 );
                break;
            default:
                break;
        }
    }

    private function backBtnClick(e:Event):void {
        if( AutorFacade(facade).userID == view.presentationVO.locked_by ) {
            subjectManagerProxy.unlockPresentation( view.presentationVO.presentation_id );
        }
        else {
            sendNotification( NotificationConstants.BACK_TO_SUBJECT_MANAGER );
        }
        view.viewStack.selectedIndex = 0;
    }

    private function addSlide(e:Event):void {
        var slide:SlideVO = new SlideVO();
        slide.presentation_id = view.presentationVO.presentation_id;
        slide.page_order = uint(view.playerModule.slideVO.page_order)+ 1;
        slide.title = "Novo Slide "+(view.playerModule.slideVO.page_order+1).toString();

        //ArrayCollection(presentationVO.slidesArray).addItemAt(slide, int( slide.page_order
 ));


        //slideSelector.presentationVO = presentationVO;

        slideManagerProxy.addNewSlide( slide );
    }
    private function editSlide(e:Event):void {
        view.viewStack.selectedIndex++;
    }
    private function deleteSlide(e:Event):void {
        if( view.playerModule.slidesArray.length <= 1 ) {
            Alert.show("Não foi possível remover o slide, a apresentação precisa ter pelo
menos 1 slide.", "Atenção");
        }
        else {
            Alert.noLabel = "Não";
            Alert.yesLabel = "Sim";
```

```
                Alert.show("Tem certeza que deseja remover este slide?", "Atenção", 3, null,
deleteSlideAlertHandler);
                function deleteSlideAlertHandler(e:CloseEvent):void {
                    if( e.detail == Alert.YES ) {
                        slideManagerProxy.deleteSlide( view.currentSlide );
                    }
                }
            }
        }
        private function setOrder(e:Event):void {
            var setOrderPopUp:SetOrderPopUp = new SetOrderPopUp();

            setOrderPopUp.slideArray = view.presentationVO.slidesArray as ArrayCollection;

            setOrderPopUp.addEventListener(SetOrderPopUp.SAVE_NEW_ORDER, function (e:Event):void
 { slideManagerProxy.setOrder( setOrderPopUp.slideArray );

PopUpManager.removePopUp( setOrderPopUp ); });
            PopUpManager.addPopUp(setOrderPopUp, view, true);
        }
        private function saveSlide(e:Event):void {
            var slide:SlideVO = new SlideVO();
            slide = view.currentSlide.clone();
            slide.title = view.titleTextInput.text;
            slide.text_body = view.bodyTextArea.htmlText;
            var regExp:RegExp = /event:\d*/g;
            var eventArray:Array = view.bodyTextArea.htmlText.match(regExp);
            ArrayCollection(slide.mediaArray).removeAll();
            for each(var string:String in eventArray) {
                ArrayCollection(slide.mediaArray).addItem(string.substring(6, string.length+1));
            }
            slideManagerProxy.saveSlide( slide );
        }
    }
}
```