

```

package br.com.optimedia.autor.model
{

    import br.com.optimedia.assets.FaultHandler;
    import br.com.optimedia.assets.NotificationConstants;
    import br.com.optimedia.assets.vo.FileVO;
    import br.com.optimedia.assets.vo.MediaVO;
    import br.com.optimedia.assets.vo.QuestionVO;

    import mx.controls.Alert;
    import mx.rpc.AsyncToken;
    import mx.rpc.Responder;
    import mx.rpc.events.FaultEvent;
    import mx.rpc.events.ResultEvent;
    import mx.rpc.remoting.mxml.RemoteObject;

    import org.puremvc.as3.multicore.patterns.proxy.Proxy;

    public class RepositoryManagerProxy extends Proxy
    {
        public static const NAME:String = "RepositoryManagerProxy";

        private var remoteService:RemoteObject;

        public function RepositoryManagerProxy(data:Object=null)
        {
            super(NAME, data);
        }

        override public function onRegister():void {
            trace(NAME+".onRegister()");
            remoteService = new RemoteObject();
            remoteService.destination = "amfphp";
            remoteService.source = "autor.RepositoryManager";
            remoteService.showBusyCursor = false;
        }

        private function generalFault(event:FaultEvent):void {
            FaultHandler.handleFault(event);
        }

        public function getMedias( subjectID:uint ):void {
            var asynkToken:AsyncToken = remoteService.getMedias(
subjectID );
            asynkToken.addResponder( new Responder(getMediasResult,
generalFault) );
            /* var xml:XML =
                <medias>
                    <cat name="Tabelas">
                        <item name="blal"
ref="referencial" icon="table" category_id="1"/>
                        <item name="bla2"
ref="referencial" icon="table" category_id="1"/>
                        <item name="bla3"
ref="referencial" icon="table" category_id="1"/>
                    </cat>
                    <cat name="Gráficos">
                        <item name="blal"
ref="referencial" category_id="1"/>

```

```

        <item name="bla2"
ref="referencial" category_id="1"/>
        <item name="bla3"
ref="referencial" category_id="1"/>
    </cat>
    <cat name="Imagens">
        <item name="bla1"
ref="referencial" icon="" id=""/>
        <item name="bla2"
ref="referencial" icon="" id=""/>
        <item name="bla3"
ref="referencial" icon="" id=""/>
    </cat>
    <cat name="Links">
        <item name="bla1"
ref="referencial" icon="" id=""/>
        <item name="bla2"
ref="referencial" icon="" id=""/>
        <item name="bla3"
ref="referencial" icon="" id=""/>
    </cat>
    <cat name="VÍdeos">
        <item name="bla1"
ref="referencial" icon="" id=""/>
        <item name="bla2"
ref="referencial" icon="" id=""/>
        <item name="bla3"
ref="referencial" icon="" id=""/>
    </cat>
    <cat name="Texto">
        <item name="bla1"
ref="referencial" icon="" id=""/>
        <item name="bla2"
ref="referencial" icon="" id=""/>
        <item name="bla3"
ref="referencial" icon="" id=""/>
    </cat>
</medias>;

    var event:ResultEvent = new
ResultEvent(ResultEvent.RESULT, false, true, xml);
    getMediasResult( event ); */
}
private function getMediasResult(event:ResultEvent):void {
    if( event.result is Array ) sendNotification(
NotificationConstants.GET_MEDIAS_RESULT, event.result );
}

    public function uploadPresentationFile(fileVO:FileVO,
presentationID:uint, type:String):void {
        var asynkToken:AsyncToken =
remoteService.uploadPresentationFile(fileVO, presentationID, type);
        asynkToken.addResponder( new
Responder(uploadPresentationFileResult, generalFault) );
    }
    private function
uploadPresentationFileResult(event:ResultEvent):void {
        if( event.result == true ) sendNotification(
NotificationConstants.UPLOAD_PRESENTATION_FILE_RESULT, event.result );
    }
}

```

```

    }

    public function uploadMediaFile(fileVO:FileVO,
mediaVO:MediaVO, subjectID:uint):void {
        var asyncToken:AsyncToken =
remoteService.uploadMediaFile(fileVO, mediaVO, subjectID);
        asyncToken.addResponder( new
Responder(uploadMediaFileResult, generalFault) );
    }
    private function uploadMediaFileResult(event:ResultEvent):void
{
        sendNotification(
NotificationConstants.UPLOAD_MEDIA_FILE_RESULT, event.result );
    }

    public function uploadMediaText(mediaVO:MediaVO,
subjectID:uint):void {
        var asyncToken:AsyncToken =
remoteService.uploadMediaText(mediaVO, subjectID);
        asyncToken.addResponder( new
Responder(uploadMediaTextResult, generalFault) );
    }
    private function uploadMediaTextResult(event:ResultEvent):void
{
        sendNotification(
NotificationConstants.UPLOAD_MEDIA_TEXT_RESULT, event.result );
    }

    public function deleteMedia(mediaID:uint):void {
        var asyncToken:AsyncToken =
remoteService.deleteMedia(mediaID);
        asyncToken.addResponder( new
Responder(deleteMediaResult, generalFault) );
    }
    private function deleteMediaResult(event:ResultEvent):void {
        if(event.result == true) {
            sendNotification(
NotificationConstants.DELETE_MEDIA_OK );
        }
        else {
            Alert.show("Não foi possível remover a mídia. Ela
pode estar associada em algum slide.", "Erro");
        }
    }

    public function saveQuestion(media:MediaVO,
question:QuestionVO, subjectID:uint):void {
        var asyncToken:AsyncToken =
remoteService.saveQuestion(media, question, subjectID);
        asyncToken.addResponder( new
Responder(creationQuestionResult, generalFault) );
    }
    private function
creationQuestionResult(event:ResultEvent):void {
        if (event.result ==true) {
            sendNotification(
NotificationConstants.CREATION_QUESTION_RESULT, event.result );
        }else {
            Alert.show("erro creation question");
        }
    }

```

}
}
}
}