```
package br.com.optimedia.player.model
{
    import br.com.optimedia.assets.FaultHandler;
    import br.com.optimedia.assets.NotificationConstants;
    import br.com.optimedia.assets.vo.QuestionVO;
    import br.com.optimedia.assets.vo.SlideVO;

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

        private var remoteService:RemoteObject;

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

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

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

        public function getPresentation(presentationID:int):void {
            var asynkToken:AsyncToken = remoteService.getPlayerSlides(presentationID);
            asynkToken.addResponder( new Responder(getPresentationResult, generalFault) );
        }
        private function getPresentationResult(event:ResultEvent):void {
            if( event.result is Array ) {
                sendNotification( NotificationConstants.GET_PRESENTATION_FOR_PLAYER,
event.result );
            }
        }

        public function registerNavigation(userID:int, presentationID:int, slideID:int):void {
            var asynkToken:AsyncToken = remoteService.registerNavigation(userID, presentationID,
 slideID);
            asynkToken.addResponder( new Responder(registerNavigationResult, generalFault) );
        }
        private function registerNavigationResult(event:ResultEvent):void {
            if( event.result == true ) {
                trace("registerNavigationResult");
            }
        }

        public function getLastViewedSlide(userID:int, presentationID:int):void {
            var asynkToken:AsyncToken = remoteService.getLastViewedSlide(userID, presentationID);
            asynkToken.addResponder( new Responder(getLastViewedSlideResult, generalFault) );
        }
        private function getLastViewedSlideResult(event:ResultEvent):void {
            if( event.result is SlideVO ) {
                var slideID:int = SlideVO(event.result).slide_id;
                sendNotification( NotificationConstants.GET_LAST_VIEWED_SLIDE_RESULT, slideID );
            }
            else {
                sendNotification( NotificationConstants.GET_LAST_VIEWED_SLIDE_RESULT, 0 );
                trace("getLastViewedSlideResult == false");
            }
        }

        public function getQuestion(questionID:int):void {
            var asynkToken:AsyncToken = remoteService.getQuestion(questionID);
            asynkToken.addResponder( new Responder(getQuestionResult, generalFault) );
        }
        private function getQuestionResult(event:ResultEvent):void {
            if( event.result is QuestionVO ) {
                sendNotification( NotificationConstants.GET_QUESTION_RESULT, event.result );
            }
       }
    }
}
```