```
package br.com.optimedia.autor
{
    import br.com.optimedia.assets.CommandConstants;
    import br.com.optimedia.autor.controller.AutorStartupCommand;
    import br.com.optimedia.autor.controller.CommentListDisposeCommand;
    import br.com.optimedia.autor.controller.CommentListStartupCommand;
    import br.com.optimedia.autor.controller.ModelStartupCommand;
    import br.com.optimedia.autor.controller.PublishPresentationDisposeCommand;
    import br.com.optimedia.autor.controller.PublishPresentationStartupCommand;
    import br.com.optimedia.autor.controller.RepositoryPanelStartupCommand;
    import br.com.optimedia.autor.controller.SendFilePopUpDisposeCommand;
    import br.com.optimedia.autor.controller.SendFilePopUpStartupCommand;
    import br.com.optimedia.autor.controller.SendMediaDisposeCommand;
    import br.com.optimedia.autor.controller.SendMediaStartupCommand;
    import br.com.optimedia.autor.controller.SlideEditorStartupCommand;
    import br.com.optimedia.autor.controller.SubjectManagerStartupCommand;
    import br.com.optimedia.autor.view.components.CommentList;
    import br.com.optimedia.autor.view.components.PresentationEditor;
    import br.com.optimedia.autor.view.components.PublishPresentationPopUp;
    import br.com.optimedia.autor.view.components.RepositoryPanel;
    import br.com.optimedia.autor.view.components.SendFilePopUp;
    import br.com.optimedia.autor.view.components.SendMediaPopUp;
    import br.com.optimedia.autor.view.components.SubjectManager;

    import org.puremvc.as3.multicore.patterns.facade.Facade;

    public class AutorFacade extends Facade
    {

        public static const IS_ADMIN:String = "IS_ADMIN";
        public static const IS_AUTHOR:String = "IS_AUTHOR";
        public static const IS_EDITOR:String = "IS_EDITOR";
        public static const IS_OBSERVER:String = "IS_OBSERVER";

        public var userID:int;
        public var userRole:String;

        public function AutorFacade(key:String)
        {
            super(key);
        }

        /**
         * Singleton ApplicationFacade Factory Method
         */
        public static function getInstance( key:String = "default" ) : AutorFacade
        {
            if ( instanceMap[ key ] == null ) instanceMap[ key ] = new AutorFacade( key );
            return instanceMap[ key ] as AutorFacade;
        }

        /**
         * Register Commands with the Controller
         */
```

```
        override protected function initializeController( ) : void
        {
            super.initializeController();
            //STARTUP COMMANDS
            registerCommand( CommandConstants.MODEL_STARTUP, ModelStartupCommand );
            registerCommand( CommandConstants.AUTOR_STARTUP, AutorStartupCommand );
            registerCommand( CommandConstants.SUBJECT_MANAGER_STARTUP,
SubjectManagerStartupCommand );
            registerCommand( CommandConstants.REPOSITORY_PANEL_STARTUP,
RepositoryPanelStartupCommand );
            registerCommand( CommandConstants.SEND_FILE_POPUP_STARTUP,
SendFilePopUpStartupCommand );
            registerCommand( CommandConstants.SEND_MEDIA_POPUP_STARTUP, SendMediaStartupCommand
);
            registerCommand( CommandConstants.SLIDE_EDITOR_STARTUP, SlideEditorStartupCommand );
            registerCommand( CommandConstants.PUBLISH_PRESENTATION_STARTUP,
PublishPresentationStartupCommand );
            registerCommand( CommandConstants.COMMENT_LIST_STARTUP, CommentListStartupCommand );

            //DISPOSE COMMANDS
            registerCommand( CommandConstants.SEND_FILE_POPUP_DISPOSE,
SendFilePopUpDisposeCommand );
            registerCommand( CommandConstants.SEND_MEDIA_POPUP_DISPOSE, SendMediaDisposeCommand
);
            registerCommand( CommandConstants.PUBLISH_PRESENTATION_DISPOSE,
PublishPresentationDisposeCommand );
            registerCommand( CommandConstants.COMMENT_LIST_DISPOSE, CommentListDisposeCommand );
        }

        /**
         * Application startup
         *
         * @param app a reference to the application component
         */
        public function startup( app:Object ):void
        {
            if (app == "model") sendNotification( CommandConstants.MODEL_STARTUP, null );
            else if (app is Autor) sendNotification( CommandConstants.AUTOR_STARTUP, app );
            else if (app is SubjectManager) sendNotification(
CommandConstants.SUBJECT_MANAGER_STARTUP, app );
            else if (app is RepositoryPanel) sendNotification(
CommandConstants.REPOSITORY_PANEL_STARTUP, app );
            else if (app is SendFilePopUp) sendNotification(
CommandConstants.SEND_FILE_POPUP_STARTUP, app );
            else if (app is SendMediaPopUp) sendNotification(
CommandConstants.SEND_MEDIA_POPUP_STARTUP, app );
            else if (app is PresentationEditor) sendNotification(
CommandConstants.SLIDE_EDITOR_STARTUP, app );
            else if (app is PublishPresentationPopUp) sendNotification(
CommandConstants.PUBLISH_PRESENTATION_STARTUP, app );
            else if (app is CommentList) sendNotification( CommandConstants.COMMENT_LIST_STARTUP
, app );
        }
```

```
        public function dispose( app:Object ):void
        {
            if (app is SendFilePopUp) sendNotification( CommandConstants.SEND_FILE_POPUP_DISPOSE
, app );
            else if (app is SendMediaPopUp) sendNotification(
CommandConstants.SEND_MEDIA_POPUP_DISPOSE, app );
            else if (app is PublishPresentationPopUp) sendNotification(
CommandConstants.PUBLISH_PRESENTATION_DISPOSE, app );
            else if (app is CommentList) sendNotification( CommandConstants.COMMENT_LIST_DISPOSE
, app );
        }
    }
}
```