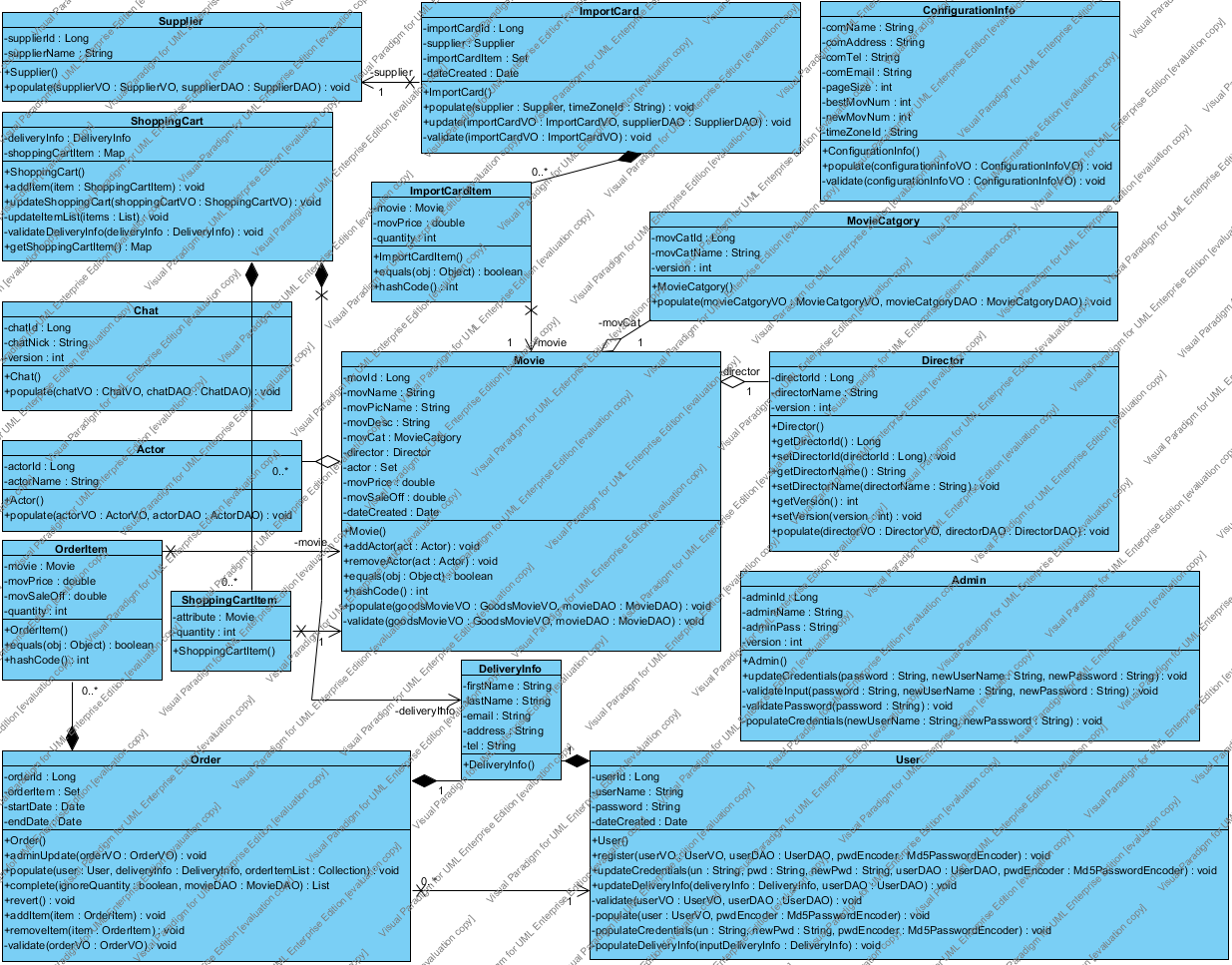
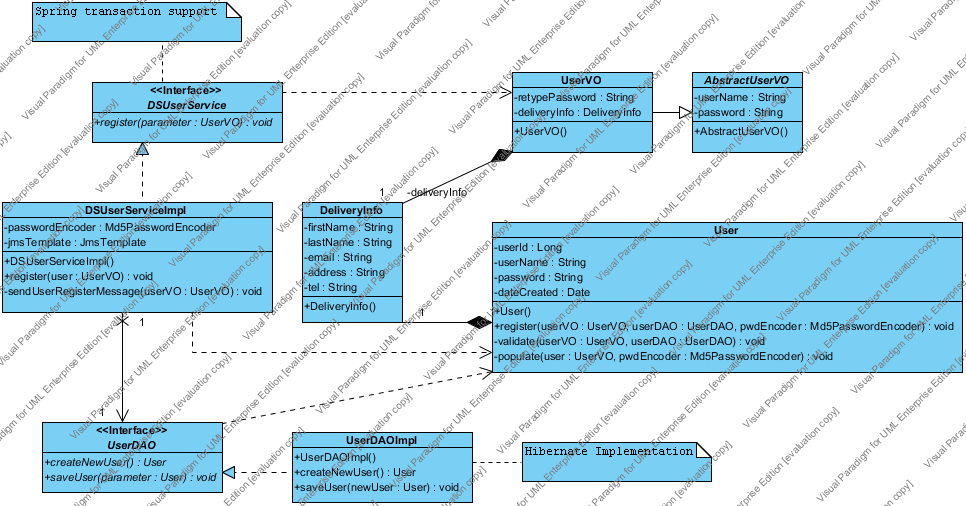
**X. ClassDiagrams:**

Domain Model Classes



Class Diagram

# DangKy\_BL



- Sử dụng Spring Transaction Support: JTA, Hibernate Transaction, JDBC Transaction. Nếu không sử dụng spring, có thể phải hard code 🡪 thay đổi khó, spring chỉ config qua xml, về nguyên tắc, spring tạo 1 proxy wrap lại object lúc init application context. Proxy này có nhiệm vụ start transaction lúc bắt đầu thực hiện method, gọi method của wrapped object, và commit transaction khi method không throw exception, trường hợp có throw exception thì rollback transaction.

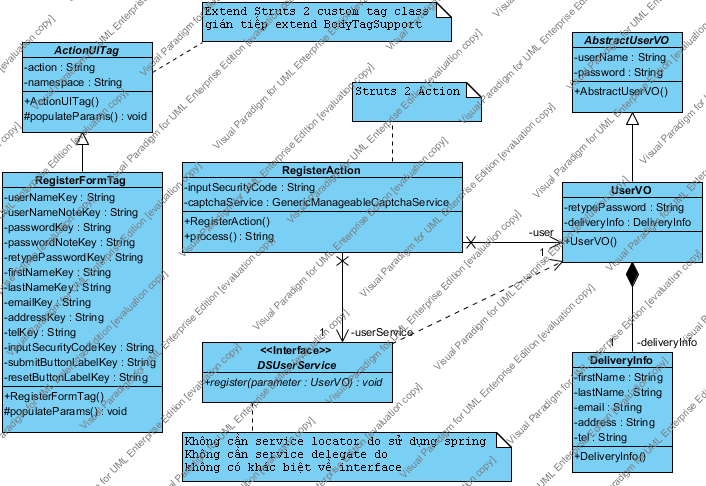
- Đây chính là khái niệm AOP (Aspect Orient Programming) developer chỉ cần tập trung xây dựng bussiness logic, không cần quan tâm tới security hay transaction. Việc áp dụng AOP giúp dễ dàng hơn cho quá trình phát triển, cũng giúp dễ dàng hơn khi maintainent, và unit test (chỉ test bussiness logic, không cần test security hay transaction).

- User.register() nhận tham số UserDAO do đây là entity 🡪 do DAO sinh, 1 số ORM framework có khả năng tạo entity với constructor có tham số, tuy nhiên nếu làm vậy có thể bị gắn với 1 số framework nhất định, vậy đưa dependency vào dưới dạng tham số (spring chỉ quản lý service và dao object, entity do hibernate framework quản lý, do vậy không thể inject, cũng không nên tự tạo dependency bằng toán tử new do không thể unit test).

- Có thể thấy 2 phương thức private validate và populate (phương thức cập nhật instance variable) chính là 2 phương thức chính chưa bussiness logic được encapsulate trong User. Tạo ra khả năng sử dụng lại code

- Có thể thấy mối quan hệ giữa Service Object với DAO là qua DAO interface

# DangKy\_P



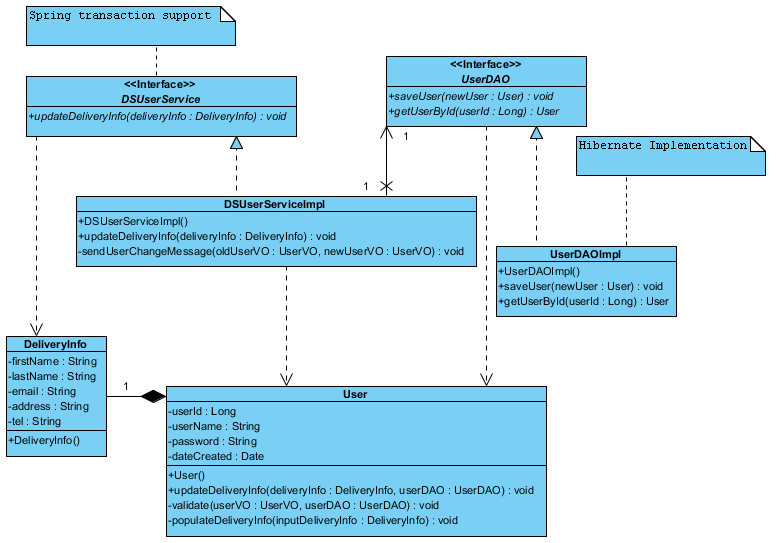
- Các trang web jsp chỉ làm nhiệm vụ layout, web được chia thành các component riêng rẽ, được thực thi dưới dạng các custom tag, thiết kế này tạo ra khả năng linh hoạt, dùng lại code, dễ dàng cho thay đổi

- Các custom tag tận dụng khả năng của struts 2 UI support. Cho khả năng quản lý theme cũng như tính linh hoạt khi không muốn sử dụng custom tag của jsp

- Struts 2 Action là sự cải tiến lớn: thực sự trở thành 1 plain old java object, không bị phụ thuộc vào framework đặc biệt là servlet api. (Sử dụng các interceptor cho các thao tác với servlet api) 🡪 tăng khả năng dùng lại code và cũng để có thể unit test. Có thể sử dụng kết hợp với spring framework để inject dependency

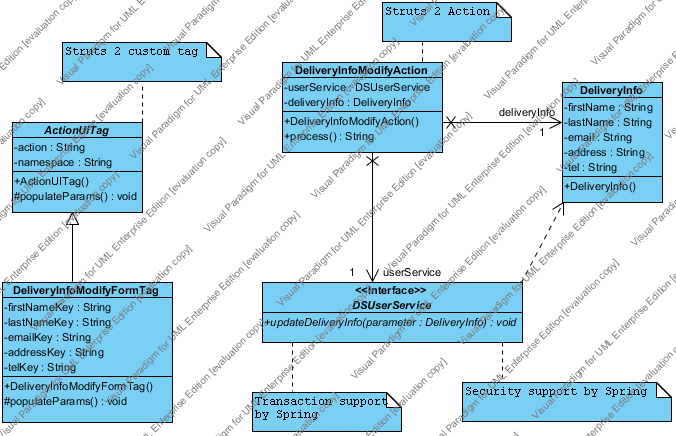
- Có thể thấy Action được spring (làm nhiệm vụ service locator) inject 1 đối tượng service, nhưng chỉ phụ thuộc ở dạng interface, không bị gắn với 1 lớp cụ thể, chính thiết kế này giúp Presentation tier có thể tách biệt 2 tầng riêng rẽ. hoạt động mà không có service thực sự 🡪 unit test

# ThayDoiThongTinCaNhan\_BL



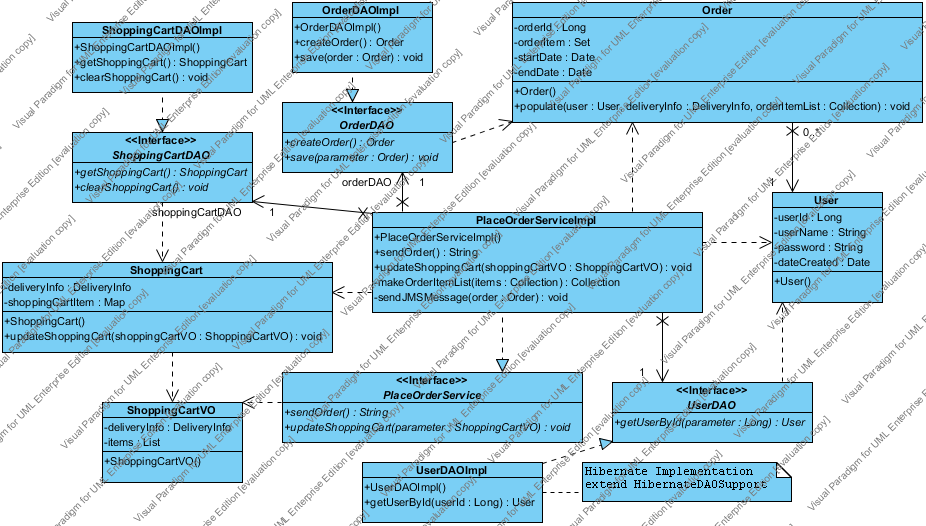
Class Diagram

# ThayDoiThongTinCaNhan\_P



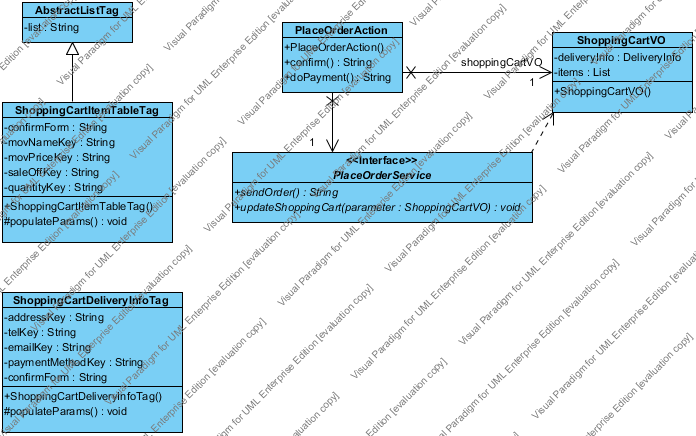
Class Diagram

# GuiDonHang\_BL



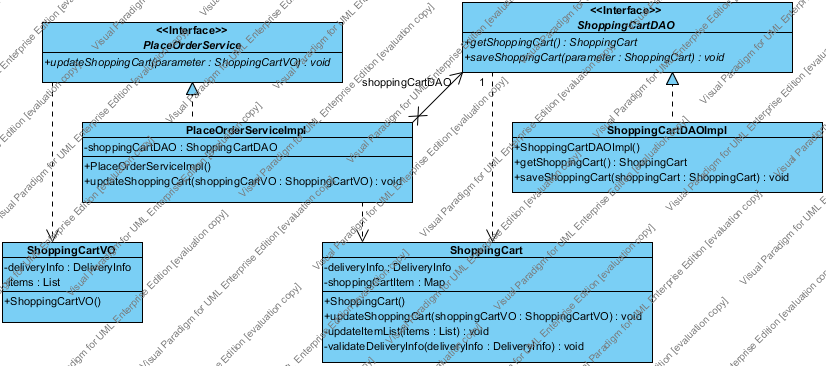
- Có thể thấy làm việc với ShoppingCart được lưu trong session không khác gì làm việc với các entity khác

# GuiDonHang\_P



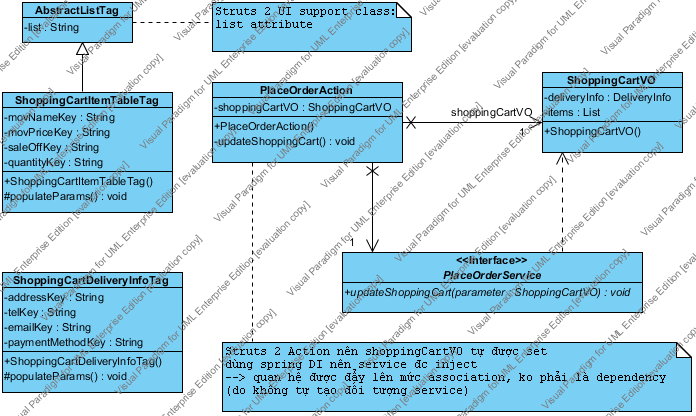
Class Diagram

# CapNhatGioHang\_BL



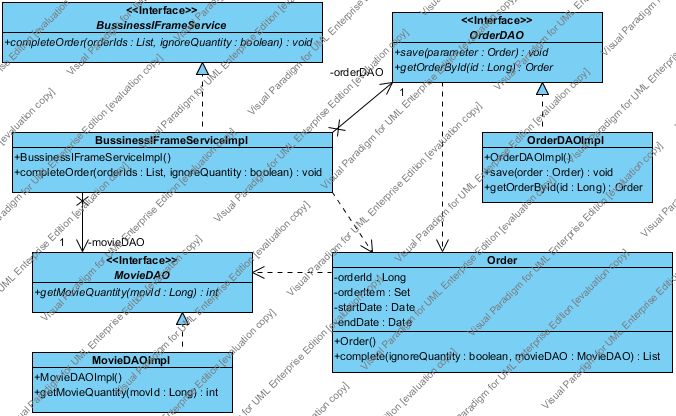
Class Diagram

# CapNhatGioHang\_P



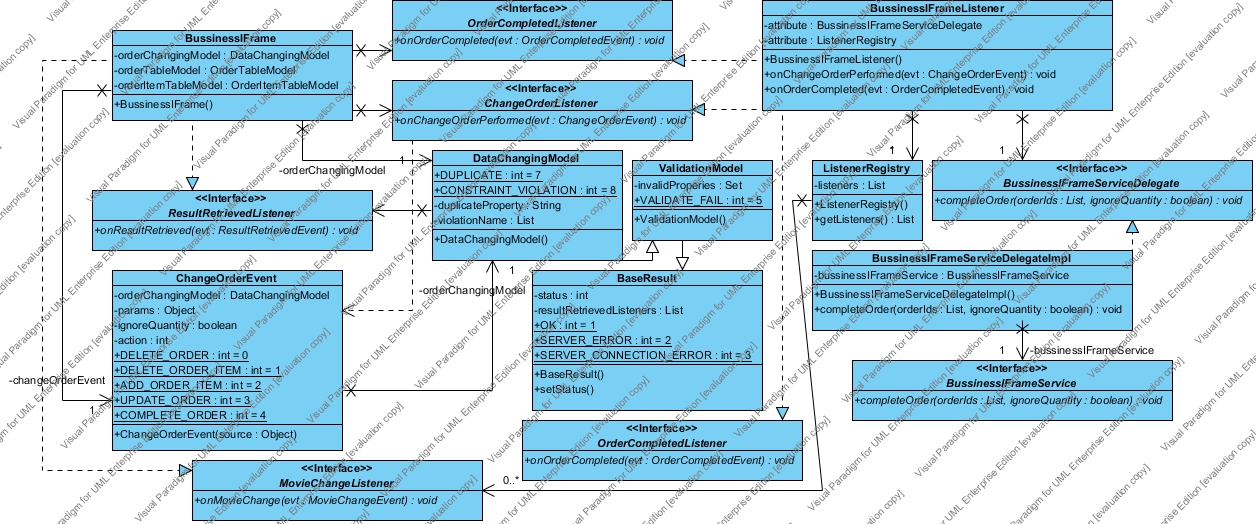
Class Diagram

# LuuPhieuXuat\_BL



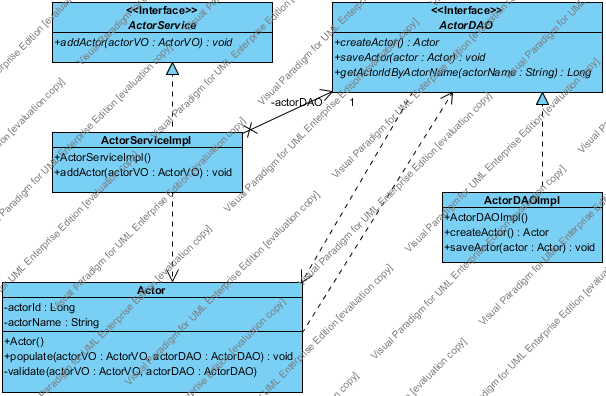
Class Diagram

# LuuPhieuXuat\_P



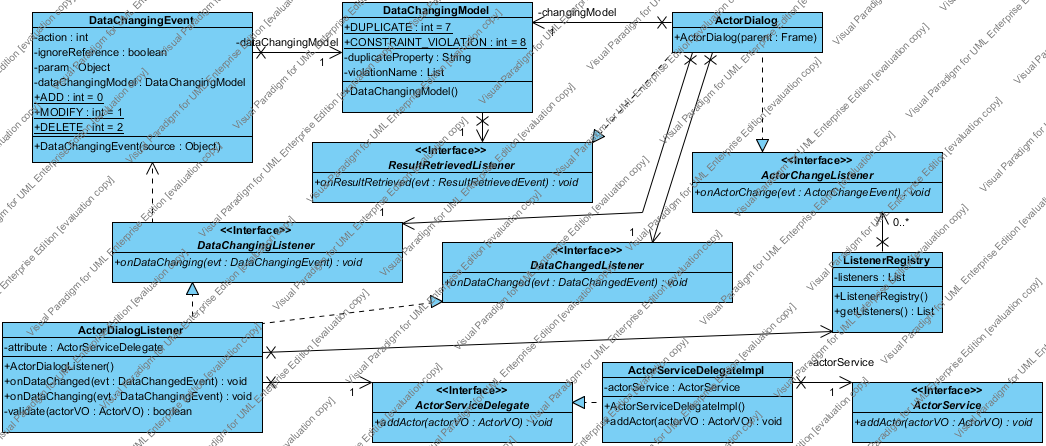
Class Diagram

# ThemDienVien\_BL



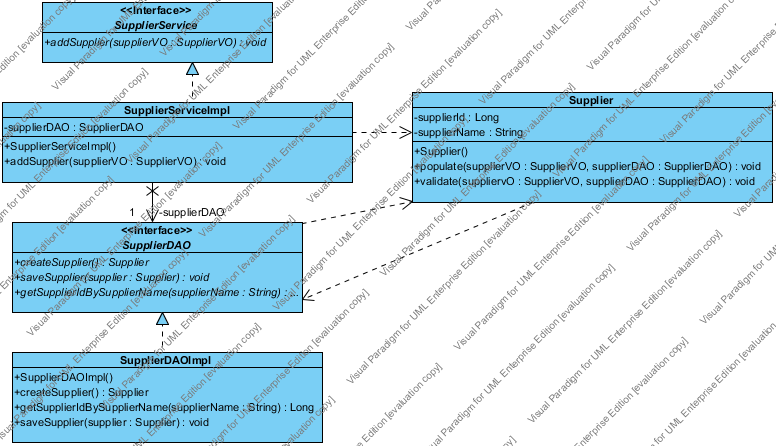
Class Diagram

# ThemDienVien\_P



Class Diagram

# ThemNhaCungCap\_BL



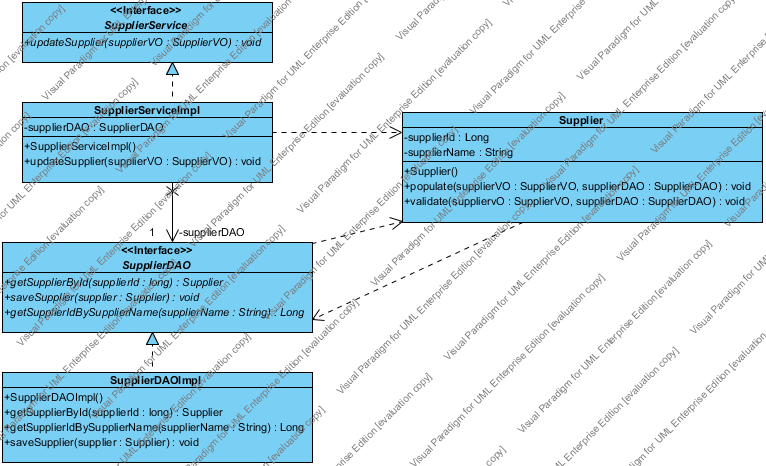
Class Diagram

# ThemNhaCungCap\_P



Class Diagram

# SuaNhaCungCap\_BL



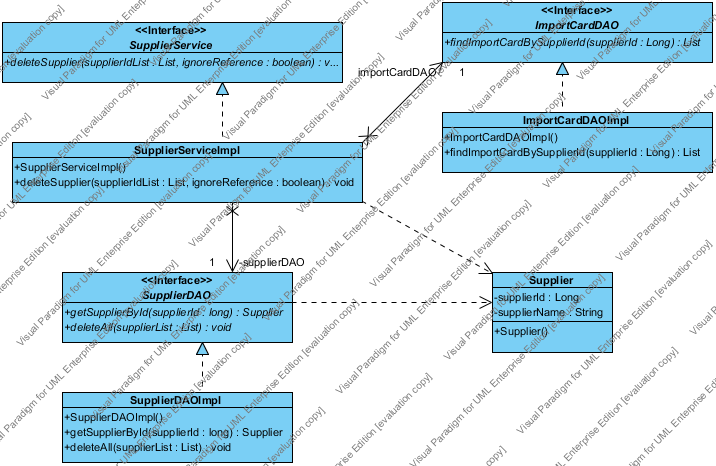
Class Diagram

# SuaNhaCungCap\_P



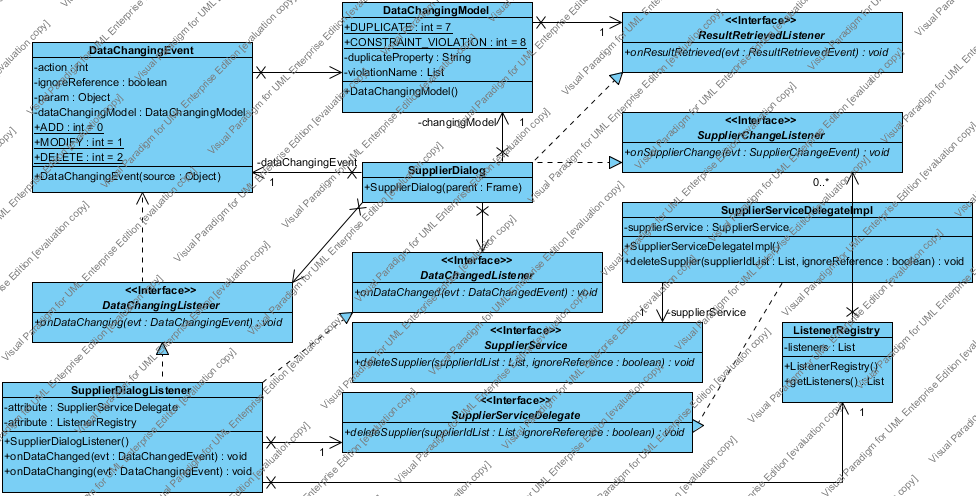
Class Diagram

# XoaNhaCungCap\_BL



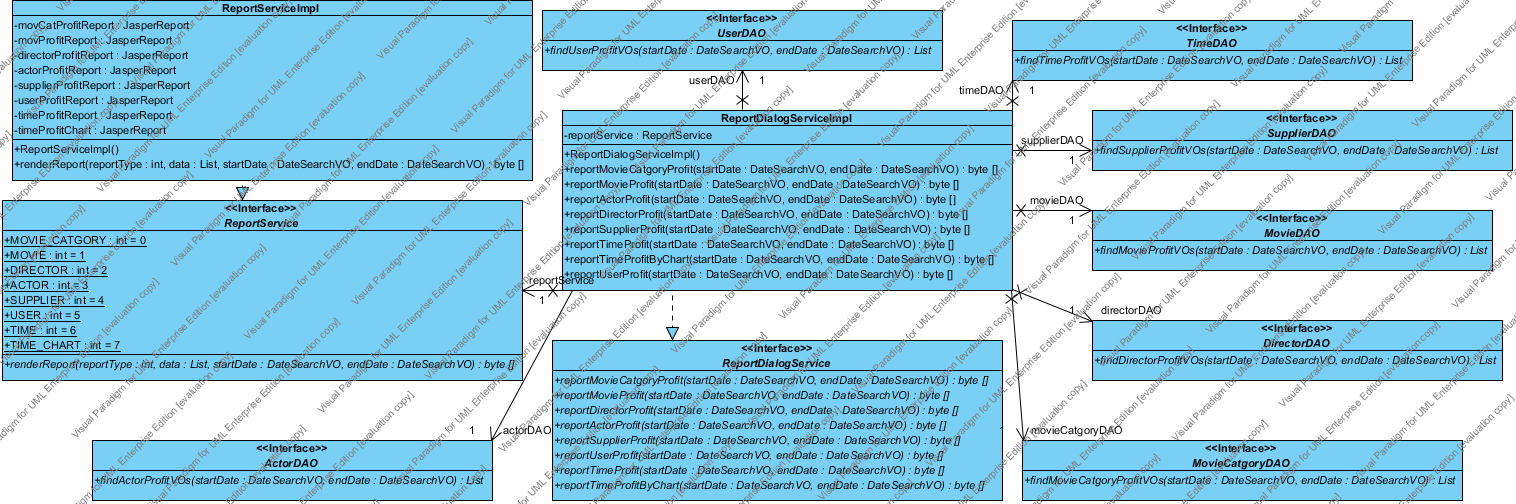
Class Diagram

# XoaNhaCungCap\_P



Class Diagram

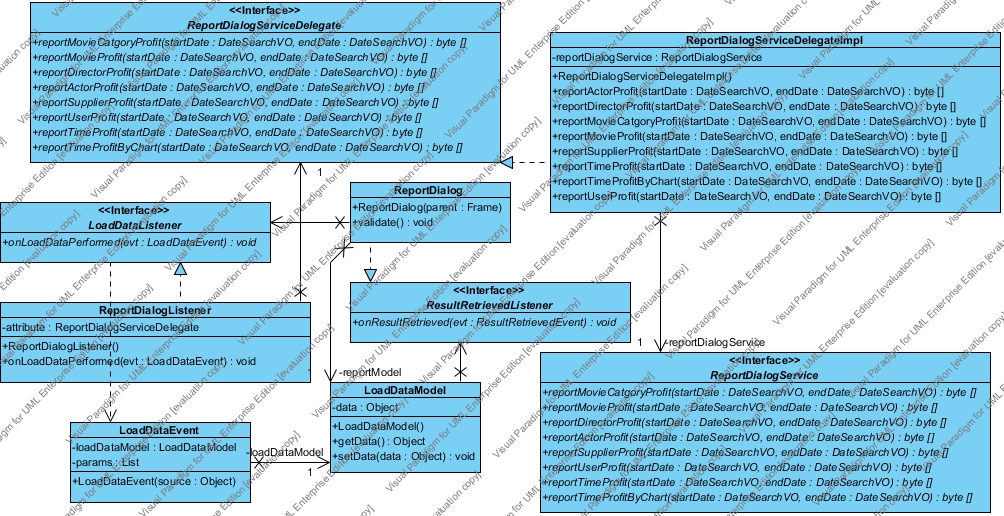
# XuatDuLieuBaoCaoDoanhThu\_BL



- Có thể thấy ưu điểm của thiết kế này: phần làm việc với dữ liệu , bussiness logic và phần gắn với Jasper Report được tách ra

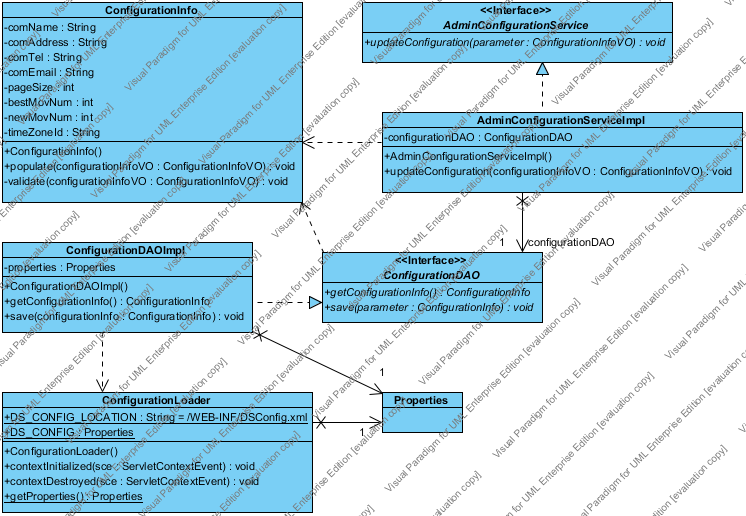
- Trên thực tế ReportService còn phục vụ mục đích không thống kê (in phiếu nhập, phiếu xuất).

# XuatDuLieuBaoCaoDoanhThu\_P



Class Diagram

# SuaCaiDatHeThong\_BL

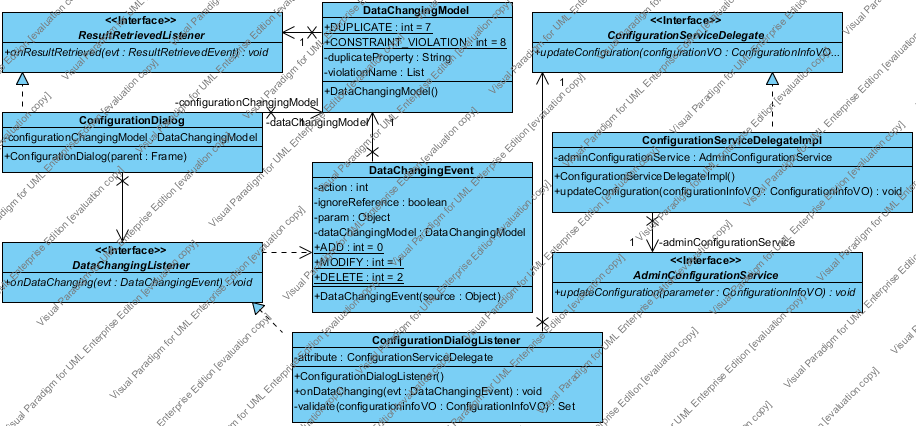


- Dễ thấy về cơ bản, cấu trúc trên không khác với các modul quản lý khác mặc dù configuration được lưu dưới dạng file xml, được load khi server start và được lưu lại dưới dạng file xml khi server stop. Điều này dễ dẫn đến việc gắn với Servlet api, tuy nhiên với thiết kế trên, các thao tác với servlet api được đóng gói trong ConfigurationLoader là 1 application listener. Nếu yêu cầu thay đổi, cần lưu configurationinfo trên csdl, hầu hết code được dùng lại

- ConfigurationInfo là 1 entity, bussiness logic được đóng gói trong này, khả năng dùng lại cao, xây dựng dễ

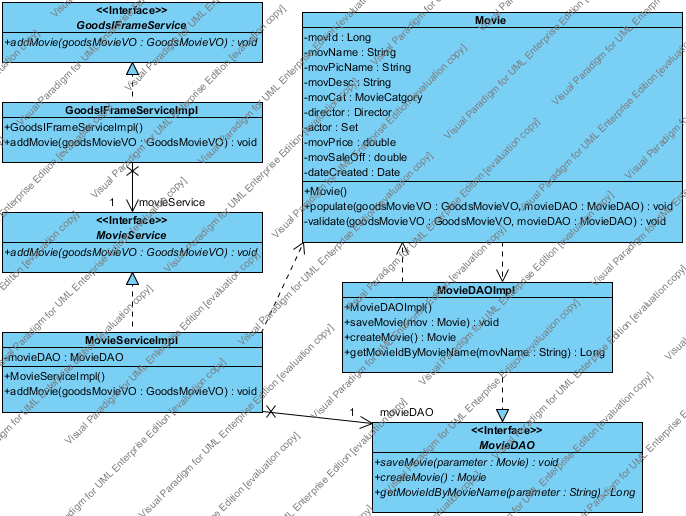
- Chú ý đây là enterprise application, phục vụ nhiều user, nhiều thread, configuration là dữ liệu chung cho nhiều thread vì thế cần threadsafe, vậy nên chọn lớp Properties, đây là lớp threadsafe, hơn nữa thông tin configuration là thông tin ít thay đổi mà thường chỉ đọc, vì thế việc lưu 1 đối tượng Properties dùng cho nhiều thread là có thể chấp nhận được về mặt tốc độ

# SuaCaiDatHeThong\_P



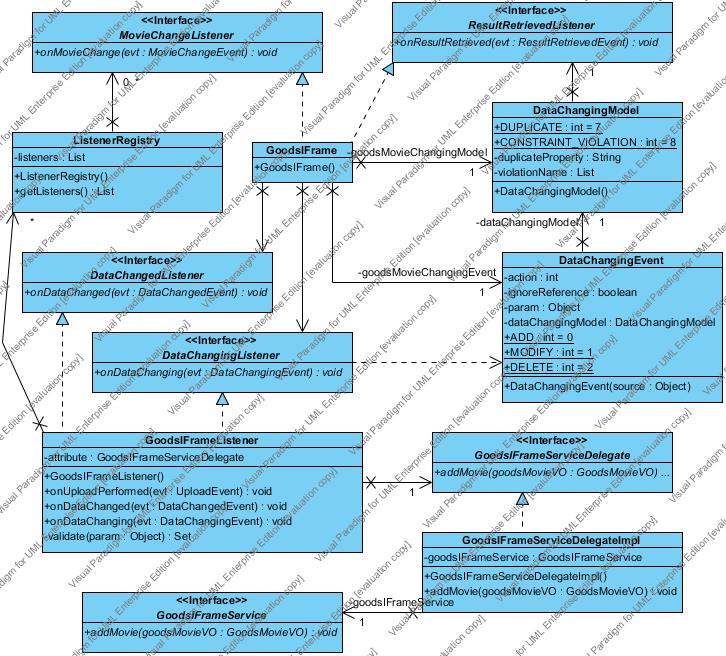
Class Diagram

# ThemPhim\_BL



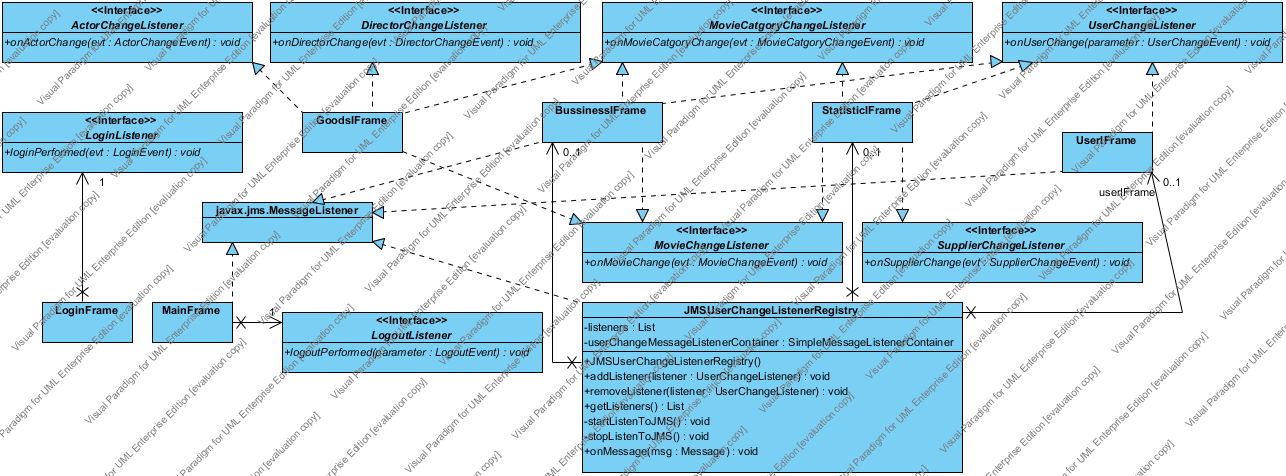
Class Diagram

# ThemPhim\_P



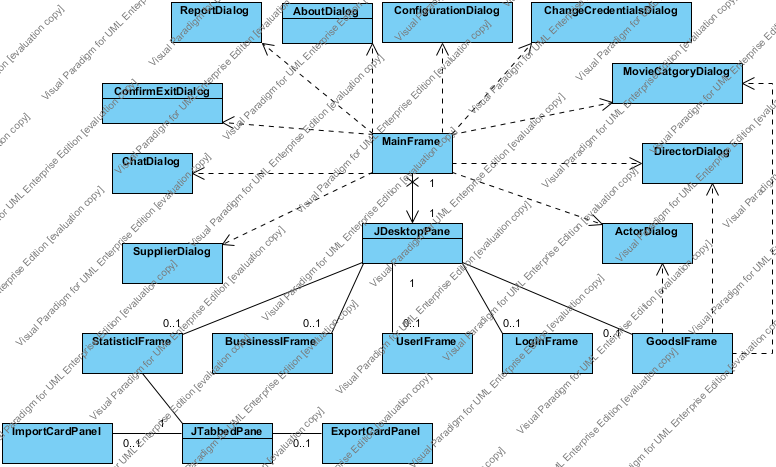
Class Diagram

# FrameAndListener



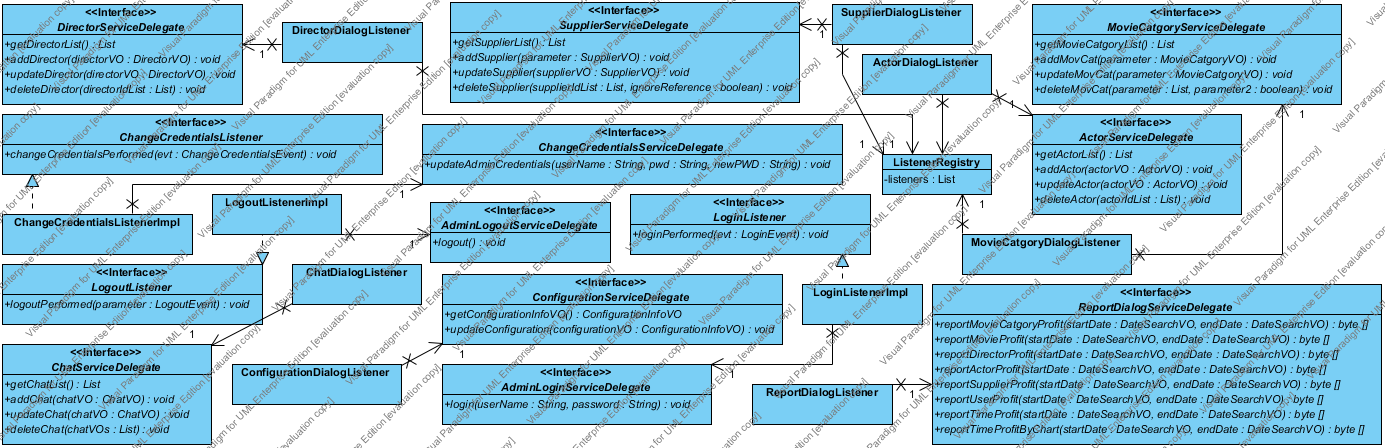
Class Diagram

# Frame



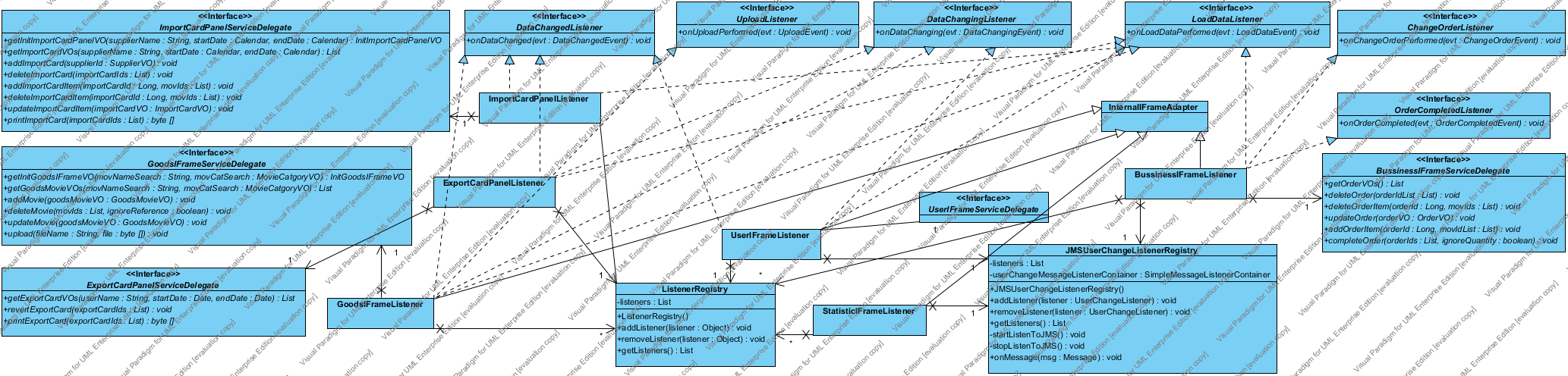
Class Diagram

# DialogListener



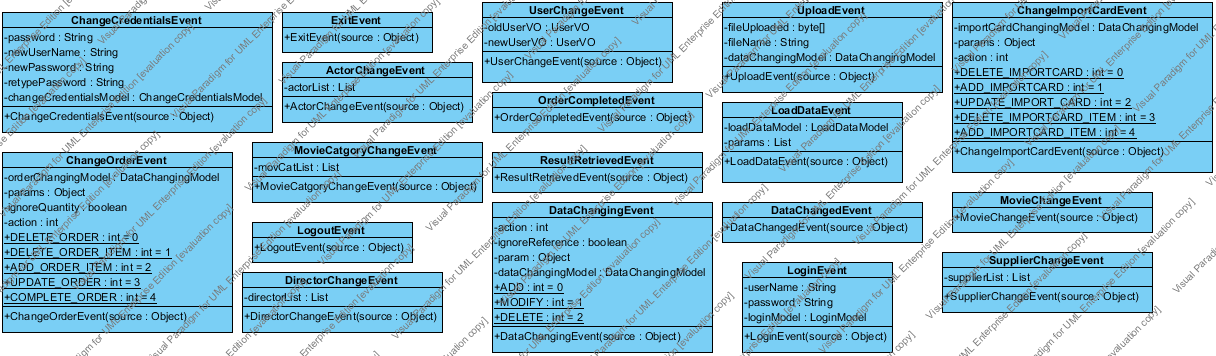
Class Diagram

# FrameListener



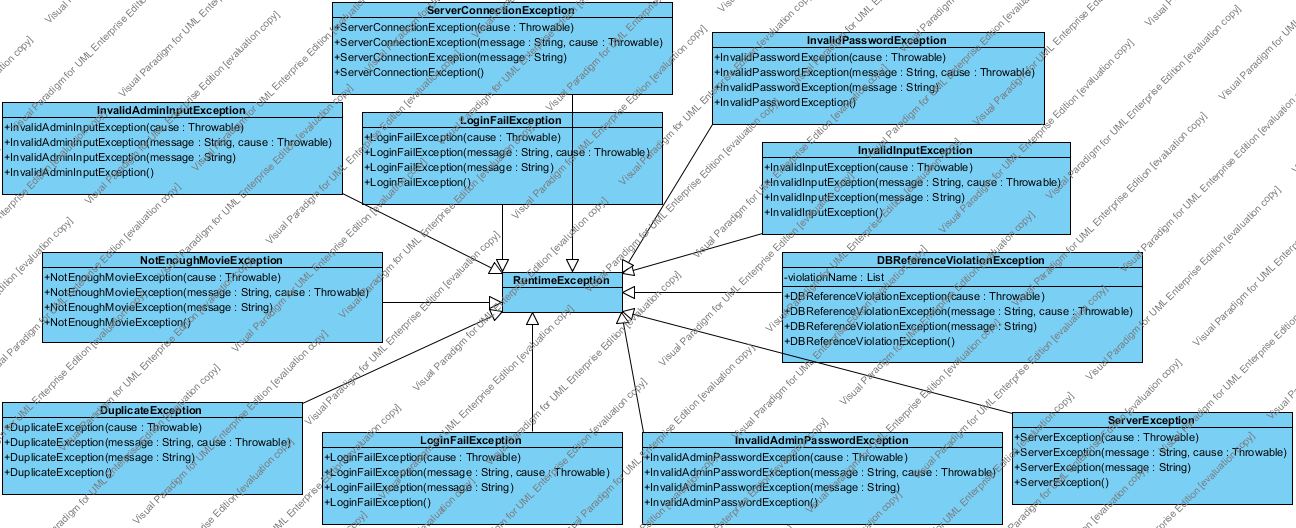
Class Diagram

# Event



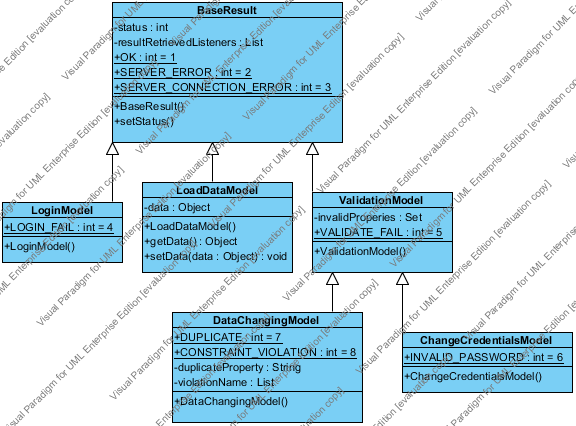
Class Diagram

# Exception



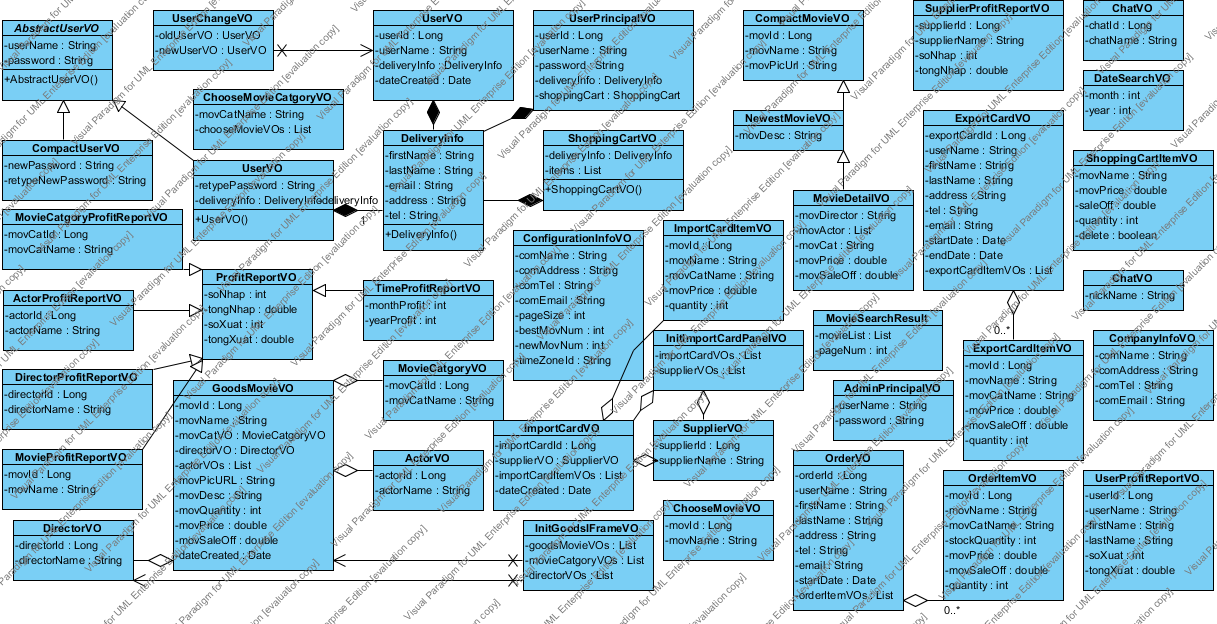
Class Diagram

# Model



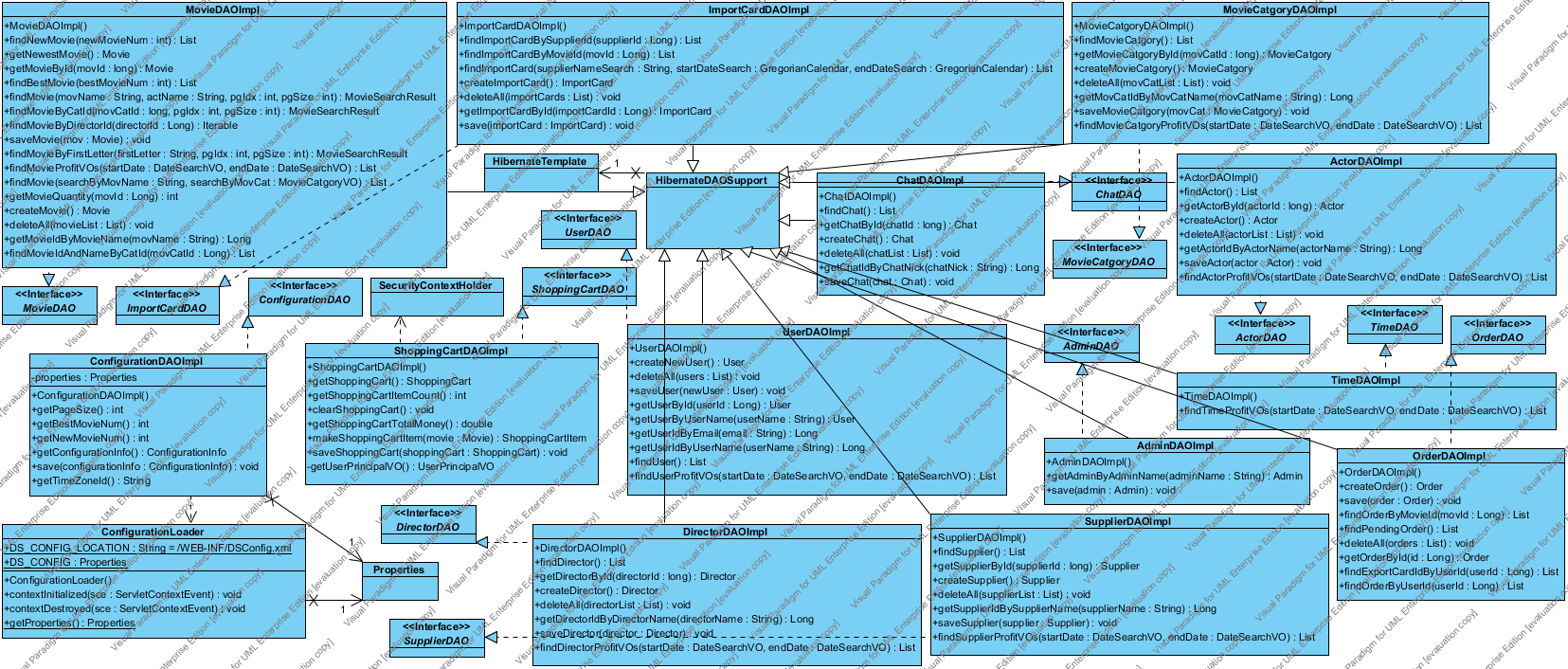
Class Diagram

# ValueObject



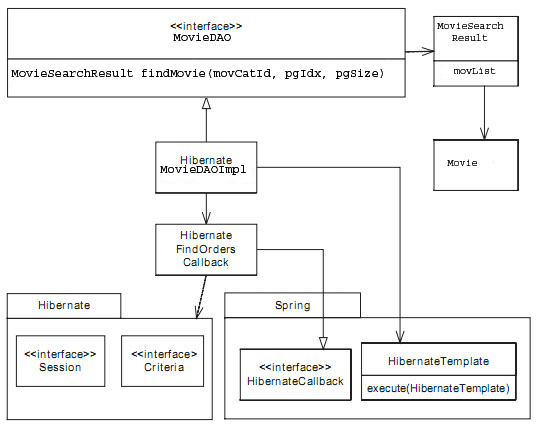
Class Diagram

# Dao

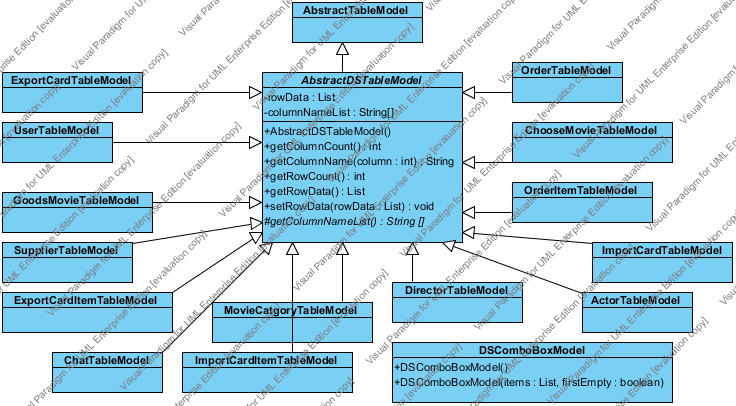


- Hầu hết các DAO Implementation đều extend HibernateDAOSupport, lớp này cung cấp thuộc tính hibernateTemplate 🡪 thực thi template design pattern, sử dụng callback method và abstract các thao tác lặp đi lặp lại

- Chỉ có ShoppingCartDAOImpl và ConfigurationDAOImpl là khác

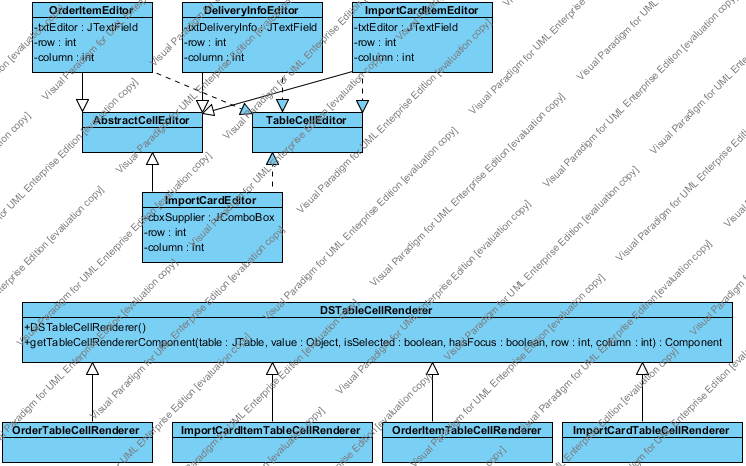


# TableModel



- AbstractDSTableModel là một lớp sử dụng generic programming, extend AbstractTableModel và implement hầu hết các lớp cần thiết, khi cần xây dựng table model cho một bảng cụ thể nào đó vd bảng Actor, chỉ việc extend AbstractDSTableModel<ActorVO> và gọi phương thức setData(List data)

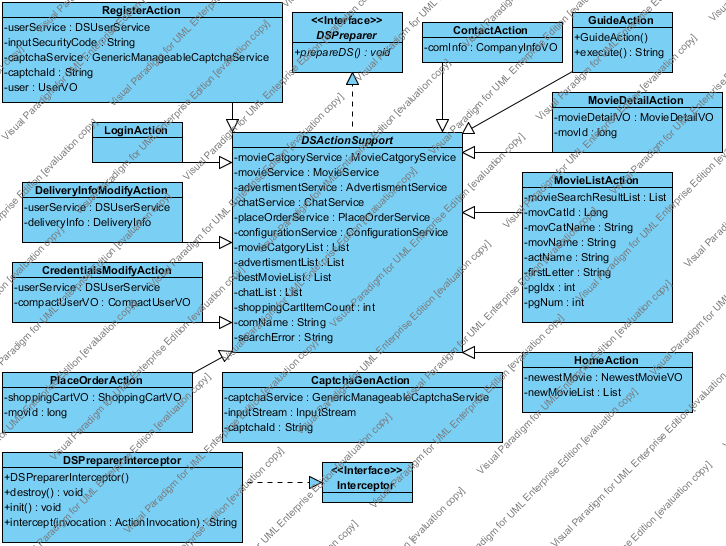
# TableCellRenderer\_Editor



- Các lớp AbstractCellEditor interface TableCellEditor là thuộc Swing

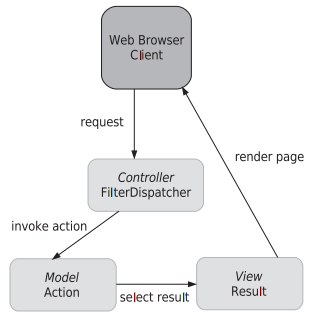
- DSTableCellRenderer thực thi render table cell cơ bản (Jlabel, và đánh dấu dòng), các bảng tùy vào yêu cầu khác nhau extend lớp này và override phương thức getTableCellRendererComponent để thay đổi việc render (in các cột đậm tùy theo yêu cầu từng bảng).

# StrutsAction

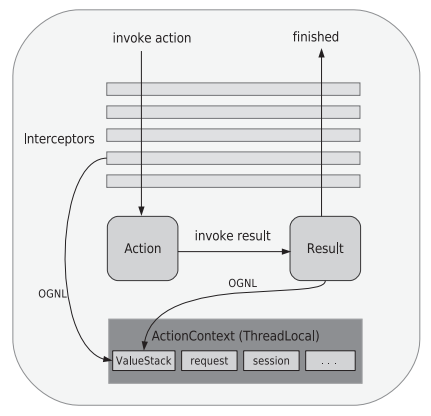


- Hầu hết các action đều extend DSActionSupport, lớp này chịu trách nhiệm lấy các dự liệu phụ (sidebar, header, footer), lớp này implement interface DSPreparer, interface này sử dụng kết hợp với DSPreparerInterceptor (sử dụng thêm bên cạnh các interceptor của struts 2) giúp cho biến action thực sự là 1 pojo. Thực chất interceptor hay filter cũng đều sử dụng khái niệm AOP giúp tách các chức năng và giúp developer tập trung vào bussiness logic, giúp action trở nên đơn giản hơn.

- DSPreparerInterceptor implement Interceptor Interface là Struts2 interface



- Khi nhận request, framework sẽ kiểm phân tích các file struts.xml 🡪 xác định Action nào sẽ execute, tạo new instance ActionInvocation 🡪 reference tới Action instance, phân tích xml 🡪 reference Inteceptor chain, request object, result list



- ExceptionInterceptor: đây là interceptor ở đầu stack, exception khi xuất hiện ở bất cứ đâu, interceptor này sẽ catch lại, đưa lên ExceptionHolder và đưa lên ValueStack, sau đó map result theo file config. Thường là 1 trang jsp báo lỗi, trang này có thể access exception qua ExceptionHolder

- ServletConfigInterceptor: đây là interceptor chịu trách nhiệm inject các đối tượng thuộc servlet api: session context, application context, request context, parameters, nhưng đều ở dạng Map cho các Action, điều này giúp tránh sự phụ thuộc của Action vào servlet api

- I18N: đây là interceptor chịu trách nhiệm nhận 1 tham số request\_locale và đưa locale lên session, action có thể lấy locale từ session, nếu không có tham số trên, sẽ đọc trong request header

- PrepareInterceptor: đây là interceptor sẽ gọi phương thức prepare() của Interface Preparable. Action sẽ cần implement interface này nếu muốn Interceptor gọi prepare(), phương thức này thường được dùng để chuẩn bị dữ liệu cho form (prepopulate data) và cần được đặt trước WorkflowInterceptor để dù dữ liệu đưa lên không hợp lệ, form vẫn được prepare (vd: hộp chọn cần có 1 danh sách dữ liệu trước).

- ModelDrivenInterceptor: đây là interceptor cho phép tách model ra khỏi action, sử dụng kết hợp với ModelDriven Interface, ModelDrivenInterceptor sẽ gọi phương thức getModel() nếu action implement ModelDriven Interface, sau đó đưa model lên ValueStack, view có thể làm việc với Model này

- ParamsInterceptor: đây là interceptor sẽ làm việc với HttpServletRequest, lấy ra parameters và inject cho Model (là các properties của action với tên tương ứng với parameter name hoặc properites của model object trên value stack nếu ModelDriven. Thực chất các properties của model object đã che đi properties của action object).

- DsPreparerInterceptor: đây là interceptor giúp chuẩn bị dữ liệu cho DVDWeb, hoạt động tương tự PrepareInterceptor

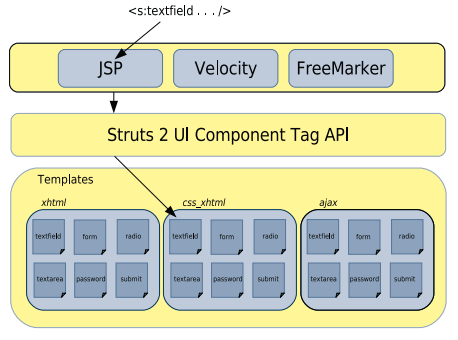
- ConversionErrorInterceptor: đây là interceptor đặt sau ParamsInterceptor giúp chuyển conversion error trong ActionContext và chuyển thành fieldError trong Action object (action object implement ValidationAware).

- ValidationInterceptor: đây là interceptor cho phép validation framework hoạt động, framework này hoạt động dựa trên xml config, thực tế trong project này, các validation definition được phân về cho từng value object, các error message, label trên web được xây dựng dưới dạng file properties (ResourceBundle) , thiết kế này giúp dùng lại code và dễ quả n lý hơn, do chia nhỏ ra và gắn trực tiếp với các đối tượng được định nghĩa. VD: movName=Tên phim. Hay DeliveryInfo là value object được sử dụng lại ở nhiều nơi: trang đăng ký, trang giỏ hàng, việc validate không cần phải định nghĩa lại cho từng trang

- WorkFlowInterceptor: đây là interceptor sẽ gọi phương thức validate() nếu action implement Validateable. Phương thức này khi được action thực hiện, nếu có error sẽ addActionError() hoặc addFieldError() (là các phương thức của ValidationAware interface) . WorkFlowInterceptor sau đó sẽ kiểm tra nếu action có action error hoặc field error sẽ không invoke action method nữa mà sẽ trả về ngay input result map.

- TokenSessionInterceptor: đây là interceptor giúp ngăn chặn tình trạng người dùng click nhiều lần vào 1 link khiến server load nặng, đặc biệt với các thao tác tìm kiếm nhiều dữ liệu. Interceptor này sinh ra 1 token, token này sẽ được đặt vào form dưới dạng hidden field hoặc đặt vào link dưới dạng query string, khi người dùng click vào link hoặc submit form lần đầu, interceptor này sẽ so sánh token trong param và token trong session hiện tại, nếu trùng, cho phép method của action được gọi, đồng thời sinh 1 token mới, trong quá trình server đang load (chưa return result) nếu người dùng submit tiếp, 1 request mới với token cũ sẽ được gửi lên server. Interceptor lúc này so sánh sẽ thấy token không còn trùng nữa lúc này không cho phép action method được gọi mà sẽ chờ đợi thread trước thực hiện (result này khi thực hiện xong sẽ lưu vào session) và lấy result này trả về cho người dùng

- ExecAndWaitInterceptor: đây là interceptor cũng giúp giảm tải cho server: Khi người dùng thực hiện 1 request có thể khiến server load mất nhiều thời gian, lập tức interceptor này (được declare cho 1 action cụ thể nào đó) sẽ trả về map tới trang wait, và đồng thời mở 1 thread khác gọi phương thức của action method. Trang wait được cài đặt trong vòng 2s sẽ tự động request lại, interceptor căn cứ kết quả thực hiện action method trên session, nếu đã thực hiện xong 🡪 trả về kết quả, nếu chưa thực hiện xong, tiếp tục trả về trang wait

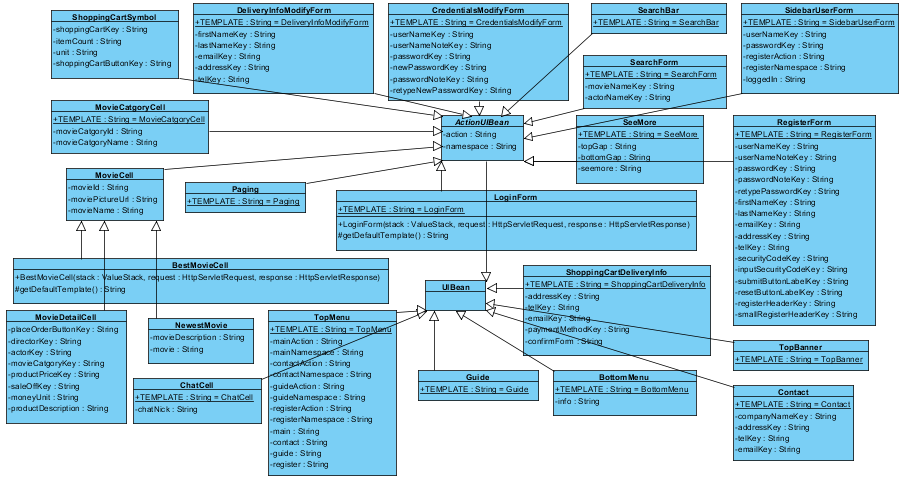


- Templates là các file \*.ftl sử dụng Freemaker để render html, đây là 1 framework khá mạnh, có thể render bất kỳ dạng text nào, gồm cả xml, html... Hiện đang được sử dụng ngày càng rộng dãi vì dễ dùng và render nhanh, tốt hơn jsp

- Việc xây dựng các thành phần của trang web thành các custom tag giúp tăng tính linh hoạt và dùng lại code, giúp tận dụng tính năng theme của Struts 2 là tính năng rất mới: khi cần thay đổi 1 thành phần nào đó, chỉ cần thay đổi 1 file flt và đóng gói thành jar (xây dựng theme mới) và thay đổi cài đặt hệ thống (tên theme), hoặc thậm chí dễ hơn có thể chỉ cần sửa file css.

- Dễ dàng thay đổi như vậy là do theme có tính năng thừa kế, vì thế những gì cần thay đổi chỉ cần overidde lại, bên cạnh đó có thể thấy những gì hardcode hoàn toàn không cần thay đổi (jsp chỉ làm nhiệm vụ layout)

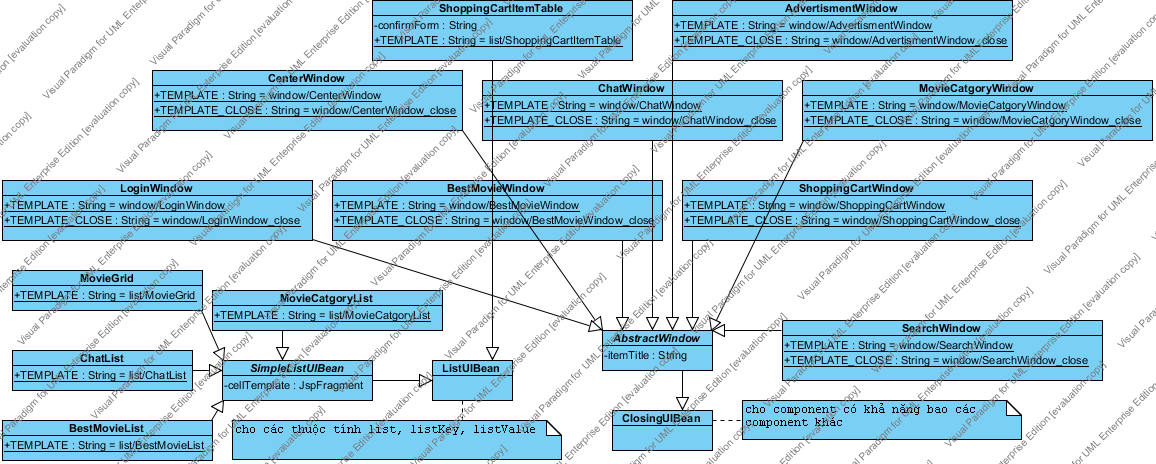
# Struts2Component



- Các lớp đều có thuộc tính TEMPLATE, chính là đường dẫn đến file \*.ftl 🡪 đây là các lớp làm việc với Freemarker api, để render ra html.

- Các lớp này extend ActionUIBean, lớp này cung cấp 2 thuộc tính dùng lại khá nhiều là action và namespace. ActionUIBean extend UIBean, lớp này cung cấp các thuộc tính dùng chung như css, id, theme, onclick, onFocus, disable, tabIndex... là các thuộc tính mà 1 thẻ html thường có. Đây là lớp của Struts 2

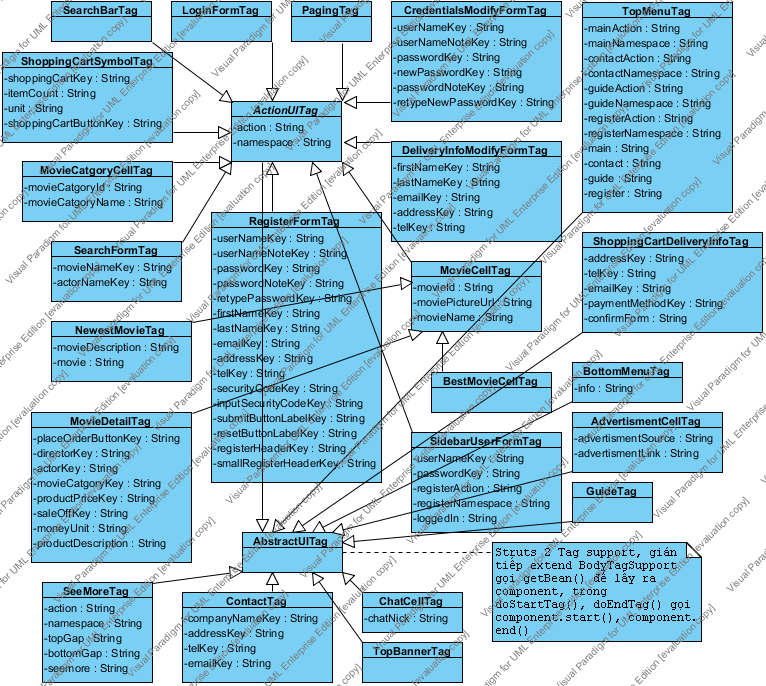
# Struts2Component\_List



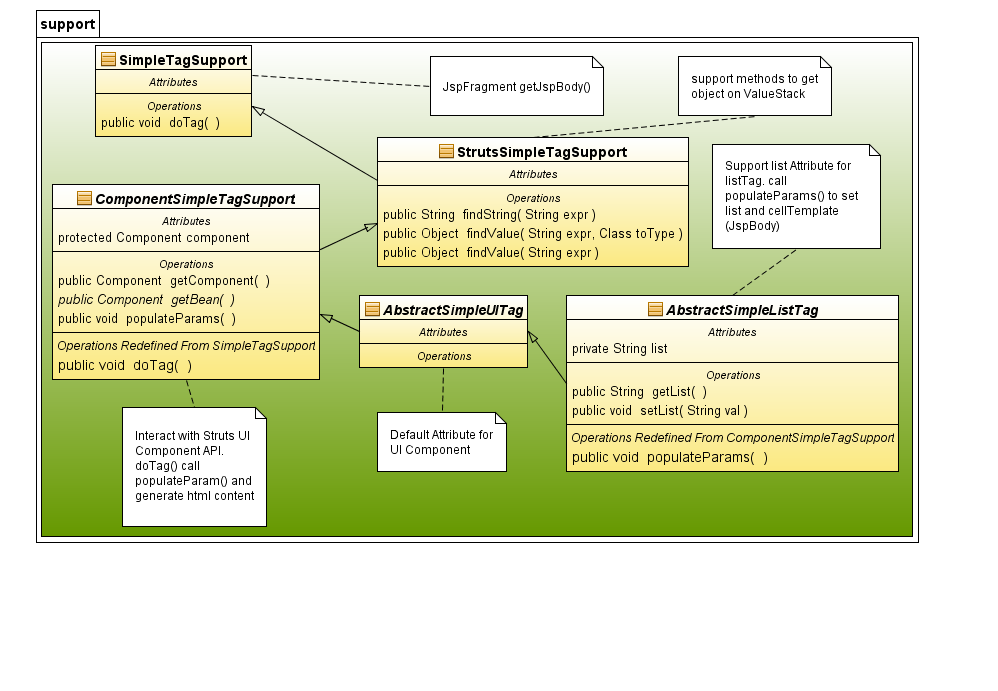
- Đây là các lớp sinh các component đặc biệt: window: chứa các tag khác, list: chứa và lặp các tag khác

- Chú ý lớp SimpleListUIBean: cho thuộc tính JspFragment là thuộc tính cho phép list component sử dụng để render html lặp lại kết quả render tag con (template)

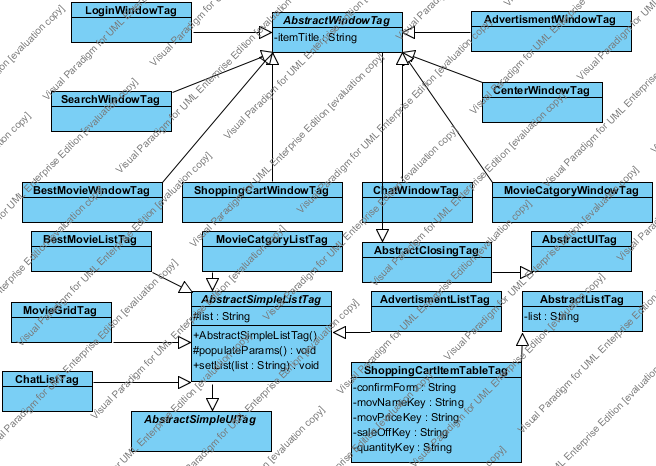
# Struts2Tag



- Các custom tag không trực tiếp làm việc với Freemarker api mà qua struts 2 component api. Các custom tag có nhiệm vụ chuyển lại giá trị các attribute của customtag cho các component tương ứng với custom tag đó



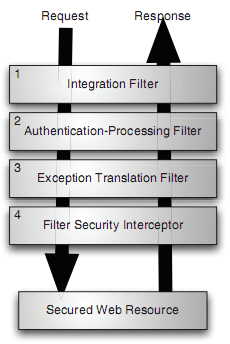
# Struts2Tag\_List



- Security:

AOP 🡪 giúp các thành phần cần security không phải quan tâm đến security một cách trực tiếp. Mặc dù có thể hard code security tuy nhiên best practice là tách những vấn đề không phải là domain problem riêng.

Sử dụng Spring AOP và Spring Security Filter cho chúng ta khả năng declarative security (config qua xml file)



- ConcurrentSessionFilter: đây là filter chịu trách nhiệm kiểm tra sessionId trong SessionRegistry, nếu sessionID này đã invalid sẽ thực hiện logout qua LogoutHandler. Đồng thời chuyển request đến 1 trang error. Filter này sử dụng kết hợp với:

HttpSessionEventPublisher là session listener được khai báo trong web.xml. Listener này chịu trách nhiệm cập nhật SessionRegistry sau mỗi sự kiện tạo và hủy 1 session.

ConcurrentSessionController : lớp này chịu trách nhiệm cập nhật SessionRegistry mỗi khi user đăng nhập thành công, và thông báo SessionRegistry invalidate các session cũ.

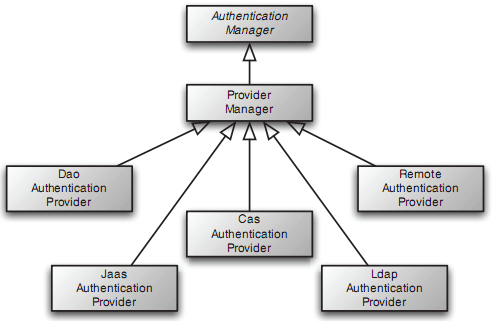
- HttpSessionContextIntegrationFilter: lớp này chịu trách nhiệm lấy ra trong session context SecurityContextHolder. Và các lớp khác có thể truy nhập thông tin SecurityContext qua lớp này (ThreadLocal), thay đổi thông tin trên SecurityContext, khi có response, Lớp này chịu trách nhiệm cập nhật thông tin từ SecurityContext vào session attribute. Thiết kế này giúp các thành phần còn lại của application không phải gắn với servlet api.

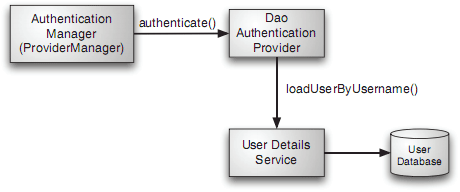
- AuthenticationProcessingFilter: đây là filter chịu trách nhiệm lấy credentials từ trong request và gọi phương thức authenticate() của lớp AuthenticationManager. Nếu đăng nhập thành công, sẽ dispatch request sang url đang bị intercept hoặc default url nếu user request login page. Nếu không đăng nhập được sẽ dispatch request sang error page.

- AuthenticationManager là abstract class cung cấp phương thức authenticate().

- ProviderManager không tự thực hiện đăng nhập mà chứa 1 list các provider khác

- DaoAuthenticationManager: được inject UserDetailService (DSUserServiceImpl) đồng thời được inject MD5PasswordEncoder. Gọi phương thức loadUserByUserName() để có đối tượng UserDetail. Thực hiện kiểm tra userName, và password

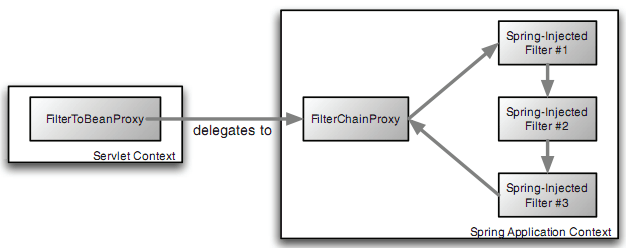




- LogoutFilter: kiểm tra nếu request url có dạng “/j\_spring\_security\_logout” thực hiện logout qua 1 đối tượng của lớp SecurityContextLogoutHandler (được inject nhờ spring). Sau đó sẽ dispatch request về trang chủ

- ExceptionTranslationFilter: đây là filter chịu trách nhiệm catch các exception từ filter security filter nếu có (AuthenticationException và AccessDeniedException) và map tới các url tương ứng

- FilterSecurityInterceptor: filter này chứa các khai báo về url pattern và các role permission. Tuy nhiên filter này không trực tiếp quyết định request tới secure resource có được chấp nhận hay không mà sử dụng AccessDecisionManager (được inject) để quyết định

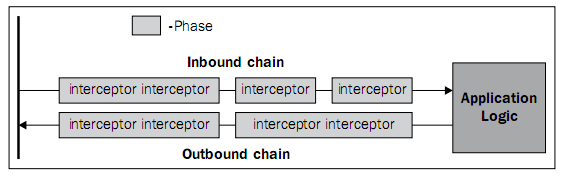


- FilterToBeanProxy: khai báo trong web.xml

- Đối với WebService: sẽ có chút khác biệt do credential không ở trong request parameters mà nằm trong soap message

- Sử dụng lớp AdminLoginService đây là 1 Webservice được inject AuthenticationManager và gọi authenticate() tương tự AuthenticationProccessingFilter. Đồng thời chịu trách nhiệm notify SessionRegistry và send kick message

- Webservice:



- Inbound phase:

UNMARSHAL: chịu trách nhiệm Unmarshalling

(PRE/USER/POST)\_LOGICAL: chịu trách nhiệm xử lý thêm unmarshalled request

PRE\_INVOKE: Trước khi gọi web method của webservice

- Outbound phase:

(PRE/USER/POST)\_LOGICAL: Xử lý đối tượng được marshall

PREPARE\_SEND: Mở kết nối tới server

MARSHAL: Thực hiện marshall

SEND: Gửi message và đóng stream

- Để đảm bảo session cho webservice, cần thêm 2 interceptor

- GetSessionIdInterceptor và PutSessionIdInterceptor: đây là 2 interceptor ở inbound phase và outbound phase, chịu trách nhiệm đọc và ghi dữ liệu lên HttpServletRequest (sesssionId)

- Có thể nói đây là cách không theo chuẩn, nhưng nó cho thấy khả năng linh hoạt mà mạnh mẽ của CXF webservice

- JMS:

- ActiveMQConnectionFactory: make connection

- JmsTemplate: được inject ActiveMQConnectionFactory template design pattern, nếu sử dụng jms api chuẩn, sẽ phải lặp lại nhiều lần các bước: make connection, make session, make message, send, receive, close connection...