**CHƯƠNG 4: CÔNG NGHỆ ỨNG DỤNG VÀ CÀI ĐẶT**

**4.1. Một số công nghệ ứng dụng trong hệ thống:**

**4.1.1. Spring Framework 2.5:**

- Đây là framework giúp DI (Dependency Injection hay IOC). AOP (Aspect Orient Programming). Giúp viết các POJO chứa Bussiness logic, có thể unit Test, sử dụng kết hợp với JMOCK

- Hỗ trợ tốt Hibernate : Hỗ trợ quản lý transaction thông qua AOP, hỗ trợ Template Design Pattern giúp viết các class DAO dễ dàng và hiệu quả

- Hỗ trợ Struts 2 : Inject service object đối với các struts Action

- Hỗ trợ JMS : Template design patern và declaration deployment đối với hầu hết các đối tượng quan trọng (SessionFactory, MessageListener, Queue, Topic). Từ đó có thể Inject vào các Service Object 🡪 Giúp Unit Test cũng như quản lý các đối tượng hiệu quả hơn (các đối tượng được khai báo trong spring context sẽ được spring quản lý như 1 bean container)

- Hỗ trợ CXF Webservice (Declaration Deployment). Giúp đơn giản hóa quá trình xây dựng hệ thống

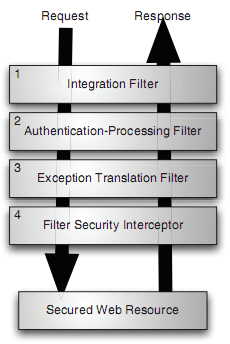
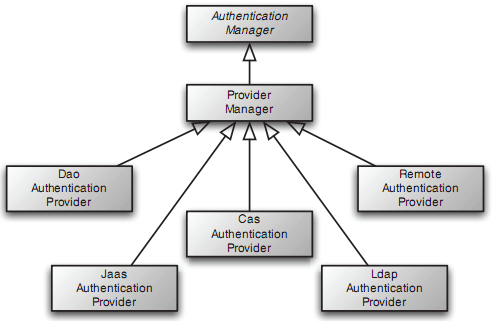
**4.1.2. Spring Security Filter :**

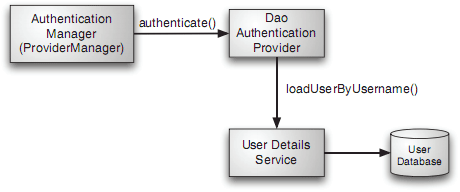
- Đây là hệ thống Security được phát triển từ project Acegi

- Là hệ thống tốt, được xây dựng và test kỹ, làm việc thông qua các declaration (config các file XML), sử dụng khái niệm AOP, giúp tập trung hơn vào việc xây dựng bussiness logic 🡪

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().

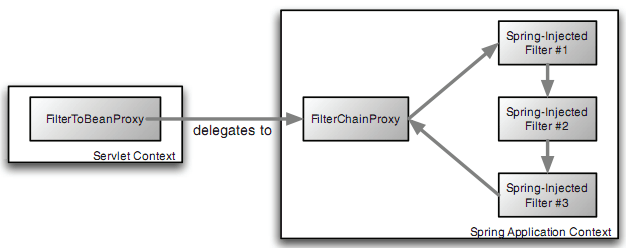
- ProviderManager không tự thực hiện đăng nhập mà chứa 1 list các provider khác

- 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.

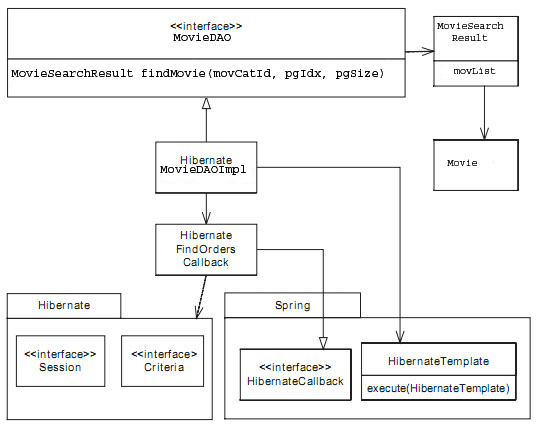
**4.1.3. Hibernate Framework:**

- Object Relation Mapping , giúp kết nối, thực hiện thao tác đối với CSDL nhanh dễ dàng

- Giúp quản lý connection (pooling), transaction, catch, quản lý entity, tự sinh chuỗi SQL thông qua các file mapping giữa các POJO và các bảng trong CSDL quan hệ.

- 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.



**4.1.4. Struts 2:**

- MVC Web Framework, cải tiến từ Struts 1, bỏ Action Form Bean, thêm khái niệm Value Stack, không sử dụng Servlet API mà dùng Thread Local, giúp xây dựng các Action Class thực sự là các POJO

- Sử dụng khái niệm AOP, xây dựng các Interceptor. Giúp cho việc xây dựng POJO thực sự tập trung vào bussiness logic. Và thực hiện các thao tác đối với Servlet API, như vậy POJO không bị phụ thuộc vào Framework cũng như Servlet API, tăng khả năng dùng lại và tạo điều kiện dễ dàng cho Unit Test

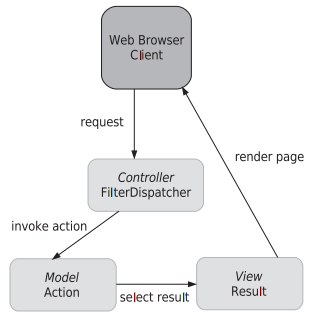
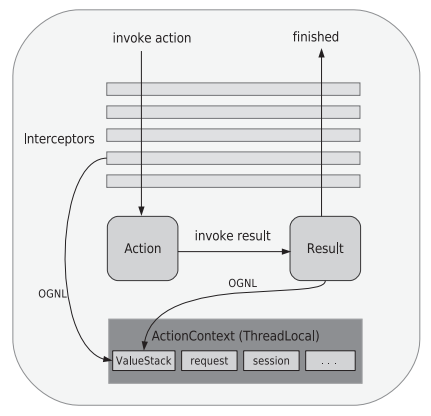
- Hỗ trợ nhiều Result type khác nhau : Velocity, Freemarker, JSP, Jasper Report…

- Hệ thống thẻ custom Tag mới, đặc biệt hỗ trợ xây dựng hệ thống Custom Tag mạnh, và hỗ trợ nhiều ngôn ngữ đánh dấu : Freemarker, Velocity, JSP. Giúp xây dựng hệ thống thư viện Component cho web ,

- Tổ chức thành dạng Theme có hỗ trợ kế thừa, vì thế có thể dễ dàng xây dựng theme mới từ theme đã có sẵn (Ghi đè các Component cần sửa).

- 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 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

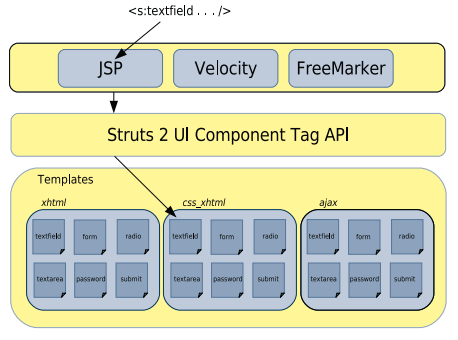
- ConversionErrorInterceptor: đây là interceptor đặt sau ParamsInterceptor giúp chuyển conversion error trong ActionContext và chuyển thành fieldError trong Action object (action object implement ValidationAware).

- 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.



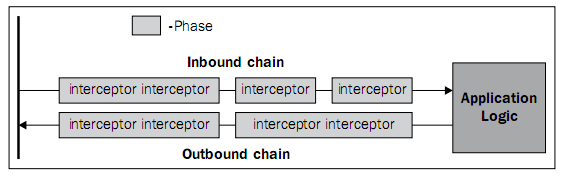
- Templates là các file \*.ftl sử dụng Freemaker để render html, đây là 1 framework khá mạnh, có thể render bất kỳ dạng text nào, gồm cả xml, html... Hiện đang được sử dụng ngày càng rộng dãi vì dễ dùng và render nhanh, tốt hơn jsp

- 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).

**4.1.5. CXF WebService, ActiveMQ JMS:**

- 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...

Ngoài ra em còn sử dụng thêm Tiles Framework: là framework giúp layout trang web , Log4j 🡪 tạo log, giúp quá trình phát triển, debug nhanh, tránh được các câu lệnh System.out.print()

- JMock, JUnit 🡪 Test Development Driven, giúp unit test các POJO và các service class, giúp phát triển hệ thống nhanh hơn do có thể phát triển từng thành phần riêng rẽ (test chạy riêng rẽ mà không cần xây dựng các dependency object, đặc biệt các đối tượng DAO là các đối tượng làm việc với CSDL).

**4.2. Hướng dẫn cài đặt:**

**4.2.1. Môi trường triển khai**

**4.2.1.1. Phần cứng tối thiểu :**

**Server :**

|  |  |
| --- | --- |
| Minimum | Preferred |
| PC Pentium IV 3 GHz | Intel Core 2 Dual |
| 1 Gb Ram | 4 Gb Ram |
| 100 Gb HDD | 500 Gb HDD |

**Client :**

|  |  |
| --- | --- |
| Minimum | Preferred |
| PC Pentium 4 1.8 GHz | PC Pentium 4 2.8 GHz |
| 512 Mb Ram | 1 Gb Ram |
| 80 Gb HDD | 160 Gb HDD |

**4.2.1.2. Phần mềm :**

**Server :**

- .Net Framework 3.5

- Sql Server 2005

- Active MQ

- TomCat 6.20

- JRE 6 u18

**Client :**

- Java Runtime Enviroment (JRE) 6 update 18

**4.2.1.3. Công cụ để phát triển**

- JDK 1.6.18

- NetBeans 6.8

- SQL Server 2005

- Jasper Report

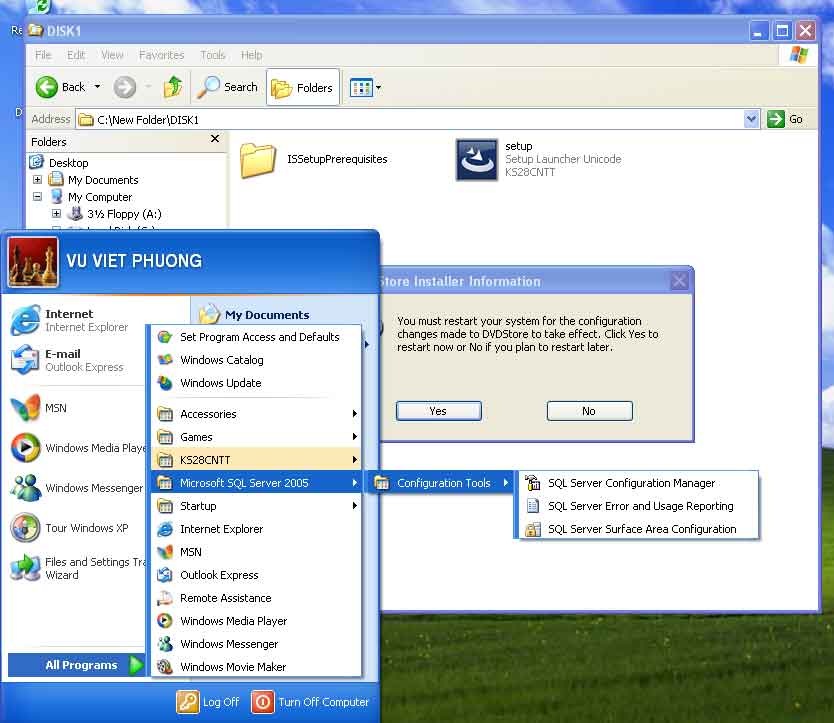
- IReport

**4.2.2. Các bước cài đặt:**

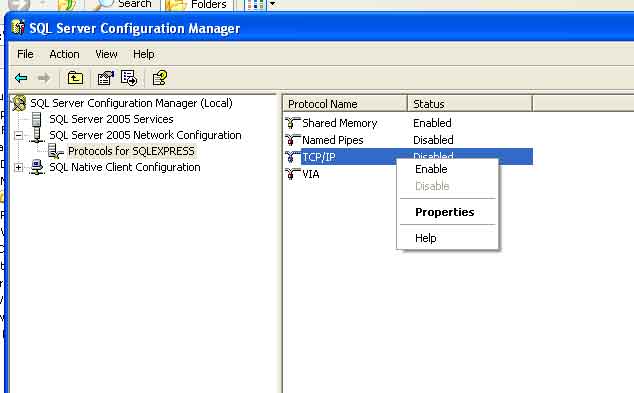
(Để chương trình DVDStore hoạt động tốt, trước khi cài đặt cần đảm bảo ngày hệ thống là **16/4/2010.** Do DVDStore là chương trình Demo, trong chương trình có sử dụng Alloy LookAndFeel là thư viện có tính phí của Incors, hạn sử dụng chỉ đến **17/4/2010**)

- Hệ thống sử dụng Sql Server 2005, để sử dụng, cần cài đặt .net framework 3.5 và chương trình Sql Server 2005 Express (download miễn phí trên trang microsoft.com), chú ý sau khi cài đặt Sql Server cần config 1 số thông số như sau:

1. Vào Start 🡪 All Programs 🡪 Sql Server 2005 🡪 Configuration Tools 🡪 Sql Server Configuration Manager



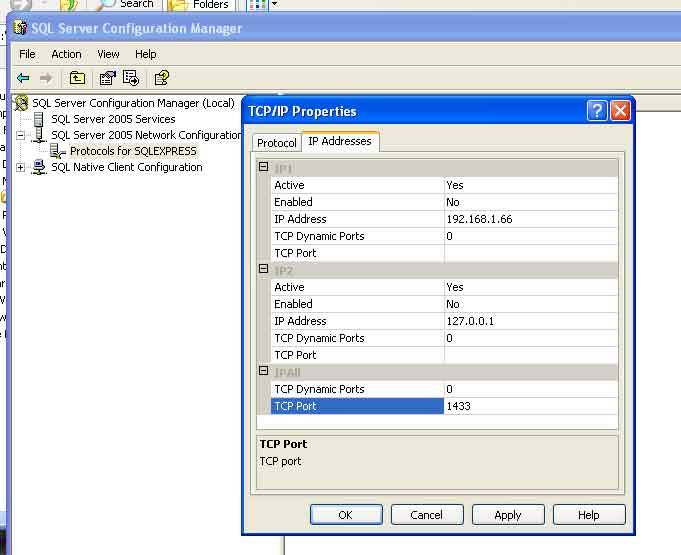
1. Vào mục Sql Server 2005 Network Configuration 🡪 Protocols for SqlExpress 🡪 TCP/IP 🡪 Click chuột phải 🡪 Enable



3. Click chuột phải TCP/IP 🡪 Properties 🡪 IP Addresses

Mục IPAll 🡪 Đặt TCP Port là 1433 🡪 OK

Khởi động lại máy tính



- Sau khi cài Sql Server, cần Attach Database, database được chứa trong thư mục: **/DVDStoreWeb/database/DVDStoreDB.mdf**

**-** Thông số truy cập SQL Server 2005 (URL, username, password) cần khớp với thông số trong file Hibernate Config: **DVDStoreWeb\** **web\WEB-INF\classes\hibernate.cfg.xml**

<property name="hibernate.connection.url">jdbc:sqlserver://vstorm83:1433;databaseName=DVDStoreDB</property>

<property name="hibernate.connection.username">sa</property>

<property name="hibernate.connection.password">123</property>

- Bộ cài bao gồm:

1. DVDStoreComponentTemplate: Thư mục này chứa thư viện Custom Tag dùng cho trang Web

2. DVDStoreWeb: Thư mục này chứa trang web và các webservice

3. DVDStoreService: Thư mục này chứa các WebService façade và Value Object

4. DVDDesktop: Thư mục này chứa chương trình desktop dùng cho Admin (Chủ cửa hàng)

5. apache-activemq-5.3.0 : Thư mục này chưa ActiveMQ JMS Server

6. Tomcat 6.20.exe : File cài đặt Tomcat

- Các thư mục trên là đều là Netbeans project, để mở source code cần cài:

- JDK 6 u18 (http://java.sun.com/javase/downloads/index.jsp)

- Netbeans 6.8 (http://netbeans.org/downloads/index.html)

- Để deploy web cần cài thêm

- JRE 6u18 (http://java.sun.com/javase/downloads/index.jsp)

- Tomcat 6.20 (chạy file cài đi kèm , làm theo hướng dẫn của chương trình), cài xong tomcat, cần chỉnh lại thông số VM option như sau: **-XX:PermSize=128m -XX:MaxPermSize=196m**

- Copy thư mục **apache-activemq-5.3.0 vào** ổ cứng, đây là chương trình jms server

- Sau khi cài tomcat, copy file **DVDStoreWeb\dist\DVDStoreWeb.war** vào thư mục ${Tomcat Dir}\webapps. Khởi động lại tomcat

- Khởi động Active MQ : chạy file **${ActiveMQ Dir}\ bin\win32\wrapper.exe**

- Để chạy chương trình desktop: chạy file **DVDStoreDesktop\dist\DVDStoreDesktop.jar**