**Tài liệu thiết kế phần mềm MỤC LỤC**

**I. GIỚI THIỆU 4**

1. Mục đích 4

2. Phạm vi 4

3. Khái niệm, viết tắt 4

**II. BIỂU DIỄN KIẾN TRÚC VÀ THIẾT KẾ 5**

1. Ràng buộc về mặt kỹ thuật [5](#_heading=h.2s8eyo1)

2. Góc nhìn logic 5

a. Presentation Layer 6

b. Business Layer 6

c. Data Access Layer 7

d. Common classes 7

3. Các package thiết kế quan trọng về mặt cấu trúc 9

a. Kết cấu package 9

b. Xử lý lỗi và ngoại lệ 10

c. Log, trace và debug 11

d. Flow chung [11](#_heading=h.3whwml4)

4. Kiến trúc phần mềm và triển khai 12

**I. GIỚI THIỆU**

***1. Mục đích***

Tài liệu này đưa ra thiết kế kiến trúc của hệ thống Java, sẽ được phát triển trong dự án AB-SD; nó cũng sẽ cung cấp thông tin chi tiết của tất cả các thành phần và giao diện trong đó.

***2. Phạm vi***

Tài liệu này sẽ mô tả bức tranh tổng quát về kiến trúc phần mềm:

* Tất cả các công nghệ sẽ được sử dụng trong ứng dụng
* Tổng quan về các tầng và mô-đun trong ứng dụng
* Các non-functional requirement chẳng hạn như khả năng sử dụng, độ tin cậy, hiệu suất, khả năng hỗ trợ và bảo mật

***3. Định nghĩa, viết tắt***

| **Term** | **Description** |
| --- | --- |
| API | Application Programming Interface |
| J2EE | Java 2 Platform, Enterprise Edition |
| JVM | Java Virtual Machine |
| UI | User Interface |
| HTML | Hyper Text Mark-up Language |
| JSP | Java Server Page |
| JSTL | Java Standard Tag Library |
| MVC | Model – View – Controller model for application interaction |
| Struts | <http://struts.apache.org/>  An open-source framework from the Jakarta Project. Struts is designed for building web applications with the Java Servlet API and JSP technology. The Struts package supplies an integrated set of reusable components. They include a controller servlet, JSP custom tag libraries, and utility classes. These components for building user interfaces can be applied to any web-based Java application. |
| DAO | Data Access Objects |
| JDBC |  |

**II. BIỂU DIỄN KIẾN TRÚC VÀ THIẾT KẾ**

***1. Ràng buộc về mặt kỹ thuật***

Sau đây là những ràng buộc phải được xem xét khi thiết kế kiến trúc cho hệ thống:

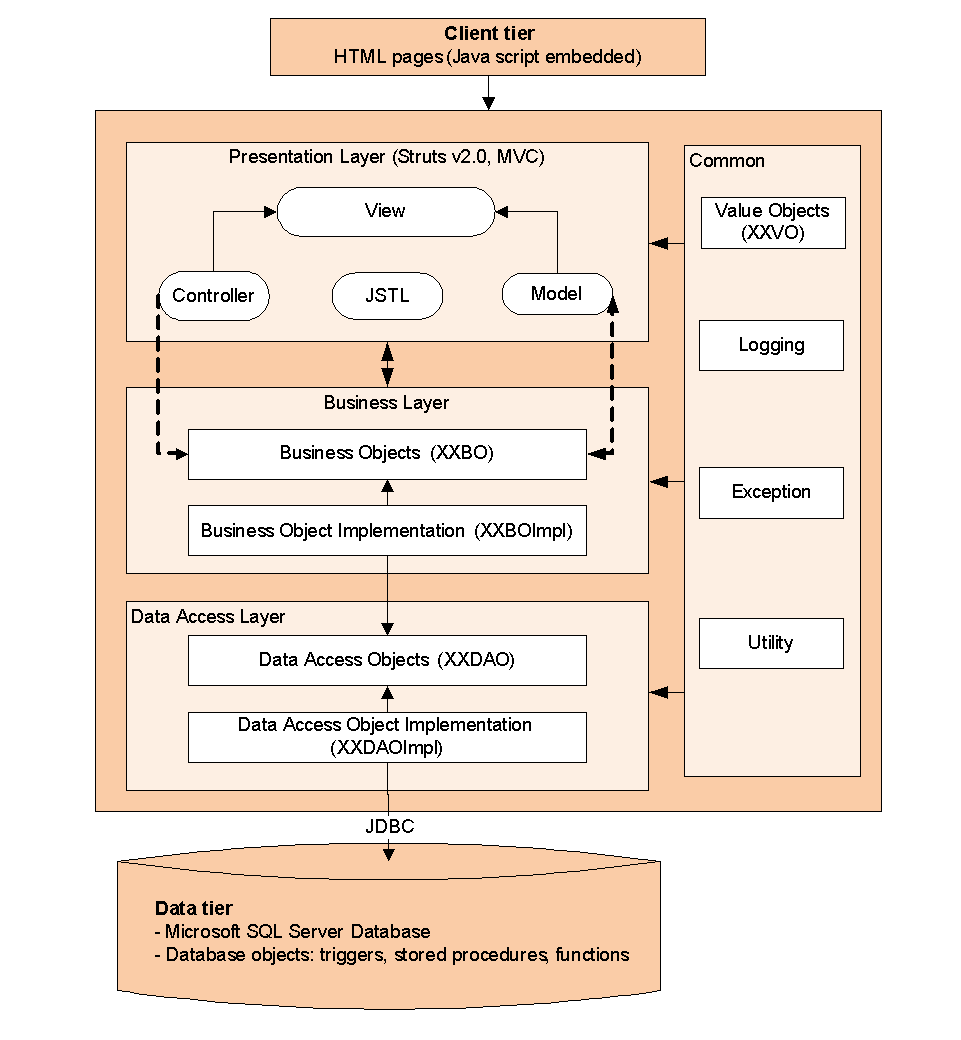
* System Framework: Use MVC2 (Struts v2.0)
* Client Framework: Use HTML, JavaScript
* Web support: IE 5.0 or later, Firefox 2.0 or later
* Database: Microsoft SQL Server
* Web Application server: Apache Tomcat

***2. Góc nhìn logic***

Phần này mô tả các công nghệ & framework được sử dụng trong tất cả các layer của ứng dụng này

Kiến trúc của hệ thống được thiết kế theo kiến trúc tiếp cận n-tier và J2EE tiêu chuẩn công nghiệp.

Sơ đồ dưới đây cho thấy các layer (tầng) logic chính của kiến trúc và cách chúng tương tác với nhau



**a. Presentation Layer**

Layer này kiểm soát việc hiển thị cho người dùng cuối;

Struts 2 framework được chọn làm phần triển khai cho mô hình MVC (Model View Controller) trong layer này. Struts framework chịu trách nhiệm về:

* Quản lý các yêu cầu/phản hồi từ/đến khách hàng.
* Kiểm soát việc hiển thị UI cho người dùng cuối.
* Lắp ráp một mô hình có thể được trình bày trong một view.
* Thực hiện xác nhận UI.
* Cung cấp một bộ điều khiển để ủy quyền các lệnh gọi đến business layer.
* Xử lý các ngoại lệ từ các layer khác tạo ra các ngoại lệ cho một Struts Action.

**b. Business Layer**

Layer này quản lý các quy tắc và logic của quá trình kinh doanh. Nó bao gồm một tập hợp các Business Object để hỗ trợ việc triển khai logic kinh doanh của hệ thống. Sự hiện diện của layer này là để thêm tính linh hoạt giữa presentation layer và persistence layer để chúng không giao tiếp trực tiếp với nhau.

* Xử lý logic kinh doanh và xác thực kinh doanh.
* Quản lý các giao dịch.
* Cho phép các giao diện tương tác với các layer khác.
* Quản lý các phụ thuộc giữa các đối tượng cấp kinh doanh.
* Thêm tính linh hoạt giữa presentation và persistence layer để chúng không giao tiếp trực tiếp với nhau.
* Hiển thị các dịch vụ kinh doanh do layer này cung cấp cho presentation layer.
* Quản lý việc triển khai từ logic kinh doanh đến data access layer.

**c. Data Access Layer**

Layer này quản lý việc truy cập vào bộ nhớ liên tục thông qua JDBC. Nó quản lý việc đọc, ghi, cập nhật và xóa dữ liệu được lưu trữ.

Lý do chính để tách data access layer khỏi Business Layer là để đảm bảo rằng việc triển khai kinh doanh và truy cập dữ liệu được liên kết chặt chẽ với nhau. Nhờ có phân tách này, việc chuyển đổi nguồn dữ liệu và Data Access Objects (DAOs) giữa các ứng dụng sau này sẽ dễ dàng hơn, giúp tăng khả năng tái sử dụng, khả năng bảo trì và tính linh hoạt của ứng dụng.

**d. Common classes**

***Value Objects***: chứa kết cấu nhẹ cho các thông tin kinh doanh liên quan. Một value object (VO –đôi khi được gọi là Transfer Object) là một đối tượng nhẹ, có thể tuần tự hóa, cấu trúc các nhóm dữ liệu thành một cấu trúc logic duy nhất. (các value object luôn triển khai java.io.Serializable).

* Một VO nhằm mục đích giảm thiểu lưu lượng mạng giữa các enterprise bean và caller của chúng (bởi vì mỗi đối số được truyền qua sẽ bắt đầu một quá trình truyền mạng).
* Một VO được thiết kế để cải thiện hiệu suất của các enterprise bean bằng cách giảm thiểu số lượng đối số của phương thức, do đó, việc truyền mạng cần phả gọi chúng.
* Ngoài ra, các VO cũng rất hữu ích trong việc giao tiếp giữa tất cả các layer của ứng dụng.

***Utility Package*:** chứa các lớp tiện ích cung cấp các chức năng phổ biến như: xử lý chuỗi, chuyển đổi ngày tháng, v.v.

***Logging Package***:

Package này chứa các lớp để ghi thông tin vào file log được dùng cho mục đích debug/audit. Commons logging framework sẽ được sử dụng cho việc triển khai Log4j.

* Thành phần log4j từ Apache (<http://logging.apache.org/log4j/docs/index.html>) sẽ được sử dụng để triển khai package này bởi vì nó có thể cho phép logging trong thời gian chạy mà không cần sửa đổi ứng dụng nhị phân. Package log4j được thiết kế để các câu lệnh này vẫn có thể ở trong code đã vận chuyển mà không phải chịu chi phí về hiệu suất cao.
* Chỉnh sửa file cấu hình mà không cần chạm vào ứng dụng nhị phân, có thể kiểm soát hành vi logging.
* Log có 4 cấp độ: Error, Info, Debug và Warn.
* **DEBUG**, đây là cấp mang lại những thông tin tốt nhất. Việc sử dụng các câu lệnh DEBUG được khuyến khích nhưng không nên để máy chủ sản xuất (production servers) ở chế độ DEBUG bởi điều này sẽ làm giảm hiệu suất của máy chủ và làm đầy các file log.
* **INFO**, thông tin ở cấp này ít chi tiết hơn so với cấp DEBUG. Các thông báo cung cấp thông tin phải làm nổi bật tiến trình của một ứng dụng. Ví dụ, cấp INFO có thể được sử dụng để đánh dấu khi một lô hệ thống đang bắt đầu và kết thúc. Lưu ý, INFO không nên được sử dụng theo cách mà các file log sẽ mở rộng một cách nhanh chóng.
* **WARN**, chỉ ra một tình huống có thể gây hại.
* **ERROR** chỉ định các event lỗi. Ví dụ, mọi ngoại lệ trong ứng dụng đều phải được ghi lại ở cấp độ này.

Trong môi trường Kiểm thử, tất cả các cấp độ log đều có thể được bật (bằng cách chỉnh sửa file cấu hình) nhằm cung cấp thông báo lỗi đầy đủ. Trong môi trường sản xuất, cấp độ ERROR log có thể được bật để chỉ cung cấp thông báo lỗi thân thiện.

***Exception Package***:

Package này sẽ bao gồm tất cả các ngoại lệ chung thường được sử dụng bởi nhiều hơn một package. Nếu có thể, nên sử dụng Struts Exception Handler. Các mệnh đồ try-catch nên được giữ ở mức tối thiểu. Trong trường hợp xuất hiện một ngoại lệ thì ngoại lệ ban đầu phải được đưa vào hàm khởi tạo.

Có hai loại ngoại lệ:

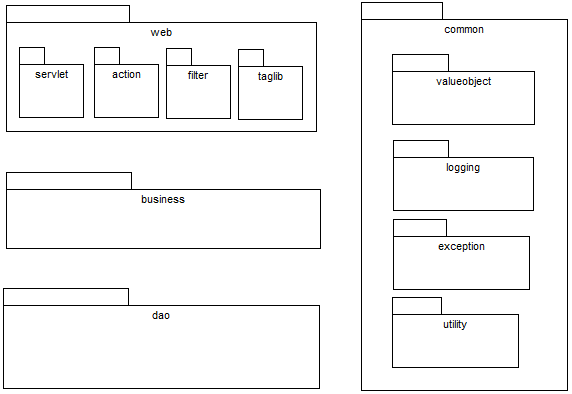
* *System Exception*: một ngoại lệ sẽ được ném ra trong trường hợp hệ thống xảy ra lỗi như lỗi kết nối với database, lỗi đọc hệ thống file…
* *Functional Exception*: một ngoại lệ liên quan đến logic kinh doanh, ví dụ như xác thực dữ liệu đầu vào, xác minh…

Tối thiểu, các chi tiết sau phải luôn được ghi lại và hiển thị trong các thông báo lỗi:

* Điều gì đã xảy ra khi lỗi được xác định. Điều này sẽ bao gồm Lớp, Hàm, Phương thức được gọi là gì và đến từ đâu.
* Giá trị của bất kỳ Biến nào tại thời điểm lỗi được xác định.
* Tại sao lại xảy ra lỗi. Bất kỳ dấu hiệu gây ra lỗi nào của hệ thống cũng phải được báo cáo.

***3. Các package thiết kế quan trọng về mặt cấu trúc***

**a. Kết cấu package**



Tên package cho ứng dụng là mock.appcode (trong đó: appcode là code hoặc tên viết tắt của ứng dụng được chọn có liên quan). Mỗi thành phần được chia thành các sub-package như được mô tả dưới đây:

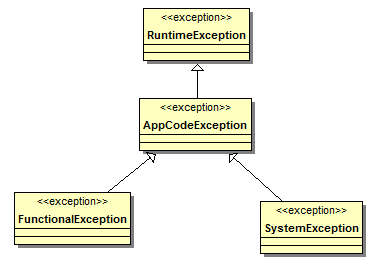
| **Package Name** | **Description** | **Naming Convention** |
| --- | --- | --- |
| mock.appcode.web.servlet | Contains all related servlet | xxServlet |
| mock.appcode.web.action | Contains all Struts action | xxAction |
| mock.appcode.web.filter | Contains filter classes | xxFilter |
| mock.appcode.web.taglib | Contains all taglibs used in the application | xxTag |
| mock.appcode.business | Contain all business interface & implementation | xxBO  xxBOImpl |
| mock.appcode.dao.daointerface | Contain all data access interface & implementations | xxDAO  xxDAOImpl |
| mock.appcode.common.valueobjects |  | xxVO |
| mock.appcode.common.utility | Contains utility classes | xxUtility |
| mock.appcode.common.exception | Contains exception classes | AppCodeException  FunctionalException  SystemException |
| mock.appcode.common.logging | Contains logging classes |  |

Trong số các package của hệ thống, business package không hề nhỏ và sẽ do các học viên chuẩn bị (thông qua các sơ đồ lớp, sơ đồ trình tự và code giả).

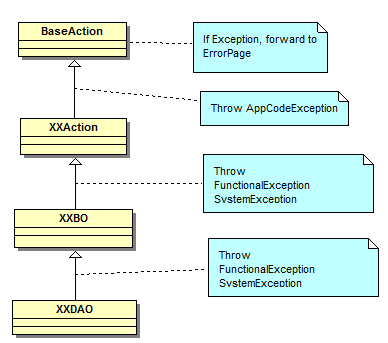
**b. Xử lý lỗi và ngoại lệ**

Ứng dụng sẽ sử dụng cơ chế Exception, trong đó tất cả các ngoại lệ ném ra AppCode sẽ được mở rộng từ RuntimeException. Biểu đồ dưới đây mô tả các package exception

***Sơ đồ lớp***



***Cơ chế sử dụng***



*FunctionalException*: là các ngoại lệ logic, chúng được ném ra trong trường hợp các tham số truyền giữa các phương thức không đầy đủ, sai kiểu, thiếu dữ liệu (dữ liệu hoặc xác thực kinh doanh), v.v.

*SystemException*: là các RuntimeException, bất kỳ lỗi không mong muốn nào như hết bộ nhớ, gián đoạn database, v.v.

**c. Log, trace và debug**

Ứng dụng sử dụng log4j làm logging package tiêu chuẩn. Chúng ta chia logging làm 4 cấp độ:

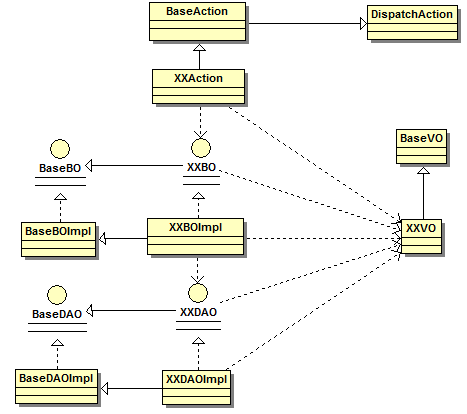
* **Info**: cấp này ít chi tiết hơn so với cấp DEBUG. Các thông báo cung cấp thông tin phải làm nổi bật tiến trình của một ứng dụng. Ví dụ, có thể sử dụng cấp INFO để đánh dấu khi một lô hệ thống đang bắt đầu hoặc kết thúc. Lưu ý, không nên sử dụng cấp INFO theo cách mà file log sẽ mở rộng nhanh chóng.
* **Warn**: chỉ ra một tình huống có thể gây hại.
* **Debug**: đây là cấp mang lại những thông tin tốt. Việc sử dụng những câu lệnh DEBUG được khuyến khích nhưng không nên để máy chủ sản xuất ở chế độ DEBUG bởi điều này sẽ làm giảm hiệu suất của máy chủ và làm đầy các file log.
* **Error**: chỉ định các lỗi của event. Ví dụ, mọi ngoại lệ trong ứng dụng đều phải được ghi lại ở cấp độ này.

**d. Flow chung**

Phần này trình bày the code flow chung theo sơ đồ lớp và sơ đồ trình tự của một use case phổ biến.

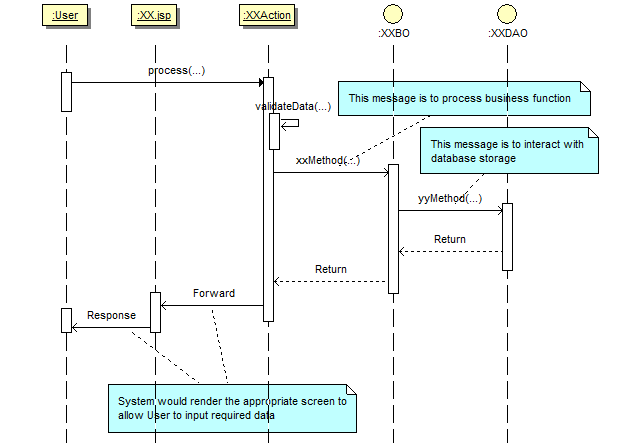
***Sơ đồ lớp***

Sơ đồ lớp điển hình cho use case được trình bày dưới đây

****

***Sơ đồ trình tự***

Sơ đồ trình tự cho use case được trình bày dưới đây

****

***4. Kiến trúc và triển khai phần mềm***

| **Application Server** | |
| --- | --- |
| J2EE Application server | Apache Tomcat |
| Operating Systems | Windows Server |
| JDBC driver | TBD |
| Application Logging | Apache Jakarta Log4j (Deployed with the Application) |
| **HTTP Server** | |
| Web Server | Apache |
| Operating Systems | Windows Server |
| **Database Server** | |
| Operating Systems | Windows Server |
| DB Management System | Microsoft SQL Server |