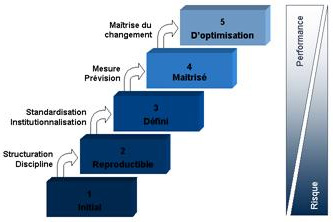
**1/ Cmmi**

# Giới thiệu CMMI

# Mô hình CMMI(Capability Maturity Model Integration) là một khung các giải pháp tối ưu cho quá trình sản xuất phần mềm. Phiên bản CMMI-DEV hiện nay (CMMI cho chuyên viên phát triển), mô tả những giải pháp tốt nhất trong quá trình kiểm soát, đo lường và kiểm tra các quy trình phát triển phần mềm. Mô hình CMMI không tập trung mô tả chính các quá trình mà chỉ mô tả đặc điểm của các quá trình hiệu quả, vì vậy mô hình CMMI đưa ra chỉ dẫn cho các công ty để họ có thể tự mình  phát triển hoặc điều chỉnh chính các quá trình của họ.



Mô hình CMMI được mô tả trên trang web chính thức CMMI website :dự án CMMI là một nỗ lực chung nhằm cung cấp các mô hình để cải thiện nâng cấp các sản phẩm và quy trình. Trọng tâm chính của dự án là tập trung xây dựng các công cụ hỗ trợ việc cải thiện các quy trình dùng để phát triển và ổn định các hệ thống và sản phẩm. Kết quả của dự án CMMI là một bộ các sản phẩm cung cấp một phương pháp tiếp cận tích hợp trên toàn doanh nghiệp để cải thiện các quy trình sản xuất mà vẫn có thể giảm bớt nhân công dư thừa, độ phức tạp, và chi phí từ việc sử dụng các mô hình CMM (quy trình quản lý sản xuất phẩn mềm) riêng lẻ và nhiều mô hình CMM.  
Mô hình này xác định năm cấp độ của CMM đối với một công ty :

1. Khởi đầu (lộn xộn, không theo chuẩn): đây là điểm khởi đầu để sử dụng một quy trình mới.
2. Lặp (quản lý dự án, tuân thủ quy trình) : Quy trình này được lặp lại nhiều lần
3. Xác lập (thể chế hóa): Quy trình này được xác lập/ xác nhận như một quy trình doanh nghiệp tiêu chuẩn.
4. Kiểm soát (định lượng): Tiến hành kiểm soát và đo lường quy trình sản xuất phần mềm
5. Tối ưu (cải tiến quy trình): Kiểm soát quy trình bao gồm việc cân nhắc kỹ để cải tiến/ tối ưu hóa quy trình.

**1.2. Ứng dụng CMMI**

CMMI bao gồm hàng loạt các quy trình trong phát triển phần mềm từ khi bắt đầu dự án đến khi kết thúc, kèm theo là các tài liệu đi kèm, quản lý cấu hình,… Do quy mô của dự án nhỏ và thời gian không cho phép nên dự án sẽ chỉ tập trung vào cách tổ chức thư mục và quản lý source trong quá trình phát triển.

Cấu trúc thư mục của dự án:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Mô hình cấu trúc thư mục của dự án (mức System) | | | | | | | |
| **Cấp 1** | Cấp 2,3,4.. | | | | | | Giải thích |
| **01\_Contracts** |  | | | | | | Lưu trữ các văn bản hợp đồng của dự án |
| **02\_Baselines** | 01\_BAL | | | 01\_BAL\_<Tên viết tắt của dự án>\_YYYYMMDD | | | Lưu trữ các dữ liệu (các thành phần cấu hình tại version xác định) tại các mốc đặt Baseline của dự án – sử dụng cho việc xây dựng lại hệ thống sau này (nếu cần) và các phiên bản bàn giao (Release) cho khách hàng (hoặc các bên liên quan)  Các thành phần cấu hình tại Baseline được lấy từ vùng lưu trữ các thành phần đã được rà soát và phê duyệt (Mục 03\_Approval) |
| 02\_REL | | | 01\_REL\_Tên viết tắt của dự án>\_YYYMMDD | | |
| **03\_Approval** | 01\_Requirements | | | | | | Các tài liệu đã được rà soát và phê chuẩn chốt phiên bản được lưu tại khu vực này.  Các tài liệu có phiên bản sẽ được chọn lựa để lập dữ liệu cho Baseline và Release bàn giao |
| 02\_Analysis & Design | | | | | |
| 03\_Sources | | | | | |
| 04\_Implementation | | | | | |
| 05\_Testing | | | | | |
| 06\_System Deployment | | | | | |
| 07\_Project Management | | | | | |
| 08\_Development | | | | | |
| 09\_Configuration & Change Management | | | | | |
| **04\_Working** | 01\_Requirements | Documents | | | | | Các tài liệu yêu cầu của dự án [Tài liệu đặc tả yêu cầu phần mềm](../../03.QT%20Quan%20ly%20yeu%20cau/Template/03.TPL.Tai%20lieu%20dac%20ta%20yeu%20cau%20phan%20mem.doc) |
| Models | | | | | Các tài liệu mô hình các thành phần của phần mềm: [Tài liệu đặc tả usecase](../../03.QT%20Quan%20ly%20yeu%20cau/Template/03.TPL.Tai%20lieu%20dac%20ta%20usecase.doc) |
| 02\_Analysis & Design | Analysis | | | | | Các tài liệu phân tích thiết kế của phần mềm: [Tài liệu Kiến trúc phần mềm](../../04.QT%20Thiet%20ke%20phan%20mem/Template/04.TPL.%20Kien%20truc%20phan%20mem.doc), [tài liệu thiết kế dữ liệu](../../04.QT%20Thiet%20ke%20phan%20mem/Template/04.TPL.Thiet%20ke%20chi%20tiet_Thiet%20ke%20CSDL.doc), [Tài liệu thiết kế lớp](../../04.QT%20Thiet%20ke%20phan%20mem/Template/04.TPL.Thiet%20ke%20chi%20tiet_Thiet%20ke%20lop.doc), [tài liệu thiết kế màn hình](../../04.QT%20Thiet%20ke%20phan%20mem/Template/04.TPL.Thiet%20ke%20chi%20tiet_Thiet%20ke%20man%20hinh.doc) |
| Design | | | | |
| Database | | | | |
| 03\_Sources | | | | | | Khu vực lưu trữ mã nguồn của phần mềm |
| 04\_Implementation | | Build | | | | Các chương trình thực thi của phần mềm |
| SubSystem | | | |
|  | | | |
| 05\_Testing | | Unit Test | | | | Các tài liệu Kiểm thử của dự án qua các giai đoạn Kiểm thử: Thiết kế Kiểm thử và các kết quả kiểm thử: [Các trường hợp Kiểm thử](../../06.QT%20Kiem%20thu/Template/06.TPL.Cac%20truong%20hop%20kiem%20thu.xls), [Báo cáo kết quả Kiểm thử](../../06.QT%20Kiem%20thu/Template/06.TPL.Bao%20cao%20ket%20qua%20kiem%20thu.doc) |
| Integration Test | | | |
| System Test | | | |
| Acceptance Test | | | |
| 06\_System Deployment | | Documents | | | | Các tài liệu triển khai dự án: [Thủ tục triển khai](../../07.QT%20Trien%20khai/Template/07.TPL.Thu%20tuc%20trien%20khai.doc), [Hướng dẫn sử dụng](../../07.QT%20Trien%20khai/Template/07.TPL.Huong%20dan%20su%20dung.doc), Hướng dẫn cài đặt … |
| Manuals | | | |
| Installation | | | |
| 07\_Project Management | | Plans | | Project | | Các tài liệu Kế hoạch của dự án: [Kế hoạch phát triển](../../02.QT%20Quan%20ly%20du%20an/Template/02.TPL.Ke%20hoach%20du%20an.doc), [Kế hoạch quản lý cấu hình](../Template/09.TPL.Ke%20hoach%20quan%20ly%20cau%20hinh.doc), [Kế hoạch Kiểm thử](../../06.QT%20Kiem%20thu/Template/06.TPL.Ke%20hoach%20kiem%20thu.doc), Kế hoạch Tích hợp[, Kế hoạch triển khai](../../07.QT%20Trien%20khai/Template/07.TPL.Ke%20hoach%20trien%20khai.doc) |
| CCM | |
| QA | |
| Test | |
| Integration | |
| Deployment | |
| TimeSheet | | | | Các tài liệu về biểu thời gian, báo cáo, ghi chú cuộc họp, các tài liệu đánh giá ước lượng dự án..  Lịch biểu thời gian cho tất cả các thành viên của dự án |
| Records | | | |
| Meeting | | | |
| Estimation | | | |
| Reports | | | |
| **08\_Development** | | | | | | Khu vực lưu trữ các sản phẩm của dự án trên các công cụ sử dụng cho dự án  (Mục này là tùy chọn) |
| 09\_Configuration & Change Management | | | | | | Khu vực quản lý cấu hình và kiểm soát thay đổi của dự án: [Danh sách các thành phần cấu hình](../Template/09.TPL.Danh%20sach%20cac%20thanh%20phan%20cau%20hinh.xls)  [Yêu cầu thay đổi](../Template/09.TPL.Yeu%20cau%20thay%20doi.doc) |
|  | | | | | | |
| **05\_Backup** | BAK\_<Tên viết tắt của dự án>\_YYYYMMDD | | | | | Khu vực lưu trữ các dữ liệu sao lưu của dự án | |
| **06\_Reused** | Nếu có | | | | | | |
| **07\_References** | Chứa tài liệu, biểu mẫu tham khảo phục vụ việc thực hiện dự án, Các tài liệu hướng dẫn đặc biệt của dự án (Project Specific Guidelines)…Người quản lý cấu hình có thể tạo thêm các thư mục khác tại đây nếu cần | | | | | | |

Bên cạnh việc tổ chức thư mục, dự án có sử dụng Subversion (SVN) và Google Code để quản lý source code.

**3/ JSF**

3.1. Sự phát triển của các web application framework

Trải qua 3 giai đoạn chính:

- Không theo mô hình MVC ( model, view, controller )

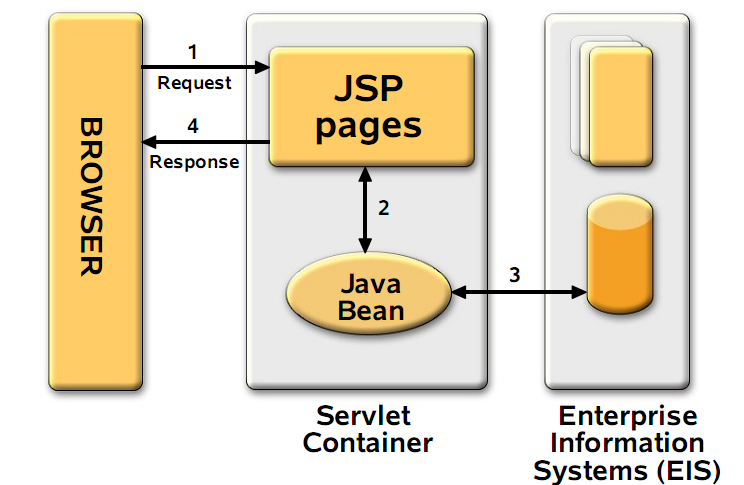
- MVC model 1 (Page-centric)

- MVC model 2 (Servlet-centric)

a/ Không theo mô hình MVC

Thời kì mới ra đời của các ứng dụng web, lúc đó chỉ tồn tại các ứng dụng web tương tác trực tiếp với người sử dụng bằng các trang web tĩnh, không có sự xử lý và ngôn ngữ phía máy chủ. Điều này đã gây cản trở rất lớn cho người lập trình cũng như người sử dụng trong việc tiếp cận với các ứng dụng trên nền web.

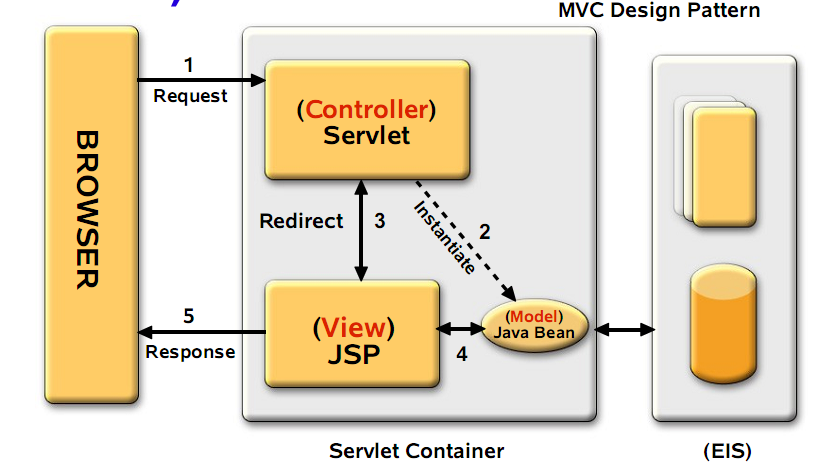
b/ MVC model 1



Trong MVC model 1, các trang JSP đóng vai trò Hiển thị (View) và Điều khiển (Controller). Có thể có nhiều trang JSP khác nhau đóng các vai trò khác nhau.  
  
\* Khi người sử dụng dùng các nút bấm, menu hoặc link … trên trình duyệt Web (Web browser) để thực hiện một thao tác, một lệnh (có thể kèm theo các tham số) được gửi tới một trang JSP tương ứng.  
\* Trang JSP này sẽ khởi tạo một hoặc nhiều Java Bean (nếu cần thiết), truyền các lệnh cần thi hành tới Java Bean. Chú ý rằng đây là các Java Bean thông thường, chứ không phải Enterprise Java Bean (EJB)  
\* Sau khi Java Bean thực hiện xong việc truy xuất hoặc cập nhật dữ liệu, trang JSP ban đầu có thể hiển thị dữ liệu lấy từ Bean (JSP ban đầu đóng luôn vai trò View), hoặc chọn một trang JSP khác để hiện dữ liệu từ Bean (JSP ban đầu đóng luôn vai trò Controller). Trong một thiết kế tốt, để bảo đảm việc tách rời phần trình bày và logic của chương trình, trang JSP nhận request chỉ đóng vai trò Điều khiển (Controller).

MVC model 1 có một nhược điểm là phần logic điều khiển được viết trong trang JSP, như vậy phần chương trình Java phức tạp dùng để điều khiển sẽ bị lẫn vào trong mã HTML dùng để trình bày. Độ phức tạp của chương trình càng cao, thì trang JSP càng khó bảo trì. Hơn nữa trong các dự án phần mềm phức tạp, thì phẩn hiển thị của trang JSP thường được làm bởi người thiết kế Web, giỏi về HTML và đồ họa, còn phần chương trình Java được viết bởi lập trình viên chuyên về lập trình. Trong các dự án phức tạp, dùng JSP làm phần điều khiển sẽ làm lẫn lộn việc phân chia ranh giới trách nhiệm giữa nhóm thiết kế đồ họa và nhóm lập trình, đôi khi dẫn đến việc bảo trì và phát triển trở nên rất khó khăn, gần như không thể làm được.  
Để khắc phục nhược điểm này, MVC model 2 ra đời

c/ MVC model 2



Trong MVC model 2, một hoặc nhiều servlet (thường là một) đóng vai trò Điều khiển, các Java Bean đóng vai trò Mô hình và các trang JSP đóng vai trò hiển thị.

Trong model 2, các logic phức tạp của chương trình được viết hoàn toàn trong các servlet, là các chương trình Java. Phần hiển thị chỉ gồm các trang JSP với một vài mã đơn giản để lấy dữ liệu có sẵn, không có logic phức tạp, vì thế hoàn toàn có thể được tạo ra bằng những người thiết kế Web.

Các yêu cầu của người dùng được gửi từ trình duyệt Web tới servlet. Servlet sẽ khởi tạo Java Bean (nếu cần thiết), ra lệnh thu thập, cập nhật thông tin. Khi Java Bean hoàn thành công việc, servlet sẽ chọn trang JSP thích hợp để hiện thông tin trong Java Bean cho người dùng.

**3.2.** **JavaServer Faces là gì?**

Công nghệ Java Server Faces là một UI framework cho việc xây dựng các ứng dụng web chạy trên Java server và thay thế UI phía sau cho client. Các ứng dụng được viết bằng JSF tuân theo mô hình MVC model 2, hỗ trợ tốt cho người phát triển web. Các thành phần chính của công nghệ JSF bao gồm:

* Một API và các bổ sung tham khảo cho: thay thế các thành phần UI và quản lý trạng thái của chúng; xử lý các sự kiện, kiểm tra phía server và chuyển đổi dữ liệu; định nghĩa navigation của trang; hỗ trợ quốc tế hóa và accessibility; và cung cấp khả năng mở rộng cho tất các đặc điểm này.
* Một thư viện thẻ tùy biến JavaServer Pages (JSP) cho việc định nghĩa các thành phần UI trong một trang JSP

Mô hình lập trình được định nghĩa tốt này và thư viện thẻ thành phần UI tạo kỹ thuật dễ dàng tải việc xây dựng và sửa chữa các ứng dụng web với các UI ở phía server. Với sự tổ chức nhỏ đó, bạn có thể:

* Điều khiển việc tạo ra các sự kiện phía client từ việc viế mã ứng dụng phía server
* Ánh xạ các thành phần UI tren một trang cho dữ liệu phía server
* Khởi dựng một UI với các thành phần có thể tái sử dụng và có khả năng mở rộng
* Lưu trữ và phục hồi trạng thái UI ngay sau các request

**3.3. Phần mềm yêu cầu**

Để xây dựng, deploy và chạy các ứng dụng JSF bạn cần một môi trường deploy chẳng hạn Java Web Software Development Pack và Java 2 Platform, Standard Edition (J2SE) SDK 1.3 trở lên. Đồng thời download bổ sung công nghệ Java Server Faces. Có thể download các phần mềm cần thiết tại các link dưới đây

* http://java.sun.com/webservices/downloads/webservicespack.html
* http://java.sun.com/j2se/1.5
* http://java.sun.com/j2se/javaserverfaces/download.html

**3.4. Đặc điểm của JSF**

Một trong những lợi điểm lớn nhất của công nghệ JSF là nó cho phép một sự phân chia rạch ròi giữa behavior (cách xử lý) và presentation (cách trình bày). Xây dựng ứng dụng web với công nghệ JSP lưu trữ từng phần của việc phân chia này. Tuy nhiên, một ứng dụng JSP không thể ánh xạ những request HTTP thành những xử lý sự kiện các thành phần cụ hể hoặc quản lý các thành phần UI như những đối tượng có trạng thái trên server. Công nghệ JSF cho phép bạn xây dựng các ứng dụng Web nhằm bổ sung việc phân chia rõ ràng hơn behavior và presentation được cho phép bởi kiến trúc UI.

Việc phân chia luận lý từ presentation cũng cho phép mỗi thành viên của một nhóm phát triển ứng dụng Web tập trung vào những phần trong tiến trình phát triển của họ, và cung cấp một mô hình lập trình đơn giản để liên kết những phần đó với nhau. Một mục tiêu quan trọng khác của công nghệ JSF là cung cấp các mức độ thân thuộc các thành phần UI và các khái niệm tầng Web mà không giới hạn bạn trong một công nghệ scripting cụ thể hoặc một ngôn ngữ đánh dấu. Trong khi công nghệ JSF bao gồm một thư viện thẻ tùy biến JSP dùng thay thế các thành phần trên trang JSP, API của công nghệ JSP được phân lớp trực tiếp trên đỉnh của JavaServlet API. Điều này cho phép bạn làm được vài điều: sử dụng công nghệ trình bày khác bên cạnh JSP, tạo ra những thành phần tùy biến của bản thân bản trực tiếp từ những lớp thành phần, và tạo ra luồng xuất cho những thiết bị client khác nhau.  
  
Quan trọng hơn hết, công nghệ JSF cung cấp một kiến trúc dành cho việc quản lý trạng thái các thành phần, xử lý dữ liệu thành phần, kiểm tra nhập liệu của người dùng và xử lý các sự kiện. Trong hầu hết những phần đó, các ứng dụng JSF cũng tương tự như bất kỳ các ứng dụng Java Web khác, Chúng chạy trên một Java Servlet container, và thông thường chứa:

* Các thành phầns JavaBean (được gọi là những mô hình đối tượng trong công nghệ JSF)
* Các event listener
* Các trang, chẳng hạn như JSP
* Các lớp helper phía server, chẳng hạn như các bean truy cập dữ liệu

Thêm vào những thành phần ở trên, một ứng dụng JSF cũng có:

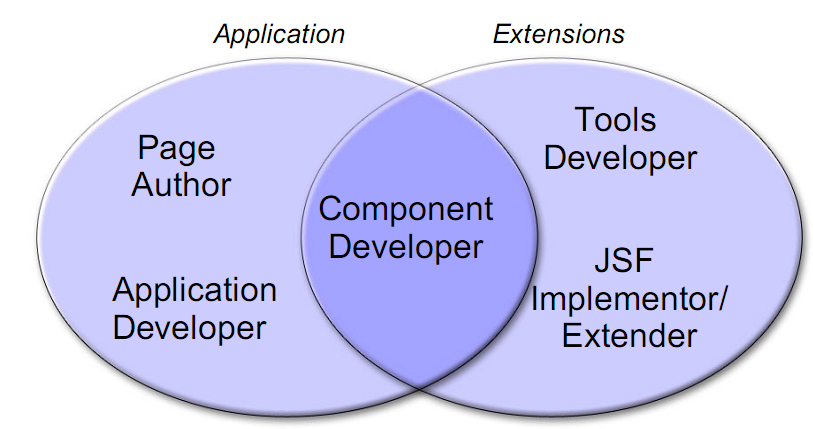
* Một thư viện thẻ tùy biến thực thi các thành phần UI trên một trang
* Một thư viện thẻ tùy biến thay thế các xử lý sự kiện, kiểm tra và những hành động khác
* Những thành phần UI thay thế trạng thái các đối tượng trên server
* Các kiểm tra, xử lý sự kiện, và xử lý navigation

Mỗi ứng dụng JSF phải bao gồm một thư viên thẻ tùy biến nhằm định nghĩa các thẻ thay thế các thành phần UI và một thư viện thẻ tùy biến nhằm thay thế các hành động cốt lõi khác, chẳng hạn như các kiểm tra và các xử lý sự kiện. Cả hai loại thư viện thẻ này được cung cấp bởi việc bổ sung JSF.  
  
Thư viện thẻ tùy biến xóa bỏ những gì cần thiết cho các thành phần UI trong HTML hoặc ngôn ngữ đánh dấu khác, kết quả là những thành phần tái sử dụng hoàn toàn. Và thư viện core tạo nên sự dễ dàng để đăng ký các sự kiện, kiểm tra và những hành động khác. Thư viện thẻ tùy biến có thể là thư viện thẻ HTML cơ bản chứa cùng với công nghệ JSF tham khảo bổ sung, hoặc bạn có thể định nghĩa thư viện thẻ của riêng mình nhằm tạo ra các thành phần tùy biến hoặc xuất ra kiểu khác HTML.  
  
Cuối cùng, công nghệ JSF cho phép bạn chuyển đổi và kiểm tra dữ liệu trên những thành phần riêng biệt và thông báo bất kỳ lỗi gì trước khi dữ liệu phía server được cập nhật.

**3.5. Vai trò của Framework**

Bởi vì việc phân chia công việc được cho phép bởi thiết kế công nghệ JSF, việc phát triển và sửa chữa các ứng dụng JSF có thể xử lý dễ dàng và nhanh chóng. Các thành viên củamột nhóm phát triển thông thường bao gồm một danh sách dưới đây. Trong nhiều nhóm, các nhà phát triển riêng biệt đóng nhiều hơn một trong những vai trò dưới đây

* PageAuthors: người sử dụng ngôn ngữ đánh dấu, giống như HTML, để tạo ra các trang cho ứng dụng Web. Khi sử dụng framework công nghệ JSF, page authors sẽ hầu hết sử dụng thư viện thẻ
* Application Developers: người lập trình mô hình các thành phần, các xử lý sự kiện, các kiểm tra, và navigation của trang. Application developer có thể cung cấp các lớp helper mở rộng
* Component Developers: người có kinh nghiệm lập trình UI và đề nghị tạo ra các thành phần tùy biến sự dụng ngôn ngữ lập trình. Những người này có thể tạo ra các thành phần của riêng họ trực tiếp từ các lớp thành phần, hoặc họ có thể kế thừa các thành phần chuẩn cung cấp bởi công nghệ JSF
* Tool Developer: người cung cấp các công cụ nhằm tạo ra công nghệ JSF xây dựng UI phía server dễ dàng hơn. Những thành viên chính của công nghệ JSF sẽ là page authors và application developers.



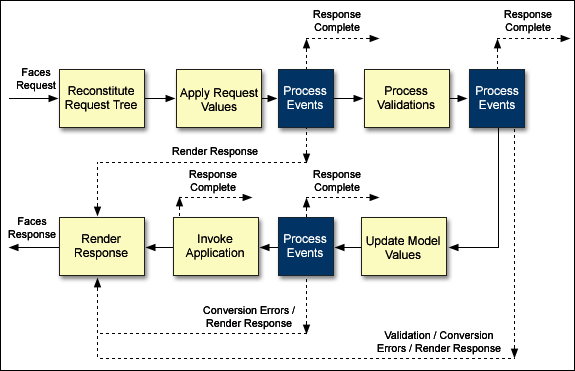
**3.6. Các thành phần chính**

Như hầu hết các công nghệ, Faces có những quy định nhằm đưa ra các khái niệm cơ bản cho những đặc điểm mà nó cung cấp. Đó là những elements như các component UI, validator và rederer. Có tám khái niệm khi phát triển các ứng dụng JSF

* UI Component (còn gọi là một control hay đơn giản là component) : một đối tượng có trạng thái, được chứa trên server, cung cấp các chức năng cụ thể để tương tác với một người dùng cuối. UI component là những JavaBean với các thuộc tính, phương thức, sự kiện. Chúng được tổ chức thành một cây các component thường hiển thị như một trang
* Rederer: Trả lời cho việc hiển thị một UI component và trao đổi một dữ liệu nhập của user vao giá trị của component. Renderer có thể được thiết kế để làm việc với một hoặc nhiều UI component, và một UI component có thể tập hợp với nhiều renderer khác nhau.
* Validator: Trả lời cho việc chắc chắn rằng giá trị nhập vào bởi user được chấp nhận. Một hoặc nhiều validator có thể được tập hợp với một UI component
* Backing beans: Các Java Bean xác định tập hợp các giá trị từ các UI component và bổ sung các phương thức listener cho event. Chúng cũng có thể nắm giữ các tham chiếu đến các UI component
* Converter: Chuyển đổi một giá trị của component thành và từ một chuỗi để hiển thị. Một UI component có thể được tập hợp với một converter duy nhất.
* Event/listener: JSF sử dụng mô hình event/listener JavaBeans (cũng được sử dụng cho Swing). UI component (và những đối tượng khác) tạo ra các event, và các listener có thể đăng ký để xử lý các sự kiện
* Messages: Thông tin hiển thị cho user. Chỉ bất kỳ phần ứng dụng nào (backing beans, validators, converter …) có thể tạo ra thông tin hoặc thông điệp lỗi nhằm hiển thị cho user
* Navigation: Khả năng di chuyển từ một trang đến trang khác. JSF có một hệ thống navigation mạnh mẽ tích hợp với những event listeners.

**3.7. Vòng đời của việc xử lý request**

Trong khi bạn tìm hiểu frameword bên dưới xử lý request của Servlet API, sẽ phân tích Faces xử lý request như thế nào. Điều này cho phép bạn xây dựng các ứng dụng tốt hơn bởi vì bạn sẽ biết chính xác cái gì được đặt ở đâu và khi nào.



* **Bước 1** - Restore View: Hiển thị thay thế tất cả các component tạo nên một trang cụ thể. Nó có thể lưu trữ trên client (thông thường trong một field ẩn trên trình duyệt) hoặc trên server (thông thường trong session)
* **Bước 2** - Apply Request Values: Mỗi UI component chấp nhận dữ liệu nhập có một giá trị được submit thay thế giá trị dữ liệu gốc từ user. Trong suốt bước này, framework ấn định giá trị được submit dựa trên tham số gởi đi trong request. Quá trình này gọi là decoding.
* **Bước 3** - Process Validation: trong bước này, JSF đặt cây component và hỏi mỗi component có chắc chắn rằng giá trị submit là có thể chấp nhận không. Bởi vì giá trị được submit của mỗi component nhập vào được cập nhật bởi bước 2, component bây giờ có hầu hết dữ liệu hiện thời của user. Trước khi validation xảy ra, giá trị được submit được chuyển đổi, bằng mỗi converter đã đăng ký cho component hoặc converter mặc định. Validation là sau khi xử lý trực tiếp bằng component hoặc công bố cho một hoặc nhiều validator.
* **Bước 4** - Update Model Values: Bây giờ chúng ta đã chắc chắn về giá trị cục bộ của các component đã được cập nhật và chính xác và đúng kiểu, có thể chuyển đi với bất kỳ bean nào được tập hợp hoặc mô hình các đối tượng. Bởi vì các đối tượng được tập hơp với các component thông qua các phát biểu JSF EL, đây là nơi các phát biểu này được kiểm tra và các thuộc tính được cập nhật dựa trên giá trị cụ bộ của component.
* **Bước 5** - Invoke Application: Bây giờ các bean cần thiết và mô hình các đối tượng được cập nhật, chúng ta có thể đi xuống việc công bố thông tin. Trong bước này, JSF quảng bá các sự kiện cho bước này đối với bất kỳ listener nào đã được đăng ký.
* **Bước 6** - Render Response: Tại thời điểm này, tất cả xử lý bới framework và ứng dụng đã trải qua. Tất cả chờ được gởi đi trả lời cho user, và đây là mục tiêu chính của bước này. Mục tiêu thứ hai là lưu lại trạng thái hiển thị để nó có thể được phục hồi trong bước Restore View nếu user yêu cầu lại. Trạng thái hiển thị lưu lại trong bước này bởi vì thường thì hiển thị được lưu trên client, vì thế nó là một phần của response nhằm gởi trả cho user. Trong trường hợp này, JSF đang lưu trạng thái trên server, vì thế hiển thị hầu hết được lưu trữ trong session của user.