



MÔ HÌNH MVC

MODEL VIEW CONTROLLER



1. Giới thiệu

- Tại sao phải có mô hình MVC?
- Một công việc được giao cho nhiều người làm theo từng công đoạn khác nhau.
- Như vậy công việc đó có tốt đẹp và khả quan không? Phải gọi là thúc đẩy làm nhanh hơn là chẳng khác.
- Chưa kể đến việc hư hỏng xảy ra, bộ phận nào làm kém chất lượng ở công đoạn nào thì bộ phận đó làm lại công đoạn đó, không ảnh hưởng đến công việc của những bộ phận khác.



1. Giới thiệu

- Mô hình MVC cũng vậy đây
- Đơn giản là chia công việc cho các phần khác nhau trong một ứng dụng.
- Các phần thực hiện mỗi công việc khác nhau, chỉ làm và trả về kết quả.
- Có lỗi phần nào thì sửa code phần đó, không có việc mở cả project lên để tìm lỗi.
- Tóm lại, dễ dễ dàng sửa chữa, gỡ rối, bảo trì và phát triển ứng dụng



1. Giới thiệu

- Mô hình **MVC** được sử dụng lần đầu tiên trong Smalltalk, sau đó được sử dụng phổ biến trong ngôn ngữ lập trình Java.
- **Smalltalk** là ngôn ngữ lập trình hướng đối tượng, kiểu dữ liệu động và có tính phản xạ. **Smalltalk** được thiết kế nhằm bảo vệ tối đa các đặc điểm ngắn gọn và trong sáng của nó và không hướng đến tính khả dụng hay tính hiệu năng.
- Mô hình MVC hiện nay rất phổ biến trong các framework PHP



1. Giới thiệu

- Ý nghĩa của mô hình nhằm tách biệt hoàn toàn phần lưu trữ và xử lý dữ liệu (**Model**) với thành phần trình bày kết quả cho người dùng (**View**)
- Cho phép người lập trình có thể tách biệt công việc trong **quá trình xây dựng chức năng của ứng dụng** và **quá trình xây dựng giao diện** cho người dùng.



1. Giới thiệu

- Việc thay đổi thành phần của dữ liệu (model) sẽ không ảnh hưởng nhiều đến giao diện của người dùng
- Mô hình đưa ra Model để không cho người dùng thao tác trực tiếp vào dữ liệu vật lý (cơ sở dữ liệu hay là tập tin) mà phải thông qua Model
- Vì vậy cho dù dữ liệu vật lý thay đổi cấu trúc nhưng cấu trúc Model cho việc truy cập, xử lý, lưu trữ dữ liệu sẽ không bị ảnh hưởng



1. Giới thiệu

- Một user A muốn xem các phim chọn lọc được lựa chọn từ các đĩa CD/DVD
- Để đọc được các đĩa CD/DVD thì phải dùng đầu máy đọc đĩa
- Đầu máy này đặc biệt không có nút điều khiển trực tiếp, đầu máy này nhận hàng loạt các đĩa
- Do vậy muốn kích hoạt và chọn đĩa xem, người dùng A bắt buộc phải sử dụng remote control – điều khiển từ xa

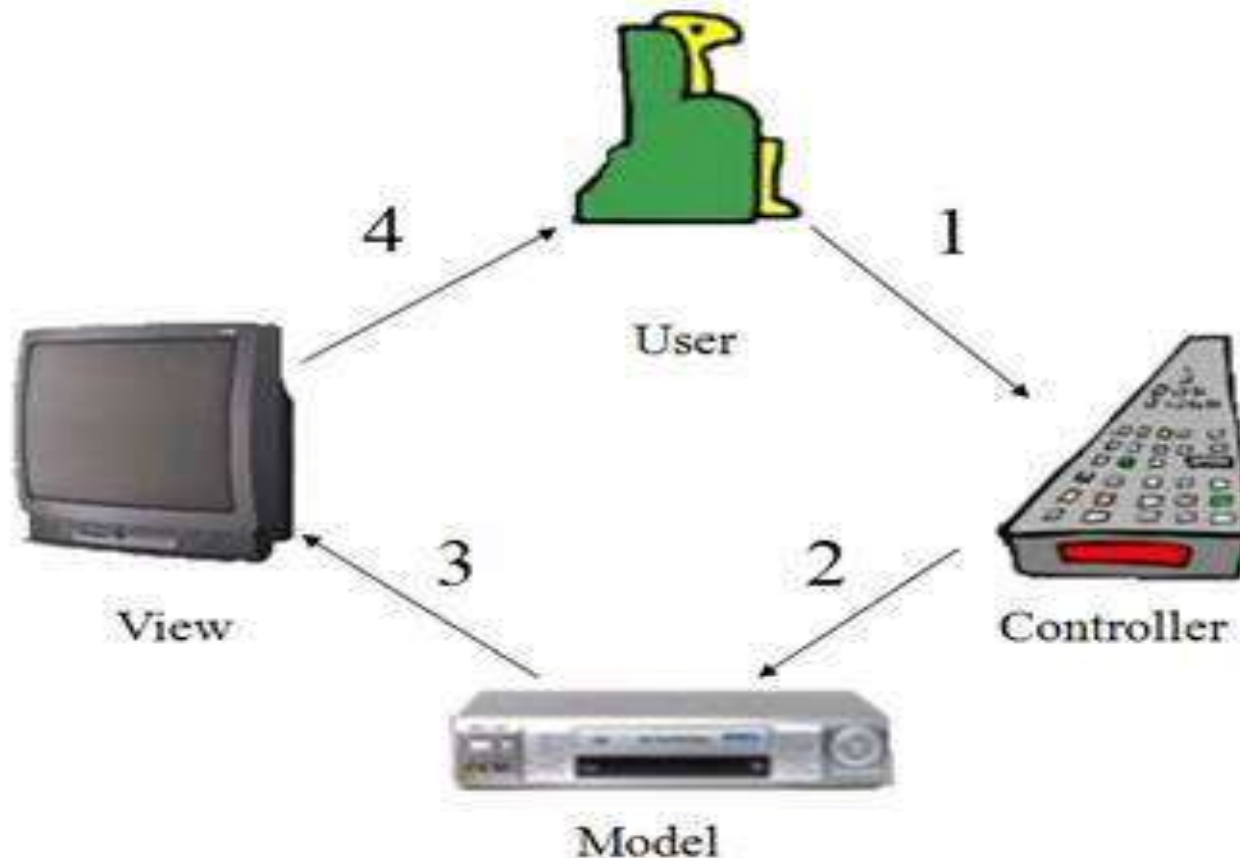


1. Giới thiệu

- Như vậy, chúng ta sẽ nhận thấy cách thức A xem phim như sau :
 - Người dùng A sẽ sử dụng remote control điều khiển đầu máy để chọn đĩa mà mình thích xem nhất để mở nó ra
 - Đĩa được chọn từ đầu máy được đọc và trình chiếu trên màn hình TV
 - Khi dữ liệu đã hiển thị trên màn hình TV thì người dùng A sẽ thấy được nội dung
 - Cách thức này sẽ được lặp đi lặp lại khi người dùng A chọn một phím khác, vẫn dùng bộ điều khiển để chọn đĩa và đầu đĩa sẽ đưa dữ liệu lên màn hình TV

1. Giới thiệu

- Xét ví dụ như sau:





1. Giới thiệu

- Chúng ta nhận thấy rằng:
 - Đầu máy là nơi xử lý dữ liệu, chọn lựa cách thức xử lý, nội dung cần thiết – Model
 - Màn hình TV chỉ làm nhiệm vụ duy nhất để trình bày kết quả mà đầu máy – Model đã thực hiện, được lựa chọn. Màn hình TV không thể lựa chọn và không có cách chọn lựa là trình bày các thành phần truyền đến đã được xử lý. Màn hình TV là View
 - Thành phần hỗ trợ đưa dữ liệu từ Model đến View đó là bộ điều khiển, ngoài ra bộ điều khiển cũng là nơi kết nối người dùng với đầu máy với màn hình TV.
 - Chức năng của điều khiển là chọn đúng model để đưa ra view. Bộ điều khiển – remote control là một Controller

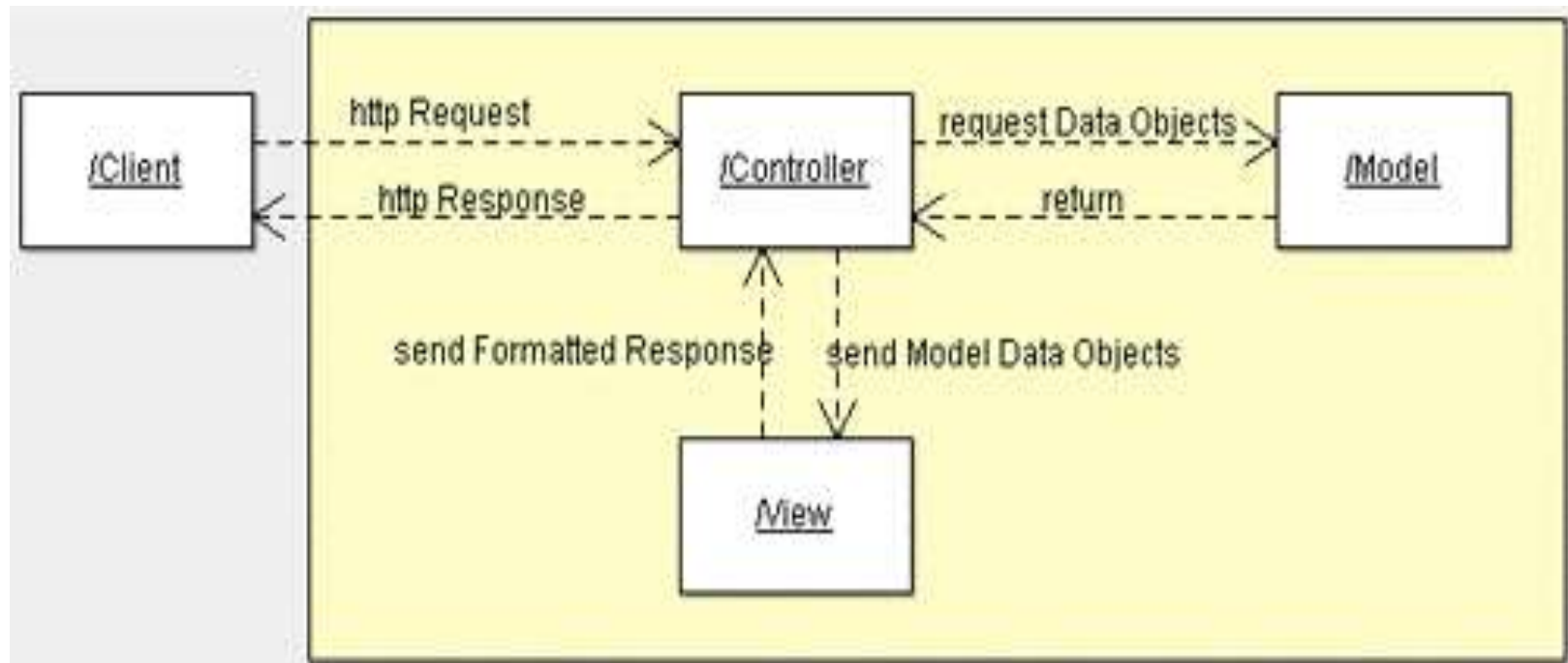


1. Giới thiệu

- Mô hình MVC là mô hình gồm 3 lớp: Model, View, Controller.
 - **Model:** Lớp này chịu trách nhiệm quản lí dữ liệu: giao tiếp với cơ sở dữ liệu, chịu trách nhiệm lưu trữ hoặc truy vấn dữ liệu.
 - **View:** Lớp này chính là giao diện của ứng dụng, chịu trách nhiệm biểu diễn dữ liệu của ứng dụng thành các dạng nhìn thấy được.
 - **Controller:** Lớp này đóng vai trò quản lí và điều phối luồng hoạt động của ứng dụng. Nhận request từ client, điều phối các Model và View để có thể cho ra output thích hợp và trả kết quả về cho người dùng.

1. Giới thiệu

- Ta có thể mô tả lại hoạt động của mô hình MVC thông qua sơ đồ sau:





2. MÔ HÌNH MVC

■ Model

Là các thành phần hỗ trợ ánh xạ dữ liệu vật lý lên bộ nhớ, lưu trữ dữ liệu tạm thời trên bộ nhớ, **hỗ trợ các cách thức xử lý dữ liệu, hỗ trợ khả năng giao tiếp và trao đổi dữ liệu giữa các đối tượng khác trong bộ nhớ và cơ sở dữ liệu**

Cụ thể là một đối tượng Object trong khái niệm của lập trình hướng đối tượng OOP và mang đầy đủ khái niệm và tính chất của một Object

Trong ứng dụng Web của Java, **Model sẽ là JavaBean hay Enterprise JavaBean hay Web Service**



2. MÔ HÌNH MVC

■ View

Là thành phần hỗ trợ trình bày dữ liệu hay kết quả ra màn hình, hỗ trợ nhập thông tin từ phía người dùng

Các thành phần này có khả năng truy cập Model, truy xuất Model thông qua những hành vi mà Model cho phép nhưng View không thể thay đổi các thành phần trong Model

Trong mô hình ứng dụng Web thì **html**, **servlet**, **jsp** ... là những thành phần đại diện cho View



2. MÔ HÌNH MVC

■ Controller

Là các thành phần hỗ trợ kết nối người dùng server, đón nhận yêu cầu người dùng, thực hiện chuyển xử lý, lựa chọn và cập nhật model và view tương ứng để trình bày về phía người dùng

Hỗ trợ kết nối giữa model và view, giúp model xác định được view trình bày

Trong mô hình ứng dụng Web thì Servlet đóng vai trò của Controller



2. MÔ HÌNH MVC

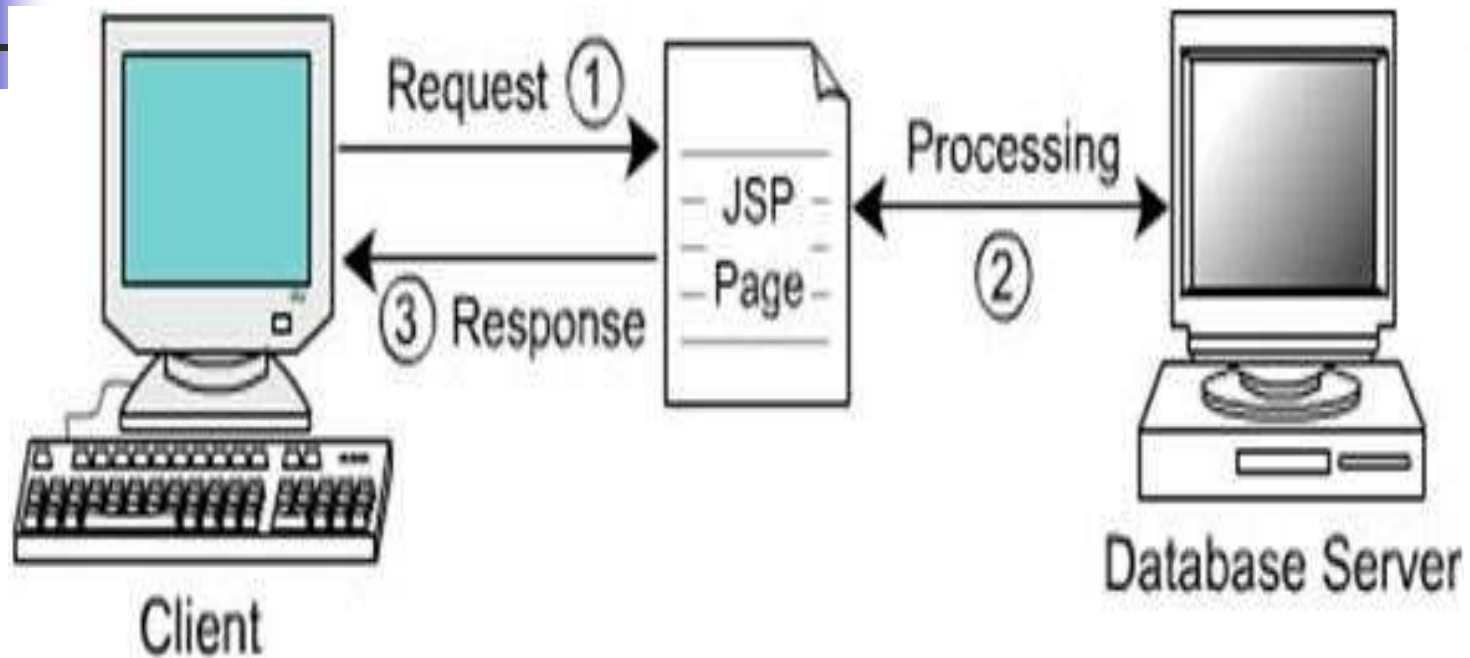
- Trong mô hình ứng dụng web thì:
 - **Model** sẽ là JavaBean, Enterprise JavaBean hay Web Service.
 - Các trang html, servlet, jsp,... là những thành phần đại diện cho **View**.
 - Cuối cùng, servlet đóng vai trò là **Controller**.
- Servlet có thể là view hoặc controller tùy vào từng trường hợp cụ thể



2. MÔ HÌNH MVC

- Trước đây khi lập trình một trang web(jsp, php, asp,..) ta có thể xen các mã html cùng với các mã jsp, php, hay asp. Do vậy, có những khó khăn có thể gặp phải:
 - Người thiết kế giao diện cũng cần phải biết các ngôn ngữ lập trình, hoặc bạn phải trực tiếp thiết kế chúng.
 - Việc bảo trì chúng thường rất khó khăn, vì một phần các mã chương trình lẫn lộn với mã html.
 - Khi có lỗi xảy ra, việc tìm ra lỗi và định vị lỗi cũng là một vấn đề khó khăn.

2. MÔ HÌNH MVC



Mô hình 1 : Mô hình lập trình cổ điển



2. MÔ HÌNH MVC

- Mô hình MVC1:

- Là mô hình tương tự như html nhưng các trang web ở dạng động có thể đón nhận và trình bày dữ liệu từ server
- Nhưng tất cả các trang liên kết đều là các đường dẫn tĩnh và các cách thức xử lý đều thực hiện trực tiếp trên trang. Ngoài ra, các trang thực hiện gọi trực tiếp lẫn nhau.
- Mô hình này chỉ phù hợp với ứng dụng nhỏ vì các đường dẫn rất khó để tìm kiếm và sửa đổi, đặc biệt trên trang đang trộn lẫn giữa code html, javascript, xml, javacode ...



2. MÔ HÌNH MVC

- Để khắc phục các khó khăn trên, người ta đưa ra mô hình 2 hay còn gọi là mô hình MVC (Model-View-Controller).
- Tương ứng với một trang JSP, ta có thể tách ra làm ba thành phần:
 - Mô hình (Model)
 - Khung nhìn (View)
 - Bộ điều khiển (Controller)



2. MÔ HÌNH MVC

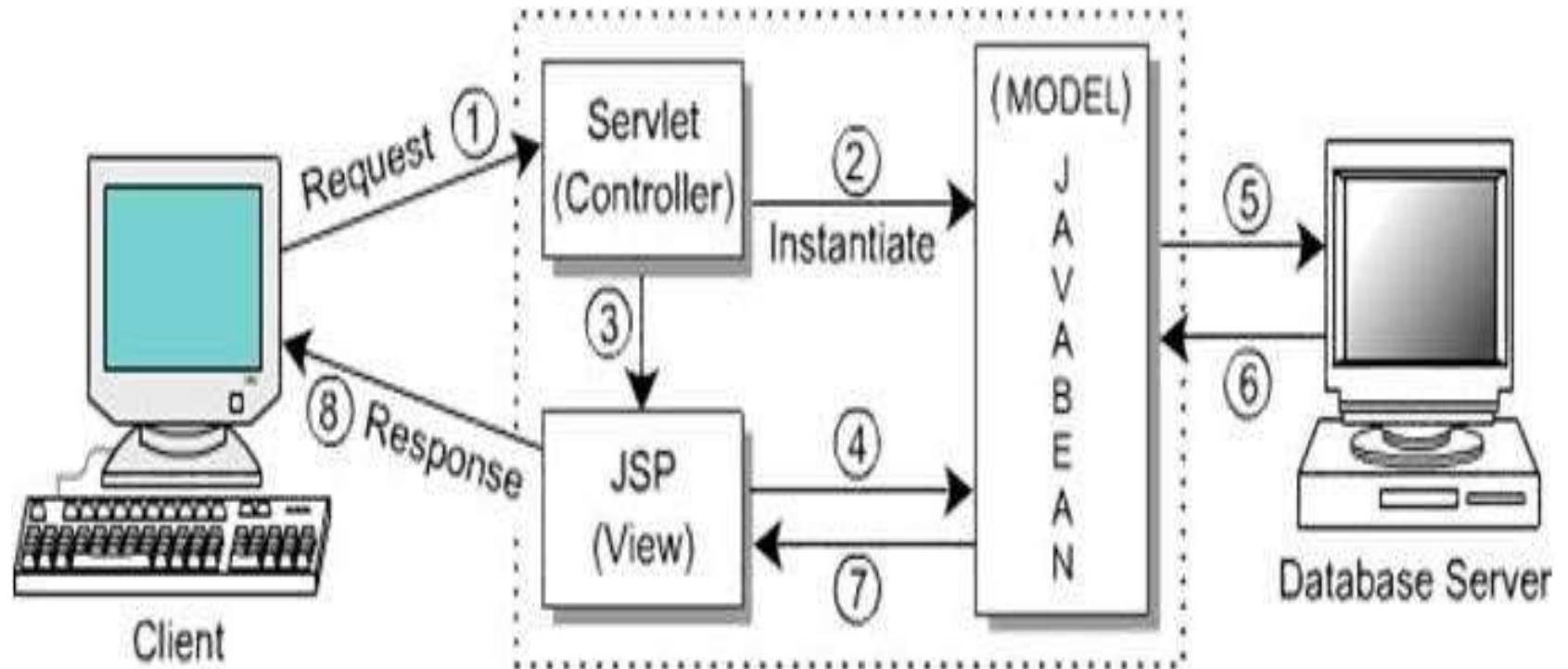
- Mô hình MVC2:
 - Đây là mô hình thực tế áp dụng theo hướng tiếp cận hiện nay
 - Tách biệt riêng từng thành phần, tạo nên sự uyển chuyển khi vận dụng và cài đặt, đặc biệt là bảo trì nâng cấp



2. MÔ HÌNH MVC

- Như vậy, chúng ta có thể tách biệt được các mã java ra khỏi mã html. Do vậy, nó đã giải quyết được các khó khăn đã nêu ra trong Mô hình 1.
- Người thiết kế giao diện và người lập trình java có thể mang tính chất độc lập tương đối.
- Việc debug hay bảo trì sẽ dễ dàng hơn, việc thay đổi các theme của trang web cũng dễ dàng hơn ...

2. MÔ HÌNH MVC





2. MÔ HÌNH MVC

■ Các thành phần trên làm việc như sau:

- **Mô hình (Model):** Mô hình là các lớp java có nhiệm vụ:
 - Nhận các yêu cầu từ khung nhìn
 - Thi hành các yêu cầu đó (tính toán, kết nối CSDL ...)
 - Trả về các giá trị tính toán cho View.
- **Khung nhìn (View):** Bao gồm các mã tương tự như JSP để hiển thị form nhập liệu, các kết quả trả về từ Mô hình...
- **Bộ điều khiển (Controller):** Đồng bộ hoá giữa Khung nhìn và Mô hình. Tức là với một trang JSP này thì sẽ tương ứng với lớp java nào để xử lý nó và ngược lại, kết quả sẽ trả về trang jsp nào.



2. MÔ HÌNH MVC

- Cơ chế thực hiện (MVC2):

Web Browser gửi request đến server thông qua các control trên form HTML hay JSP, hay query string url hay qua cookies.

Servlet – Controller đón nhận request và xác định Model tương ứng để tạo ra instance của JavaBean để đón nhận các giá trị nhập từ request để lưu trữ và xử lý

Model thực hiện xử lý, kết nối dữ liệu vật lý dưới DBMS (nếu có) và trả kết quả trả về cho Controller

Kết quả xử lý được chuyển vào Servlet – Controller, Servlet Controller thực hiện tạo hay lựa chọn View để từ đó đưa kết quả xử lý hay dữ liệu lấy từ Model để cập nhật lại trang kết quả View.

Controller gửi View qua response cho người dùng để browser có thể trình bày dữ liệu trong Web Browser



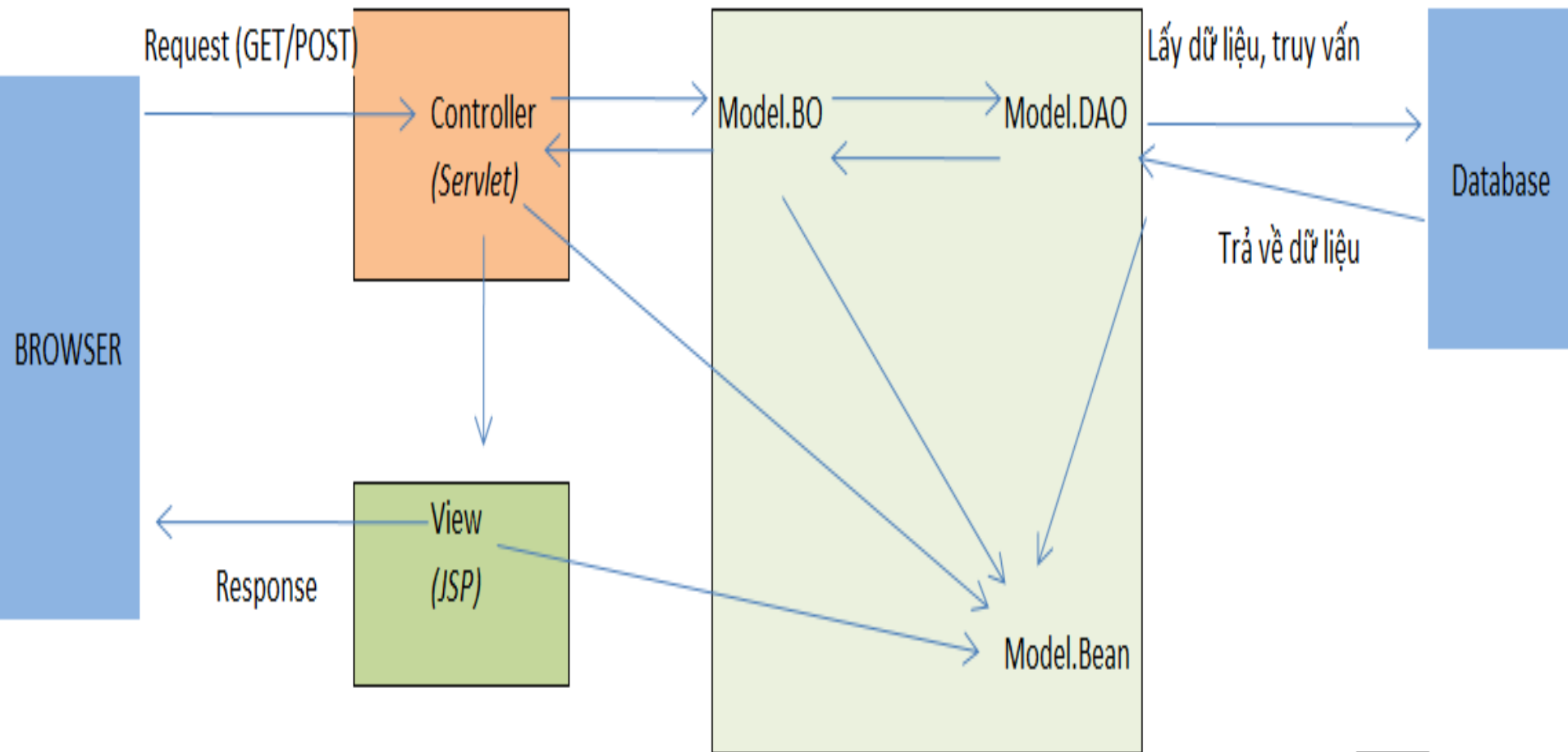
2. MÔ HÌNH MVC

- Cơ chế thực hiện trên cho thấy mọi tập trung xử lý và kết xuất đều hướng vào Controller

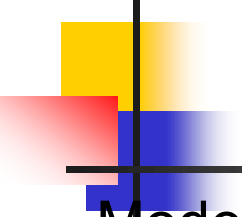
Do vậy, đây cũng là một phần khuyết điểm khi Controller là nơi tập trung xử lý dữ liệu

- Do vậy, một trong những khái niệm để giảm bớt tải của Controller chính là Filter

SƠ ĐỒ HOẠT ĐỘNG CỦA MÔ HÌNH MVC



Nhiệm Vụ Các Tầng Của Mô Hình MVC

- 
- Model: Tách riêng phần logic và phần hiển thị
 - Model.Bean: Chứa các thực thể, gồm dữ liệu (private), kèm theo các phương thức set/get
 - Model.Dao: Thực hiện các công việc liên quan đến cơ sở dữ liệu
 - Kết nối, lấy dữ liệu, truy vấn, chỉnh sửa, xóa dữ liệu trực tiếp với database
 - Model.Bo: Truyền yêu cầu từ Servlet chuyển qua DAO, lấy dữ liệu từ DAO về cho Servlet. Ngoài ra còn xử lý các yêu cầu nghiệp vụ
 - View: Là các trang JSP, trả kết quả hiển thị cho người dùng
 - Controller: Là các Servlet, có nhiệm vụ nhận yêu cầu từ người dùng, đưa yêu cầu và nhận dữ liệu từ tầng Model, từ đó chuyển hướng và trả về cho View



Mối Quan Hệ Giữa Các Thành Phần

- Mối quan hệ giữa Controller và Model

Ý tưởng đó là khi chúng ta thay đổi đầu máy đọc, bộ điều khiển từ xa có thay đổi theo hay không? Hay chúng ta vẫn dùng cái điều khiển cũ (không tính điều khiển đa năng)? Bộ điều khiển được thiết kế dựa trên đầu máy hay đầu máy được thiết kế dựa trên đồ điều khiển?

Đó là controller phụ thuộc vào model bởi vì controller được thiết kế để kết nối với model, điều khiển, truy xuất model.

Do vậy, model có sự thay đổi là controller phải thay đổi theo để có thể truy xuất và điều khiển model cho phù hợp



Mối Quan Hệ Giữa Các Thành Phần

- **Mối quan hệ giữa Model và View**

View lệ thuộc vào Model bởi vì khi các tổ chức interface của Model thay đổi nghĩa là View phải thay đổi theo

Chúng ta tránh nhầm lẫn về khái niệm ở trên là tách biệt giữa View và Model là có lợi nhưng tại sao ở đây View lại lệ thuộc Model?

Khái niệm của chúng ta ở đây đó là tách biệt dữ liệu và thành phần xử lý bên trong, hệ thống sẽ trở nên uyển chuyển khi chúng ta thay đổi thành phần cài đặt trong Model, không phải thay đổi interface Model.

Và khái niệm trên đã nêu rõ giữa dữ liệu vật lý và dữ liệu được chứa trong Model



Mối Quan Hệ Giữa Các Thành Phần

- Mỗi quan hệ giữa View và Controller
 - Controller sẽ là đối tượng lựa chọn View
 - Tùy thuộc theo việc yêu cầu sử dụng của người dùng
 - Tùy thuộc theo kết quả xử lý của Model



2. MÔ HÌNH MVC

- Khi một ứng dụng có rất nhiều Model và nhiều View. Vì vậy, mô hình cần có một thành phần lựa chọn và kết nối các thành phần này lại với nhau theo cách hiệu quả nhất.
- Controller là một trong những đối tượng đưa ra để đón nhận yêu cầu nhập xuất từ người dùng, xác định model tương ứng với view để chuyển cho model xử lý
- Kết quả xử lý của model sẽ được chuyển đến controller để controller xác định view kết xuất nhằm đưa kết quả xử lý và hiển thị cho người dùng



2. MÔ HÌNH MVC

■ Mô tả thành phần Controller - Servlet

- Servlet là đoạn chương trình java thực thi trên Web Server hỗ trợ người lập trình Java xây dựng trang web động mà không cần học ngôn ngữ lập trình web mới
- Servlets nhận request – yêu cầu từ client, sau đó thực hiện các yêu cầu xử lý để gửi response – phản hồi đến người dùng sử dụng HTTP
- Servlet được load sẵn ở Web Server duy nhất lần đầu tiên khi ứng dụng được deploy và đáp ứng tức thời yêu cầu của người dùng thông qua Web Container.
- Servlet được server hỗ trợ cơ chế multithread giúp giảm tài nguyên và quá tải trong việc xử lý của server hay container



2. MÔ HÌNH MVC

- Mô tả thành phần Controller - Servlet

- Khuyết điểm

Không thích hợp cho việc trình bày và xử lý giao diện vì code html được viết trong chuỗi String của các câu lệnh Java, rất khó trong việc checking và kiểm tra lỗi về tính đúng đắn của văn bản xml (**well-form**)

Không hỗ trợ đầy đủ các thành phần liên quan đến session

Về phía người dùng có thể nói tương tác chỉ là single thread vì người dùng không thể xác định instance servlet phục vụ cho mình

Container và server tự động xác định instance tương ứng và yêu cầu nó xử lý.

Chúng ta chỉ biết được xử lý khi thấy kết quả được hiển thị ở browser. Nghĩa là mọi thứ xử lý phải lệ thuộc container



2. MÔ HÌNH MVC

- Mô tả thành phần Controller - Servlet

- Cơ chế hoạt động

Khi có request từ client gửi đến Server hay Web Container Container sẽ lựa chọn một instance Servlet tương ứng để đáp ứng request đó (người dùng sẽ không bao giờ biết instance nào được lựa chọn, nó lựa chọn khi nào, servlet xử lý khi nào)

Servlet lựa chọn sẽ thực hiện xử lý và kết nối DB nếu cần

Sau khi servlet thực hiện xong, sẽ gửi kết quả ra container để gửi response về cho người dùng

Browser đón nhận kết quả và trình bày ra màn hình dữ liệu



2. MÔ HÌNH MVC

- Mô tả thành phần Controller - Servlet
 - Chu kỳ sống hay tồn tại của servlet trong server hay container

Khi một ứng dụng chưa được deploy vào trong server thì servlet chưa được khởi tạo (**uninstantiated**)

Khi ứng dụng được deploy vào server thì container sẽ thực hiện khởi tạo instance cho servlet, trong lúc khởi tạo kích hoạt hàm init. **Lưu ý, hàm init chỉ được kích hoạt lần duy nhất khi deploy**

Khi có một request đến servlet của người dùng, container đón nhận request và chọn instance bean bất kỳ tương ứng với yêu cầu để đáp ứng.



2. MÔ HÌNH MVC

- Mô tả thành phần Controller - Servlet
 - Chu kỳ sống hay tồn tại của servlet trong server hay container

Sau khi chọn được bean, container sẽ kích hoạt hàm service tương ứng. **Lưu ý ở đây, ở lần yêu cầu thứ 2 thì container cũng sẽ chọn bean và kích hoạt hàm service tương ứng, do vậy, hàm service sẽ được gọi rất nhiều lần**

Khi servlet được cập nhật mới hay server bị crash hay undeploy ứng dụng ra khỏi server – container, thì hàm destroy của servlet được kích hoạt. **Nghĩa là servlet chỉ kích hoạt phương thức destroy lần cuối cùng duy nhất**



2. MÔ HÌNH MVC

• Mô tả thành phần Controller – Servlet

Servlet định nghĩa 3 tầm vực thao tác: request, session, ServletContext

Đây là vùng không gian bộ nhớ (memory segment) được cung cấp cho mỗi ứng dụng web dùng để chứa các thông tin để giao tiếp với các thành phần khác trong server

Mỗi vùng không gian này tồn tại trong một khoảng thời gian nhất định tùy theo qui định

- request: tồn tại từ lúc gửi request cho đến khi response
- session: một khoảng thời gian từ lúc mở trình duyệt đến đóng trình duyệt, hết thời gian session, session bị hủy, ...
- ServletContext: có thể gọi là application tồn tại từ lúc bắt đầu ứng dụng đến khi ứng dụng bị undeploy ra khỏi server hay server bị crash



2. MÔ HÌNH MVC

- Mô tả thành phần Controller - Servlet
 - Servlet cung cấp thêm một interface `RequestDispatcher` để hỗ trợ việc giao tiếp và xác định view tương ứng trong xử lý
 - `RequestDispatcher` hỗ trợ container chuyển request object** từ đối tượng của server từ thành phần này sang thành phần khác (đây là ưu điểm vượt trội so với **`response.sendRedirect` hay click một link trên trang web vì 02 đối tượng này không truyền object request đi**)
 - Đối tượng cuối cùng là đối tượng sẽ reponse kết quả trả về hay cho phép nhúng đối tượng này sang đối tượng khác
 - Cơ chế này còn giúp che dấu thông tin xử lý của các đối tượng xử lý trên thanh url của trình duyệt – đảm bảo tính bảo mật cao



2. MÔ HÌNH MVC

Mô tả thành phần Model:

- Một thành phần cấu thành object và chứa đầy đủ đặc tính của object đó là

Một object bao gồm state (trạng thái) và behaviors (các hành vi)

Đảm bảo 4 tính chất

- abstraction (mang tính chất chung nhất của object)
- encapsulation (cho phép người dùng truy cập các trạng thái của object thông qua các behavior)
- inheritance (có tính kế thừa)
- modularity (phân chia module theo từng nhóm chức năng và tách biệt các thành phần theo dạng component để có thể dễ dàng cài đặt, bảo trì và tái sử dụng)



2. MÔ HÌNH MVC

- Mô tả thành phần Model

- **JavaBeans** là một đối tượng đại diện cho object và được sử dụng như **Model** bởi vì nó chứa đầy đủ các yêu cầu đã nêu trên
- Đặc điểm của JavaBeans

JavaBeans là một java class được implements từ Serializable

Đây là một object sử dụng qua protocol và để giao tiếp với các thành phần trong và ngoài server, do vậy nó phải được chuyển đổi từ thành dạng byte stream để dễ dàng truyền đi

JavaBeans bắt buộc phải được cài đặt có package để có thể tái sử dụng thông qua lệnh import.



2. MÔ HÌNH MVC

Mô tả thành phần Model

- Đặc điểm của JavaBeans

Các thuộc tính properties trong JavaBeans bắt buộc phải được khai báo là *private*

Việc khai thác các thuộc tính này phải được thông qua các hàm `getTênThuộcTÍNH` hay `setTênThuộcTÍNH`

- Hàm *get* sẽ được đổi thành hàm *is* nếu kiểu dữ liệu là kiểu boolean

Bắt buộc phải có một constructor không tham số để có thể khởi tạo object mà không cần khởi tạo giá trị ban đầu cho object luôn ở trạng thái đảm bảo thao tác không bị lỗi kể cả khi giá trị thuộc tính của các object chưa cập nhật gì cả



2. MÔ HÌNH MVC

■ Mô tả thành phần View

- Những đối tượng có khả năng trình bày dữ liệu ra màn hình như html, jsp ...
- JSP tích hợp bao gồm HTML, XML, Java Code, và kể cả Servlet

Bản chất của JSP là Servlet, do vậy các thành phần của Servlet sẽ có tồn tại hết trên JSP

Ngoài ra, JSP không cần phải biên dịch mà nó được biên dịch khi có request lần đầu tiên yêu cầu đến server, do vậy JSP khắc phục nhược điểm chỉnh sửa phải cần có source code và biên dịch lại khi deploy sau khi chỉnh sửa của Servlet

JSP cung cấp các thành phần implicit Object để người dùng có thể sử dụng các thành phần tương tác trên server mà không cần khai báo và khởi tạo

Kết xuất của JSP thực chất là HTML



2. MÔ HÌNH MVC

- Package “service” có thể đổi tên thành “**bo**”.
 - **Service** là dịch vụ, chuyên chứa các class xử lý business logic, các đối tượng được tạo từ các class đó gọi là **business object**.
 - Do vậy có thể viết tắt thành **bo** để đặt tên cho package
- Các package như **dao**, **service (bo)** lại được chứa trong package **model**.
- Các class nhận nhiệm vụ ánh xạ tới table trong database (hay gọi với tên **Java Bean**) được chứa trong model thì giờ lại chuyển vào trong package **bean** (package bean **chứa vào** package model luôn)



CÁC CHÚ Ý CỦA MÔ HÌNH MVC

- Bean chứa các thực thể, có thể sử dụng ở bất cứ nơi nào cần thiết: DAO, BO, JSP, Servlet, Bean
- Các kết nối đến Database chỉ thực hiện ở DAO, các tầng khác không liên quan đến database
- DAO chỉ cho phép gọi từ BO, các nơi khác không được gọi DAO
- BO chỉ cho phép gọi từ Controller (servlet), các nơi khác không được gọi BO
- Servlet chứa các nhiệm vụ nhận dữ liệu, trả dữ liệu và điều hướng. Cho nên các xử lý nghiệp vụ xử lý ở BO
- JSP chỉ nhận dữ liệu từ Servlet, không được sử dụng BO và DAO để lấy dữ liệu
- Không được trực tiếp gọi trang JSP, mà phải thông qua Servlet



Hết