**Bài 11 Ki*ế*n trúc MVC**

Module: JWBD

# Kiểm tra bàitrước

Hỏivàtraođổivềcáckhókhăngặpphảitrongbài“JSP & JSTL"

Tómtắtlạicácphầnđãhọctừbài“JSP & JSTL”

## Mụctiêu

* TrìnhbàyđượckiếntrúcMVC
* Trìnhbàyđượckiếntrúcphântầng
* TrìnhbàyđượccáclợiíchcủakiếntrúcphântầngvàkiếntrúcMVC
* Triểnkhaiđượccácứngdụngtheokiếntrúcphântầng
* TriểnkhaiđượccácứngdụngtheokiếntrúcMVC

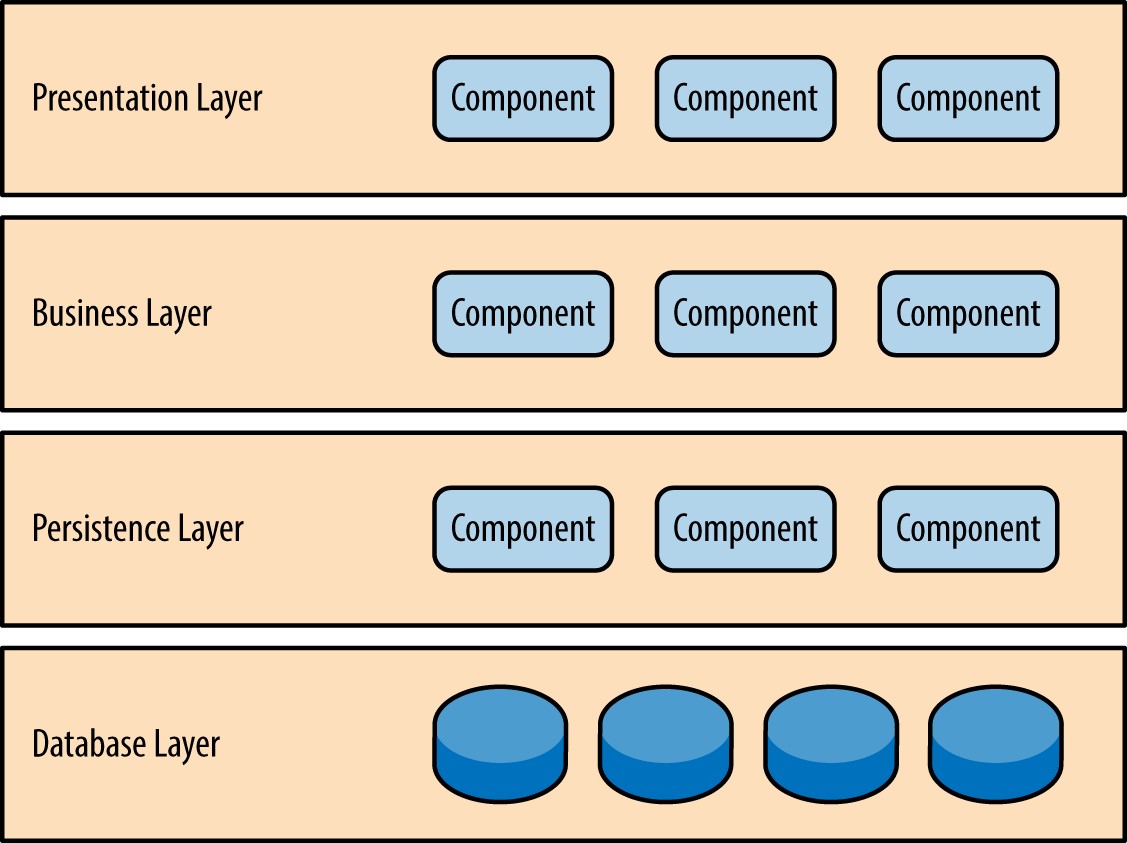
3

## Kiến trúc phântầng

* Kiến trúc phân tầng (multilayered/multitier/n-tier/architecture) là một kiến trúc phần mềm trong đó các thành phần của hệ thống được tổ chức theo cáctầngtheo chiều ngang,mỗitầng thựchiện mộtnhómnhiệmvụcụthể
* Chẳnghạn:Tầnggiaodiện,tầngđiềukhiển,tầngdịchvụ,tầnglưu trữdữliệu…
* Dạngkiếntrúcn-tierphổbiếnlà3tầng(three-tier)

4

## Kiến trúc phân tầng: Vídụ

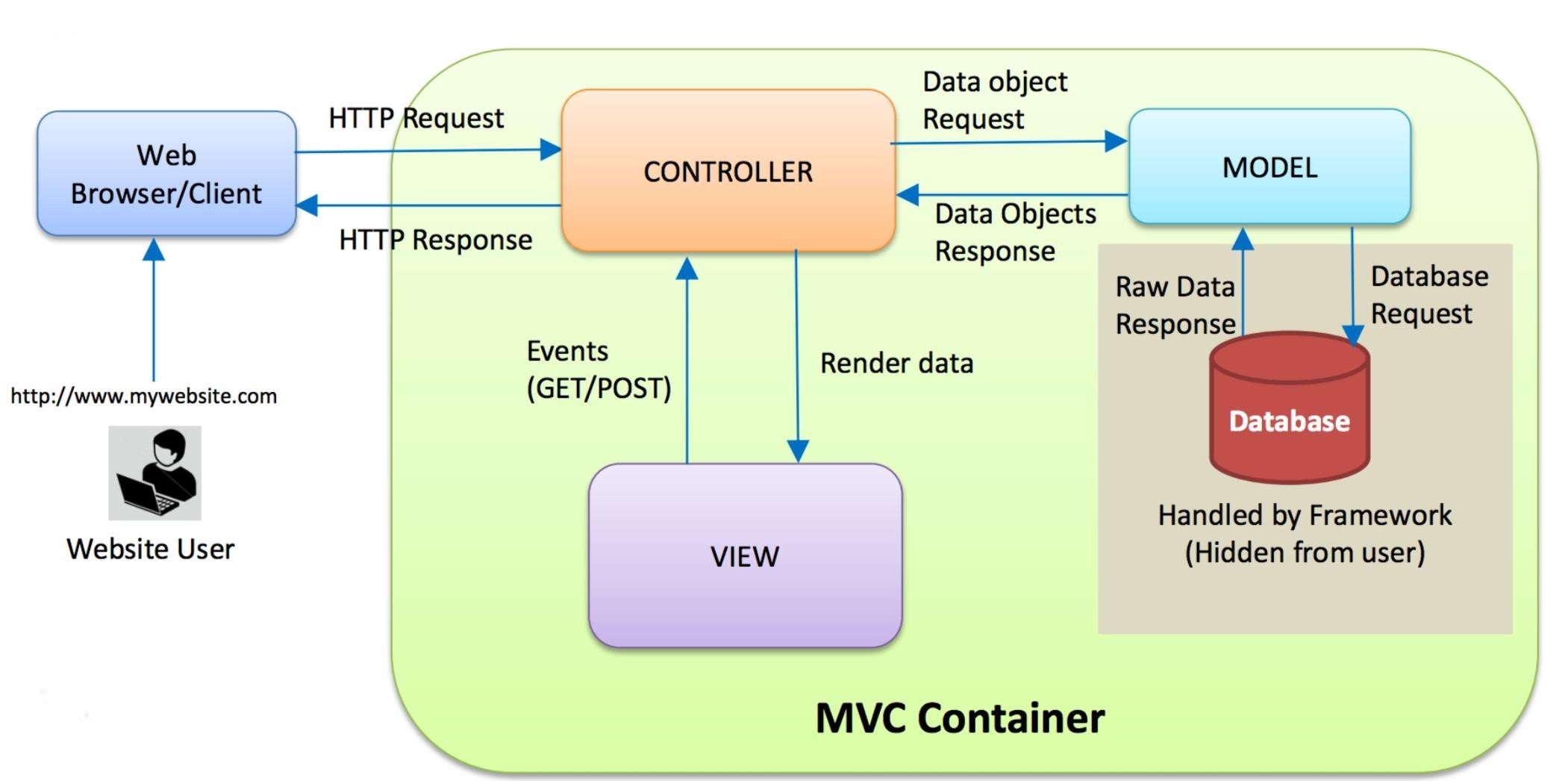


## Kiến trúcMVC

* MVClàviếttắtcủaModel-View-Controller
* Kiến trúc MVC tổ chức các thành phần của hệ thống vào 3 tầng riêngbiệtvàcókếtnốiđếnnhau:
* TầngModel:Biểudiễndữliệuvàcáclogicnghiệpvụ
* TầngView:Hiểnthịdữliệuvàlàgiaodiệntươngtácvớingườidùng
* Tầng Controller: Xử lý các thao tác từ người dùng, kết nối giữa Model và View

6

## MVC làgì



### Lợi ích của kiếntrúc phân tầng

* Tách các trọng tâm của từng thành phần trong ứng dụng

(separation ofconcerns)

* Các thành phần trong một tầng thì chỉ thực hiện các nhiệm vụ liên quan đến tầngđó
* Dễ pháttriển
* Dễ kiểmthử
* Dễ quảnlý
* Dễ bảotrì

### Lợi ích của kiếntrúc MVC

* Dễ tái sửdụng
* Dễ mởrộng
* Táchphầnviewvàphầnnghiệpvụriêngbiệt
* ChophépcácLậptrìnhviênlàmviệctrêncácthànhphầnkhác nhautrongcùngmộtthờiđiểm
* Dễ bảotrì

# DEMOMVC

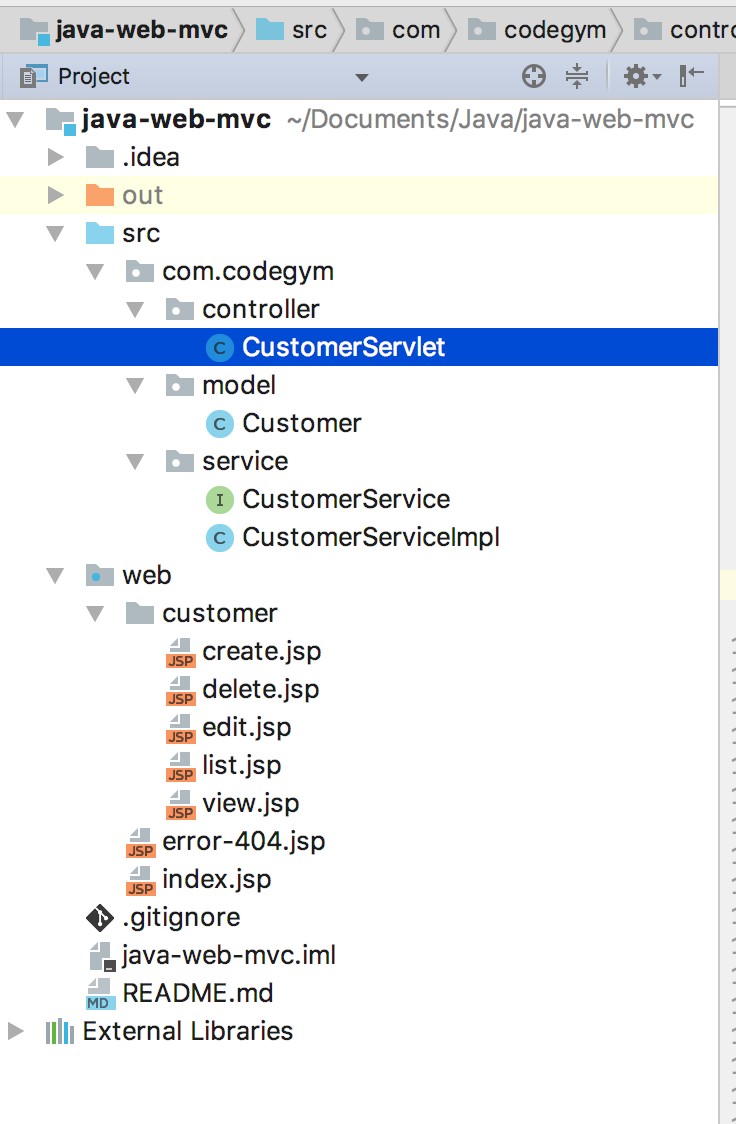
Ví dụ demoModel-View-Controller

## Bài toán quản lý danh sách kháchhàng

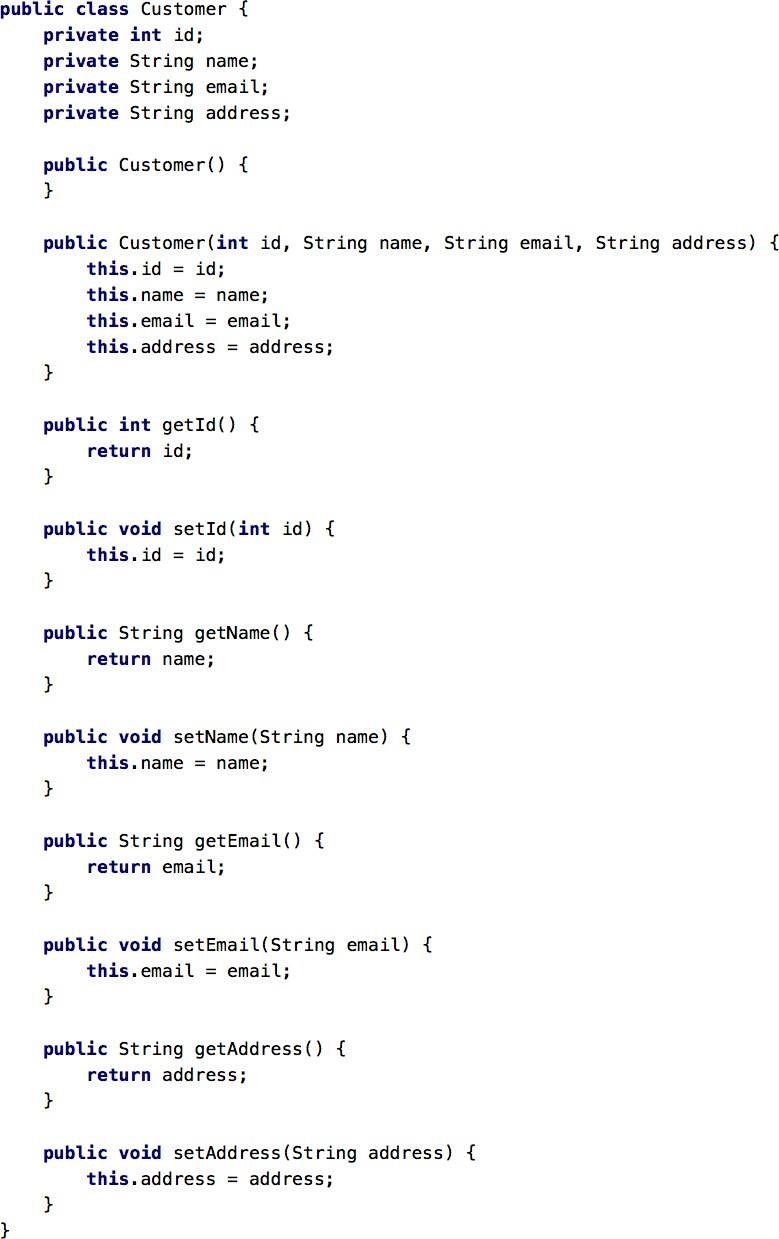
* Pháttriểnmộtứngdụngquảnlýkháchhàng.Ứngdụngcócácchức năng chínhsau:
* Hiểnthịdanhsáchkháchhàng
* Thêm một khách hàngmới
* Sửađổithông tincủakháchhàng
* Xoá một kháchhàng
* Xemthôngtinchitiếtcủakháchhàng

### Bước 1: Tạo dự án mới vớitên java-web-mvc

* Cấutrúccủacácthưmụcvàfilecủaứngdụngsaukhihoànthànhsẽ nhưsau:



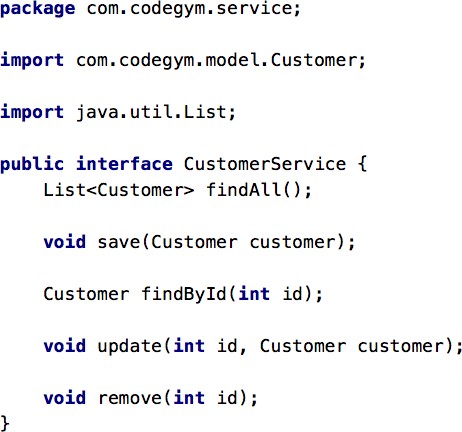
**Bước 2: Tạo Customermodel**

* Lớp com.codegym.model.Customer baogồmcácthuộctính:
* id: Id của kháchhàng
* name: Tên của kháchhàng
* email:Emailcủakháchhàng
* address:Địachỉcủakháchhàng

### Bước 3: Tạo interfaceCustomerService

* InterfaceCustomerServicelàinterfaceđịnhnghĩacácphươngthứcở tầngServiceđểthaotácvớiđốitượngCustomer.
* InterfaceCustomerServicebaogồmcácphươngthức:
* findAll():Trảvềdanhsáchtấtcảkháchhàng
* save(): Lưu một kháchhàng
* findById():TìmmộtkháchhàngtheoId
* update():Cậpnhậtthôngtincủamộtkháchhàng
* remove():Xoámộtkháchhàngkhỏidanhsách

### c 3: Tạo interface CustomerService



### c 4: Tạo lớpCustomerServiceImpl

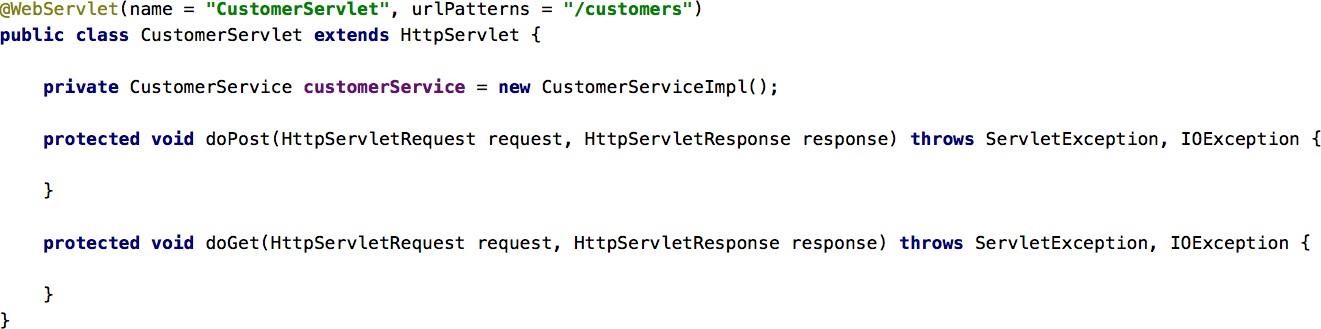
* CustomerServiceImpllàmộtlớp triểnkhaiđầyđủcácphươngthức củainterfaceCustomerService.
* Trong trường hợp này, chúng ta giả lập lớp CustomerServiceImpl làmđốitượngcungcấpdữliệu.Danhsáchkháchhàngsẽđượclưu vàotrong mộtđối tượng Map.Trong một ứng dụng thực tế,danh sáchkhách hàng sẽđược lưu vàocơ sở dữ liệu. Chúng ta sẽphát triểntínhnănglưutrữvàoCSDLởtrongcácphầnsau.

### c 4: Tạo lớpCustomerServiceImpl



**Bước5:TạoCustomerServletđểxửlýcácrequest.**

• Lớp CustomerServlet cómột đối tượng CustomerServiceImpl dùng đểtruyxuấtđếndữliệu.



### Bước6:Điềuhướngviệcxửlýcáctínhnăng

* ĐiềuhướngviệcxửlýcáctínhnăngkhácnhaucủaCustomerServlet thông quatham số action.
* Thamsốactionđượcsửdụngđểquyđịnhhànhđộngmà

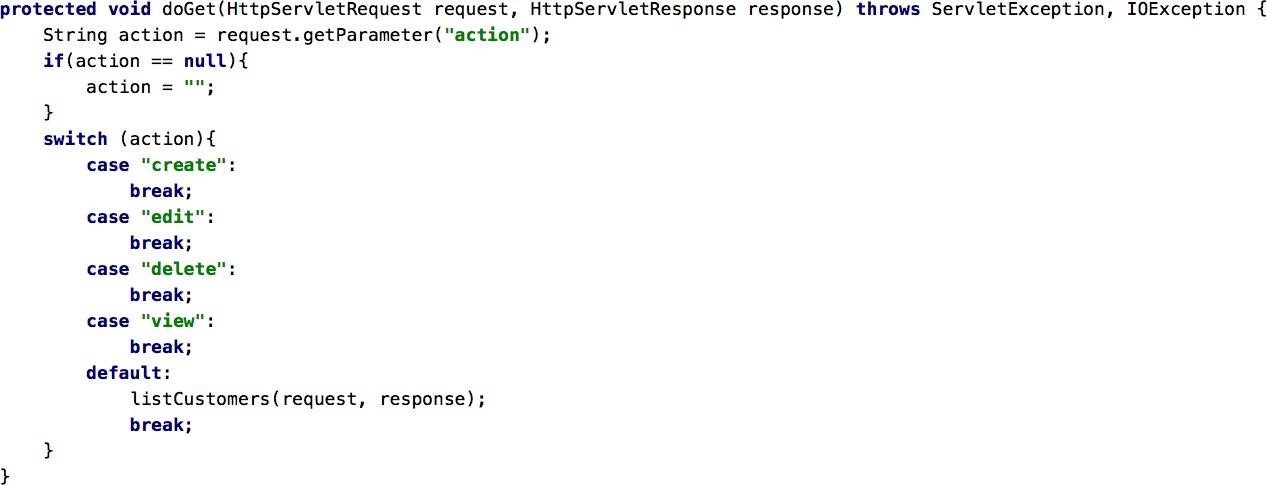
CustomerServlet xử lý trong từngrequest:

* /customers?action=create:Tạomộtkháchhàngmới
* /customers?action=edit:Cậpnhậtthôngtinkháchhàng
* /customers?action=delete:Xoámộtkháchhàng
* Cácgiátrịkhác:Hiểnthịthôngtinkháchhàng

## Bước6:Điềuhướngviệcxửlýcáctínhnăng



* Điềuhướngactionhiểnthịdanhsáchkháchhàngvềphươngthức listCustomers(request,response);



* PhươngthứclistCustomers(request,response)nhậnvềdanhsách kháchhàngvàchuyểnsangviewcustomer/list.jspđểhiểnthị.



* Tạofilecustomer/list.jspđểhiểnthịdanhsáchkháchhàng.



## Bước 8: Khởi chạy ứngdụng

• Khởichạyứngdụngvàđiđếnđường dẫnhttp://localhost:8080/customersđểquansátkếtquả

## Tóm tắt bàihọc

* Kiến trúc phân tầng (multilayered/multitier/n-tier/architecture) là một kiến trúc phần mềmtrong đó các thành phần của hệ thống được tổ chứctheo cáctầngtheo chiều ngang,mỗitầngthựchiện mộtnhóm nhiệmvụcụthể
* KiếntrúcMVCtổchứccácthànhphầncủahệthốngvào3tầng riêng biệt và có kết nối đến nhau: Tầng Model, Tầng View và Tầng

Controller.

* Lợi ích MVC: Dễ tái sử dụng, Dễ mở rộng, Tách phần view và phần nghiệp vụ riêng biệt, Cho phép các Lập trình viên làm việc trên các thànhphầnkhácnhautrongcùngmộtthờiđiểm,Dễbảotrì.

# Hướngdẫn

Hướngdẫnlàmbàithựchànhvàbàitập Chuẩnbịbàitiếptheo:JDBC & CRUD