Java魔法營

高級(集大成者)

羯聲 capricorn.rich.cat@gmail.com

Java Web

應用與實務

- Architecture
  - JakartaEE Pattern
  - 架構實作
  - 開發工具

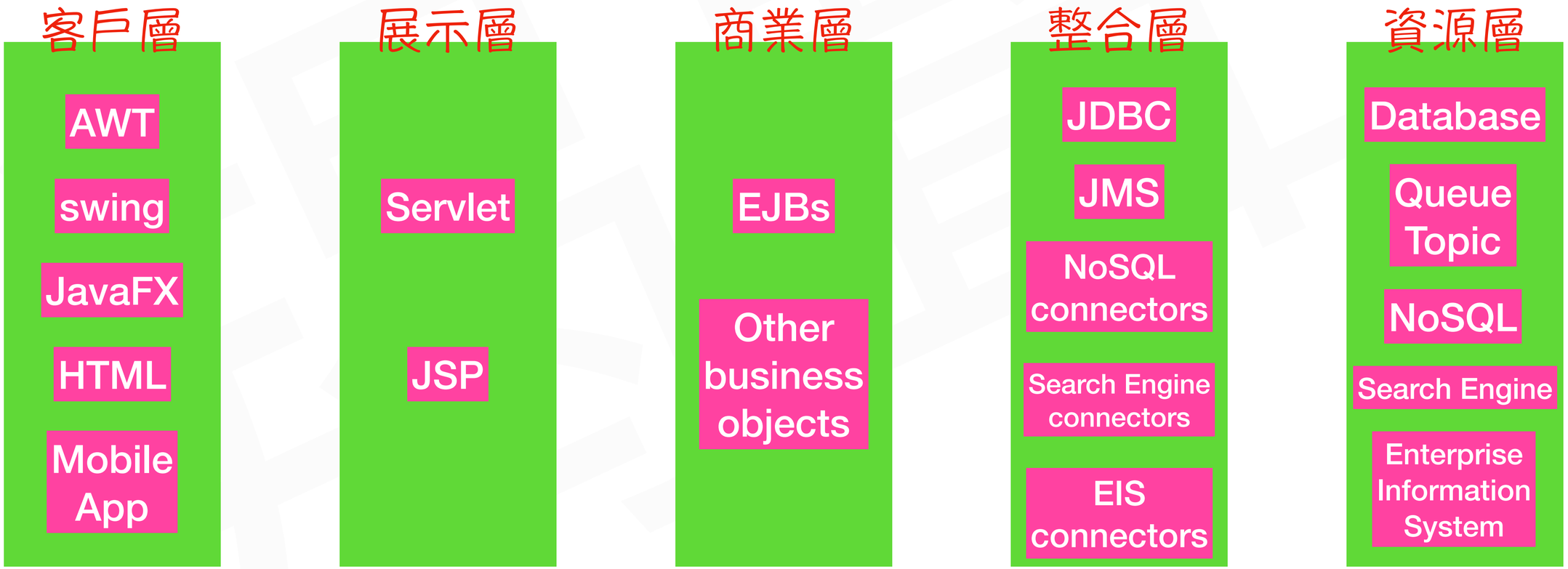
Java Web

應用與實務

- JakartaEE Pattern
  - Intercepting Filter
  - Front Controller
  - MVC
  - Business Delegate
  - Service Locator
  - Data Access Object(DAO)
  - Data Transfer Object(DTO)

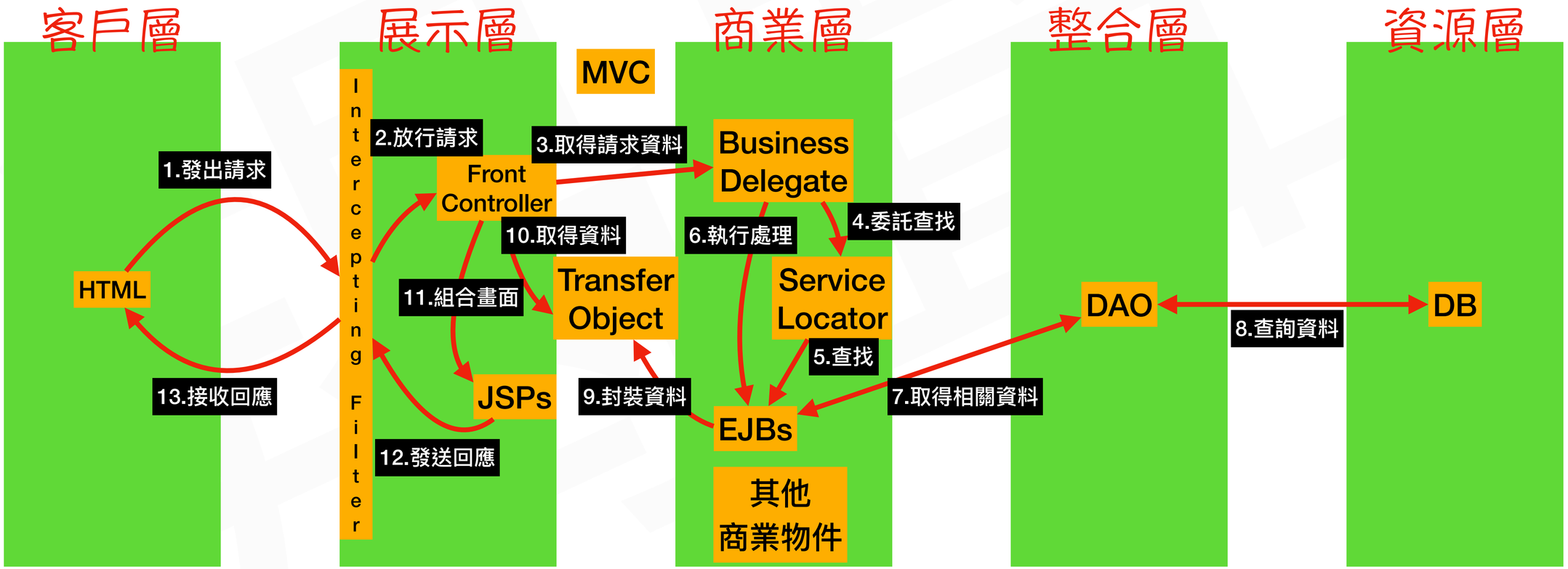
Java Web

JakartaEE Pattern



Java Web

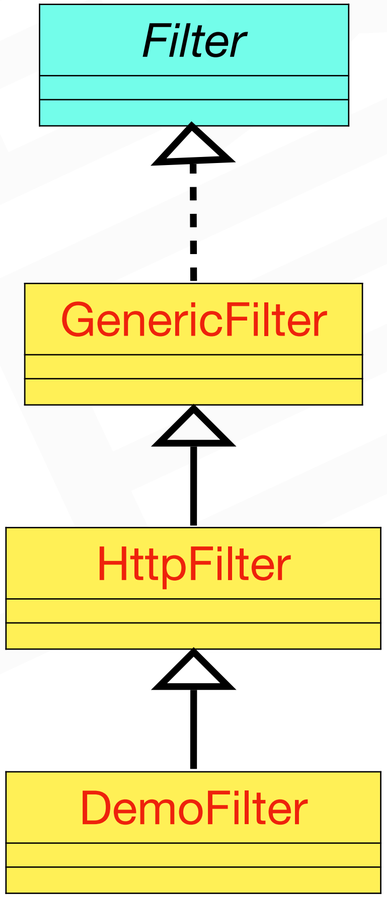
JakartaEE Pattern



Java Web

JakartaEE Pattern

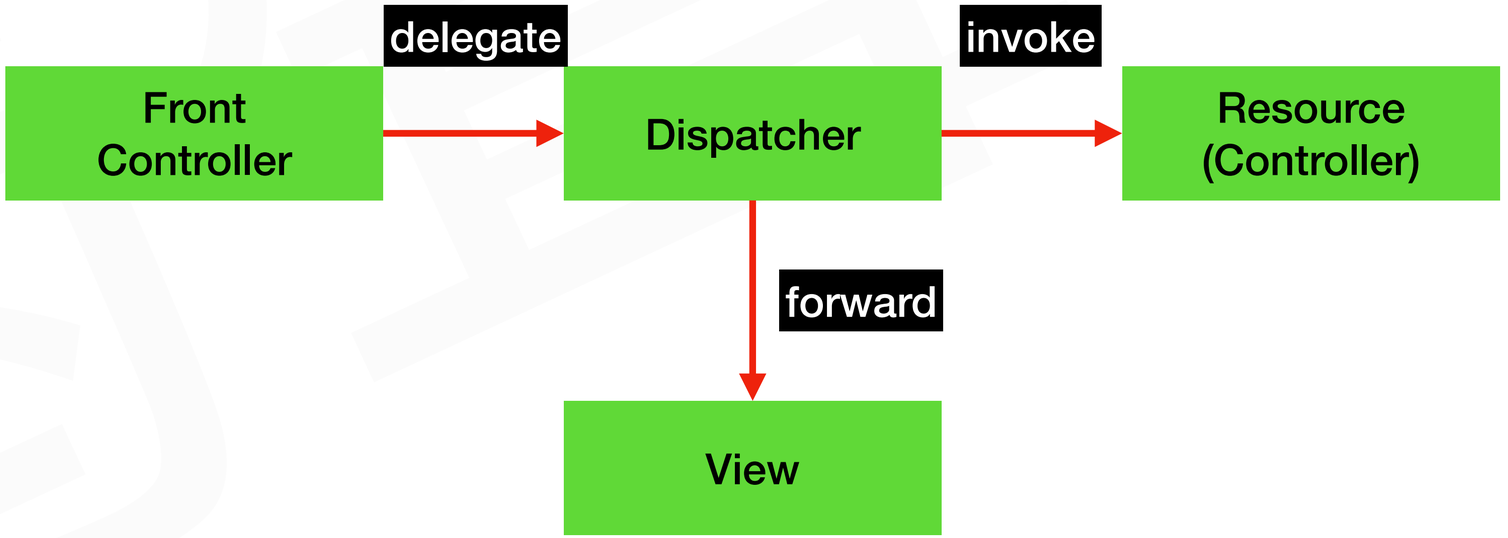
- Intercepting Filter
  - 進出controller或resource前後的過濾



Java Web

JakartaEE Pattern

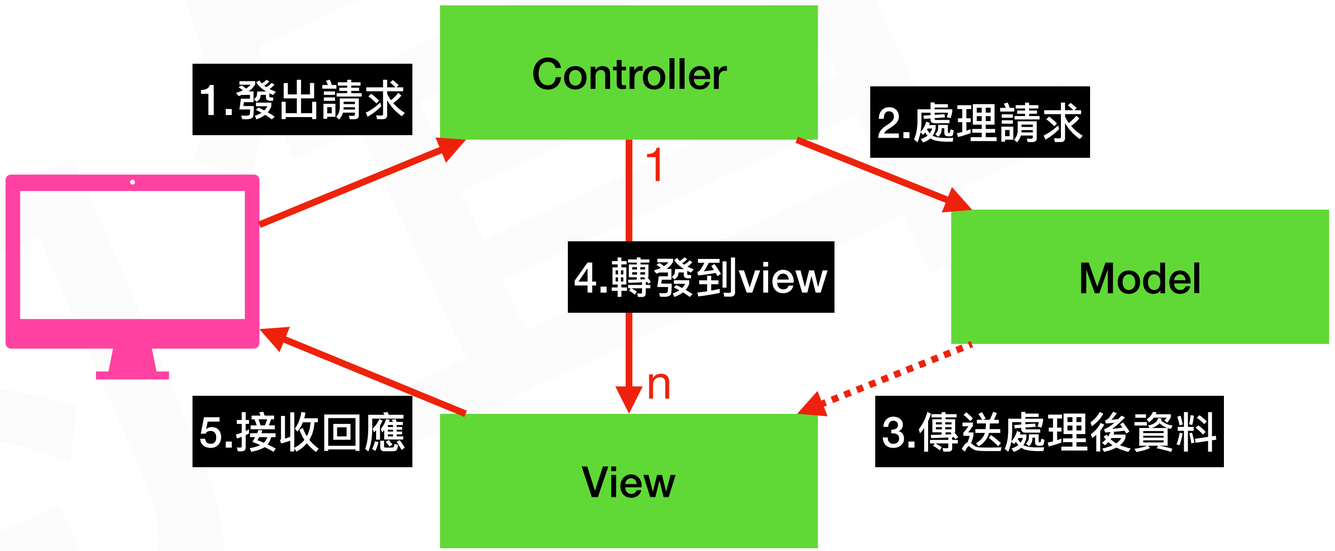
- Front Controller
  - 集中處理request並轉發到相對應處理的resource



Java Web

JakartaEE Pattern

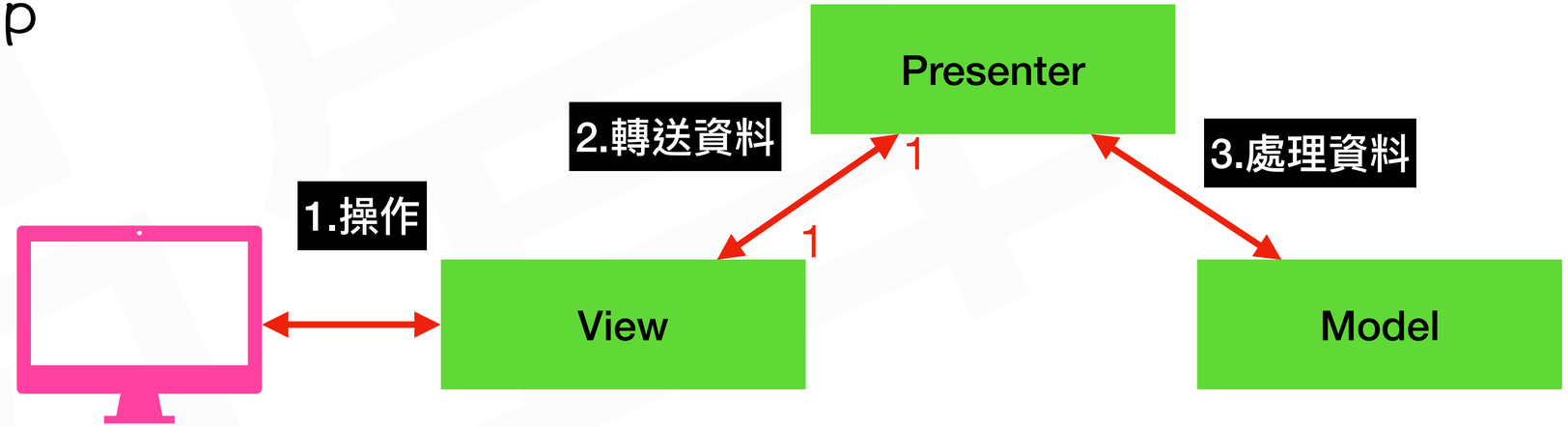
- Model-View-Controller, MVC
  - 將流程控制、呈現畫面及商業邏輯分離
  - Spring MVC, Struts, Struts2
  - 設計模式
    - 觀察者：Model發生變化則通知View進行改變
    - 策略：Controller依行為策略選擇回應的View
    - 組合：View可能具多個子畫面組合而成



Java Web

JakartaEE Pattern

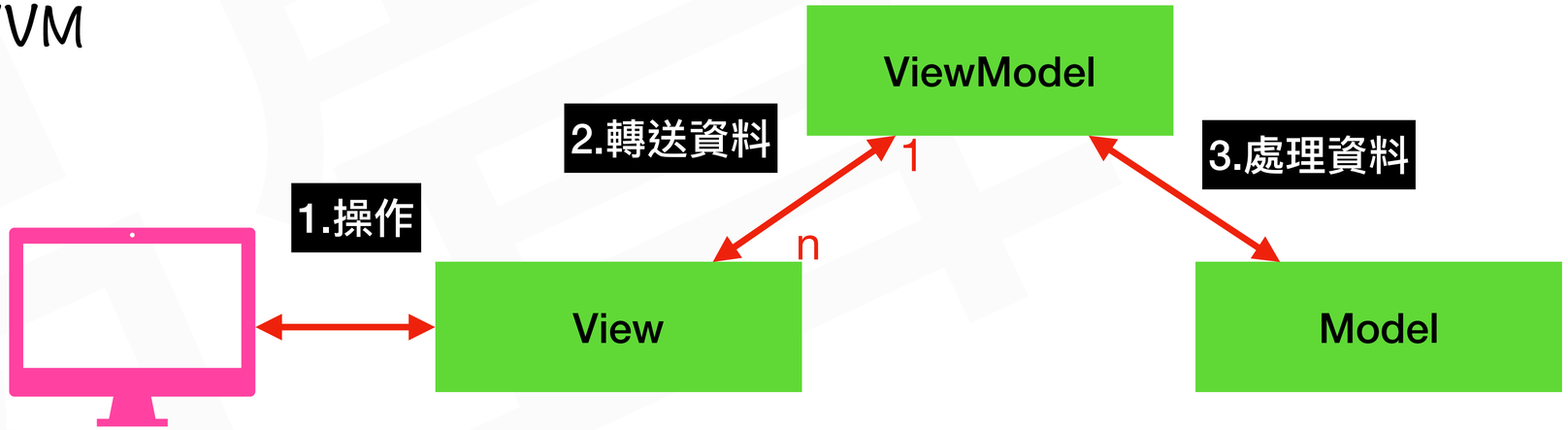
- Model-View-Presenter, MVP
  - MVC的變體
  - View與Model解耦合



Java Web

JakartaEE Pattern

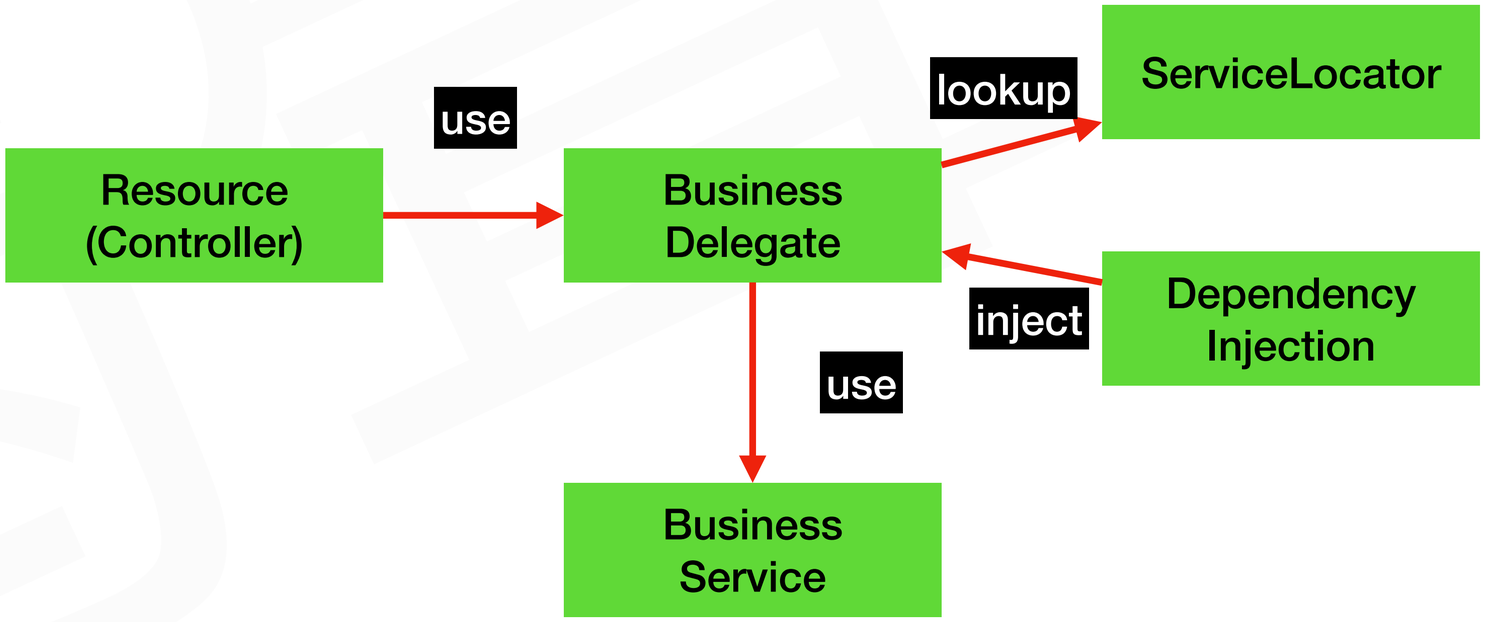
- Model-View-ViewModel, MVVM
  - MVC的變體
  - 透過ViewModel使View與Model之間雙向綁定



Java Web

JakartaEE Pattern

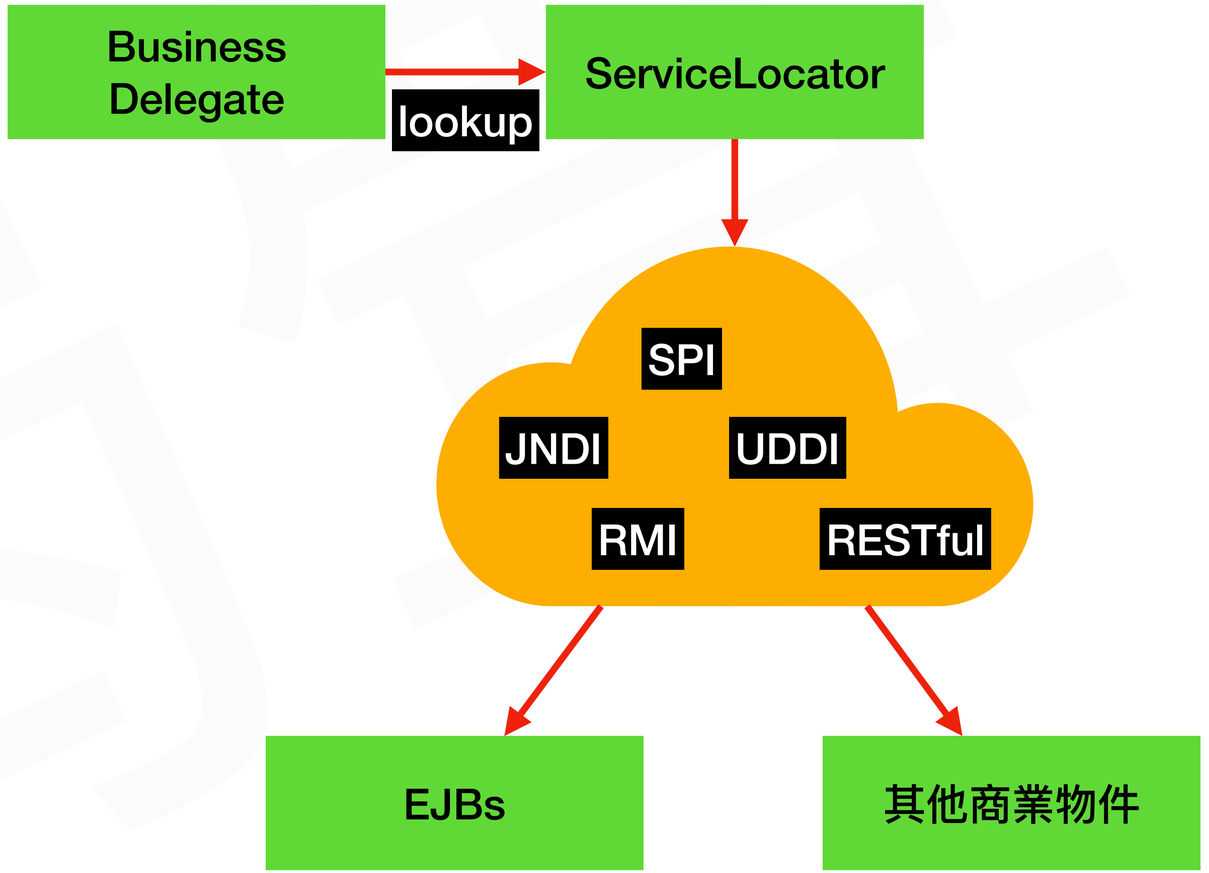
- Business Delegate
  - 封裝查找及商業邏輯
  - 將資料與商業邏輯的程式碼分離



Java Web

JakartaEE Pattern

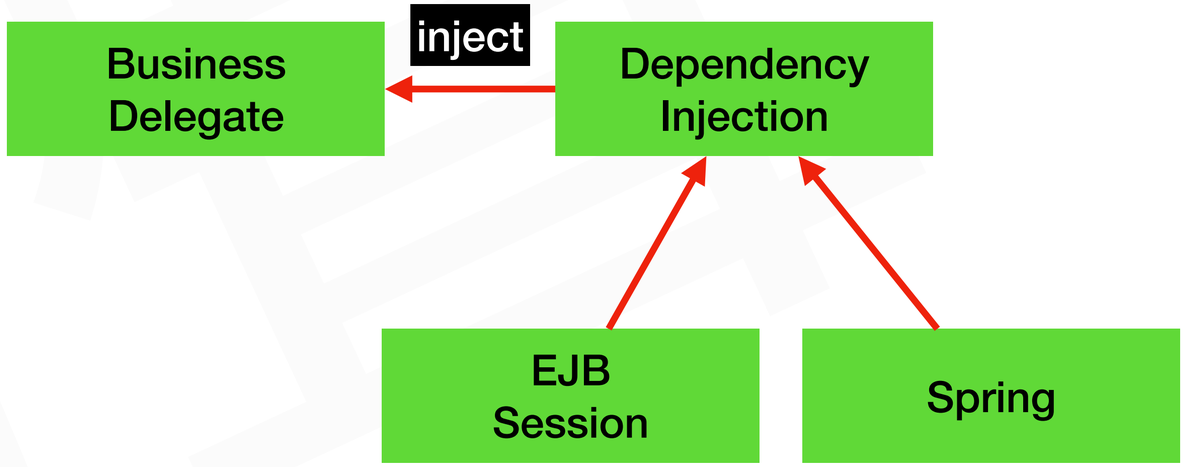
- Service Locator
  - 主動查找服務物件



Java Web

JakartaEE Pattern

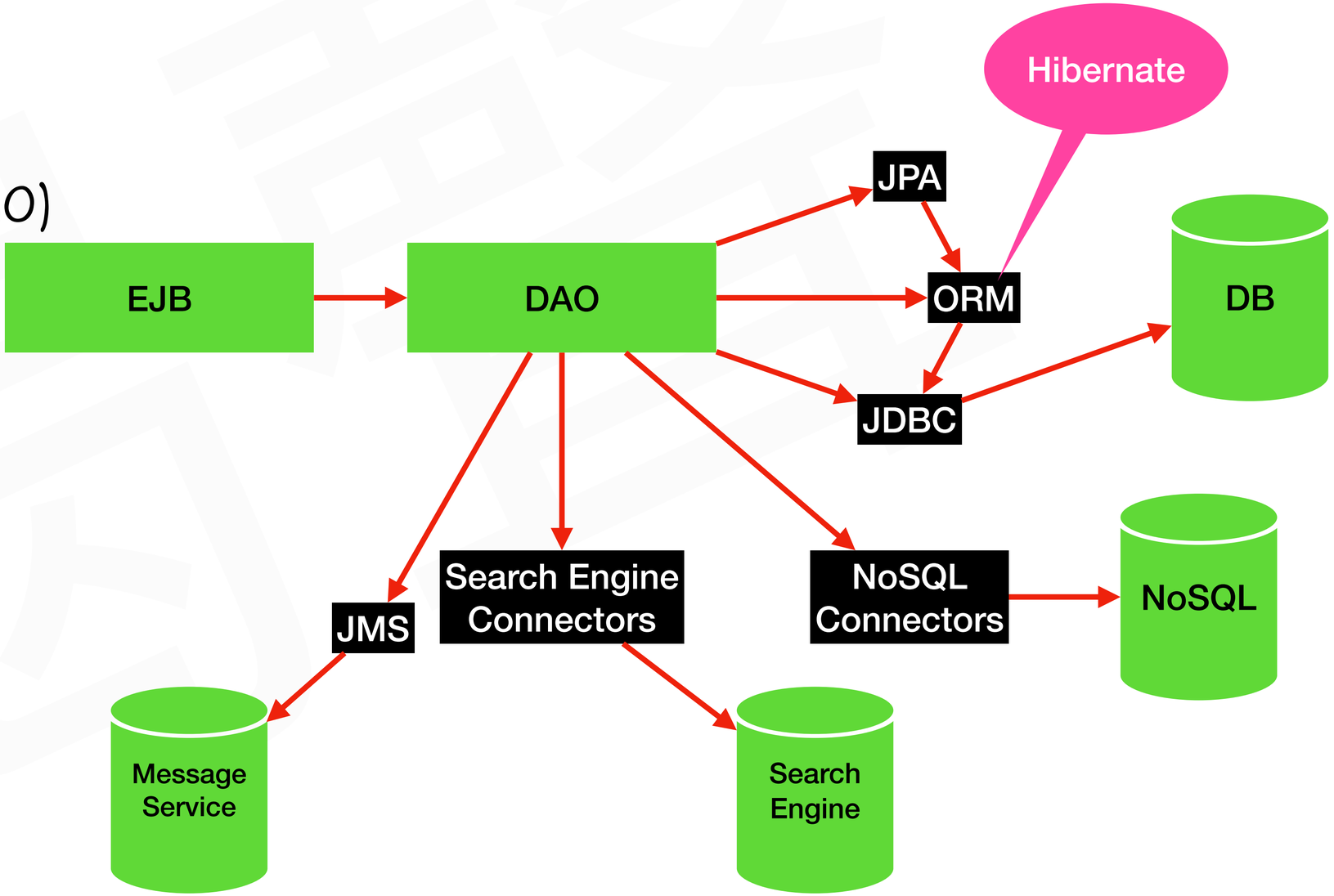
- Dependency Injection
  - 容器或框架主動提供依賴物件
  - Spring, EJB3



Java Web

JakartaEE Pattern

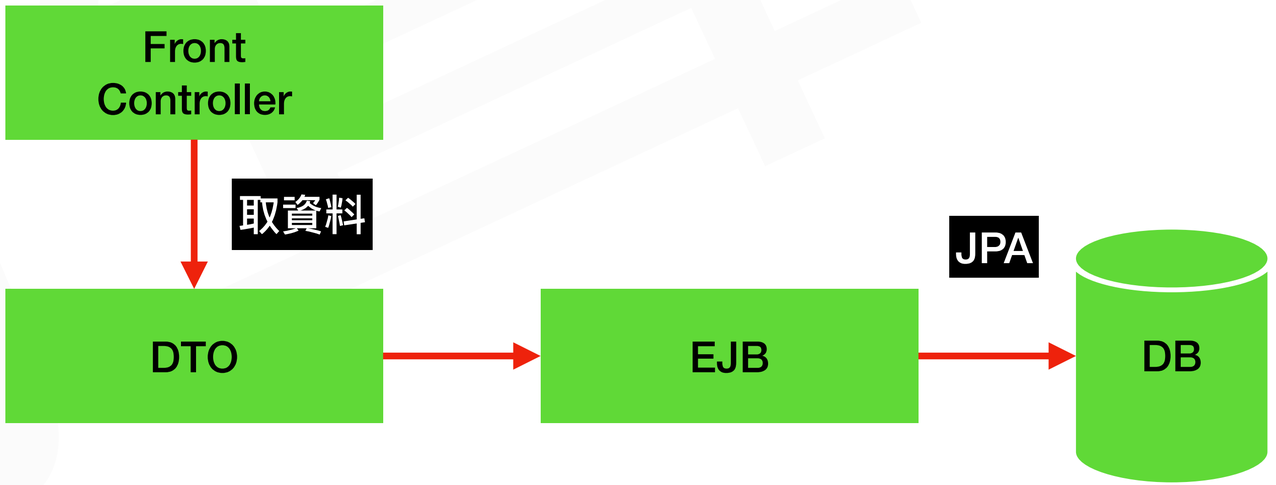
- Data Access Object(DAO)
  - 存取資料的操作



Java Web

JakartaEE Pattern

- Data Transfer Object(DTO)
  - 封裝商業服務資料
  - 減少網路traffic



Java Web

應用與實務

- 架構實作
  - Dependency Injection
    - Aspect Oriented Programming
  - Model View Controller
    - RESTful
  - Object Relational Mapping

Java Web

架構實作

- 準備資料庫(使用postgresql)

```
create database web;
\c web
create table users(
    id bigint not null,
    name varchar(255) not null,
    age int,
    primary key (id)
);
create table role(
    id bigint not null,
    name varchar(255) not null,
    primary key (id)
);
```

Java Web

架構實作

- 第三方套件
  - JSTL
    - jakarta.servlet.jsp.jstl-api-3.0.0.jar
    - jakarta.servlet.jsp.jstl-3.0.1.jar
  - JDBC
    - postgresql-42.6.0.jar
- JSON
  - jackson-core-2.15.2.jar
  - jackson-databind-2.15.2.jar
  - jackson-annotations-2.15.2.jar

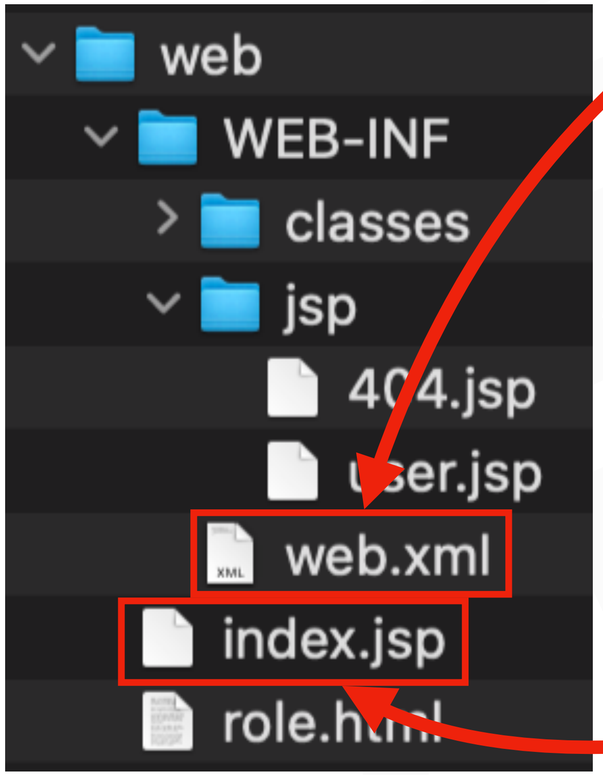
Java Web

架構實作

- 編譯
  - mkdir workspace && cd workspace
  - javac -d ../apache-tomcat-10.1.11/webapps/web/WEB-INF/classes -cp ../apache-tomcat-10.1.11/lib/servlet-api.jar:../apache-tomcat-10.1.11/lib/jackson-databind-2.15.2.jar:../apache-tomcat-10.1.11/lib/jackson-core-2.15.2.jar *.java

Java Web

架構實作

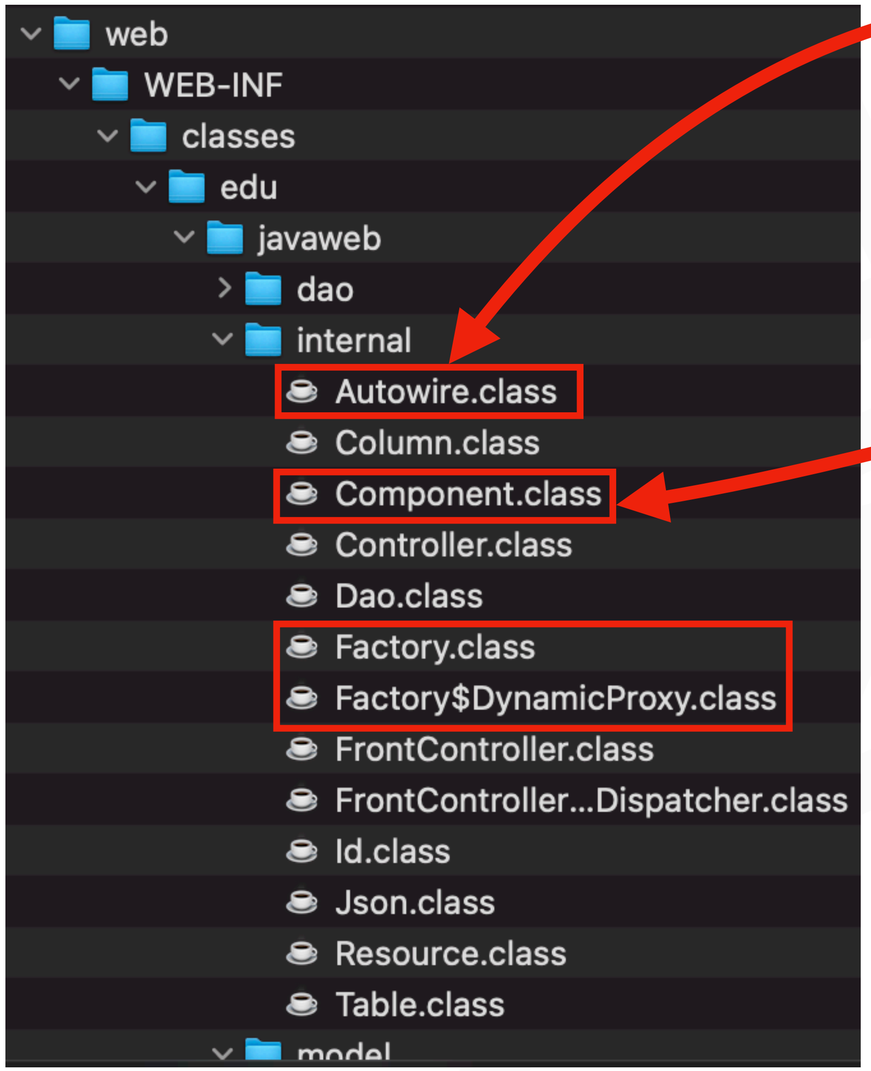



```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
                      https://jakarta.ee/xml/ns/jakartaee/web-app_6_0.xsd"
  version="6.0"
  metadata-complete="false">

    <request-character-encoding>UTF-8</request-character-encoding>
    <response-character-encoding>UTF-8</response-character-encoding>
</web-app>
```

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"
         trimDirectiveWhitespaces="true" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Index Page</title>
</head>
<body>
    <h1>Welcome to the Index Page</h1>
    <p>
        <a href="<c:url value='/mvc/user'/>">Go to User Page</a>
    </p>
    <p>
        <a href="<c:url value='role.html'/>">Go to Role Page</a>
    </p>
</body>
</html>
```

架構實作

Dependency Injection

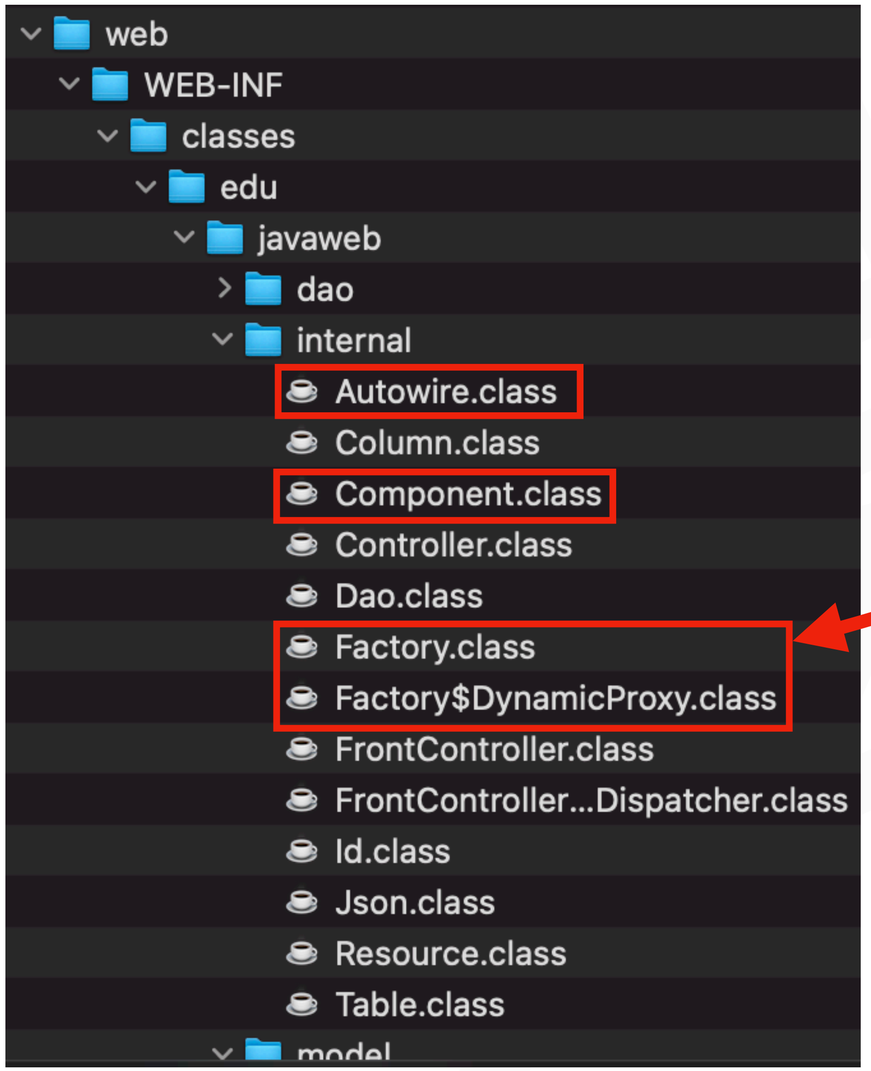



```java
package edu.javaweb.internal;
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
public @interface Autowire {
}
```

```java
package edu.javaweb.internal;
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface Component {
}
```

架構實作

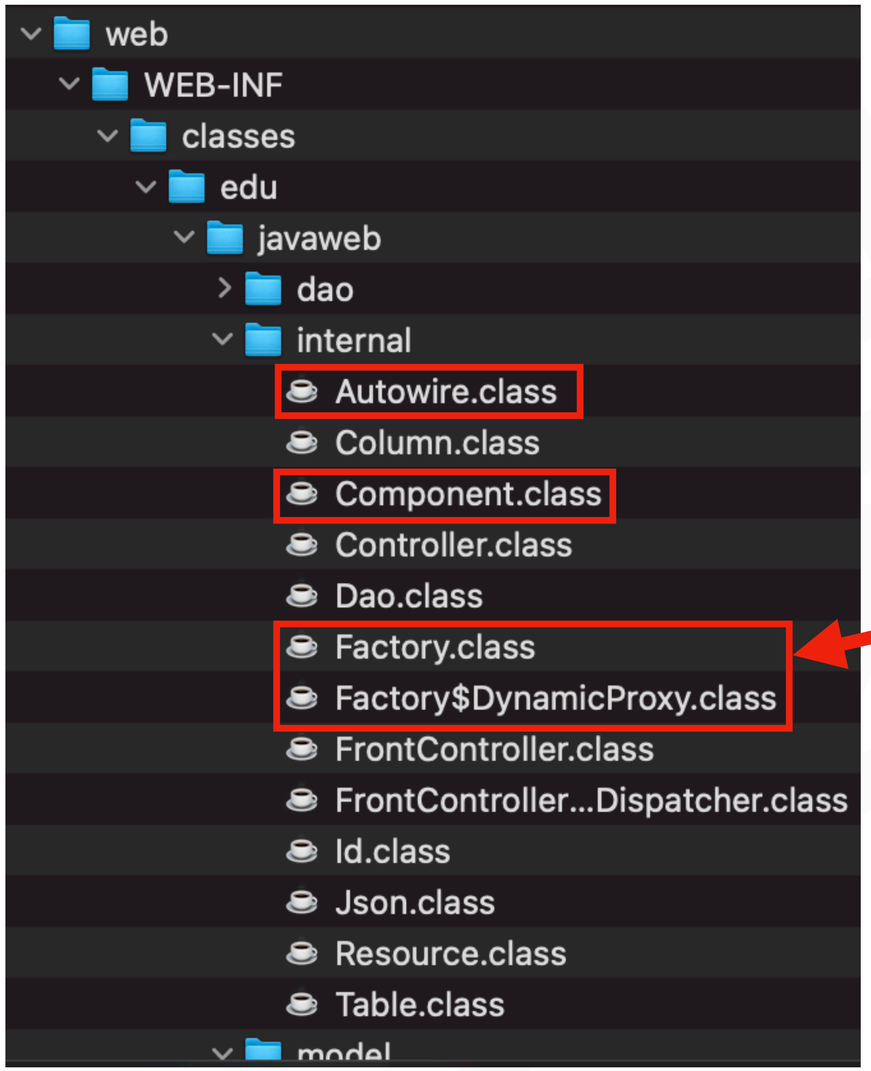
Dependency Injection




```java
package edu.javaweb.internal;
import java.io.*;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.util.*;
import java.util.function.*;
import java.util.concurrent.*;
import edu.javaweb.service.*;
import edu.javaweb.dao.*;
import edu.javaweb.resource.*;
import java.lang.reflect.*;
@WebListener
public class Factory implements ServletContextListener {
    private static Map<Class<?>, List<DynamicProxy>> pool
        = new ConcurrentHashMap<>();
    private static List<Class<?>> classes = List.of(
        UserServiceImpl.class,
        UserDaoImpl.class,
        UserResource.class,
        RoleServiceImpl.class,
        RoleDaoImpl.class,
        RoleResource.class
    );
```

架構實作

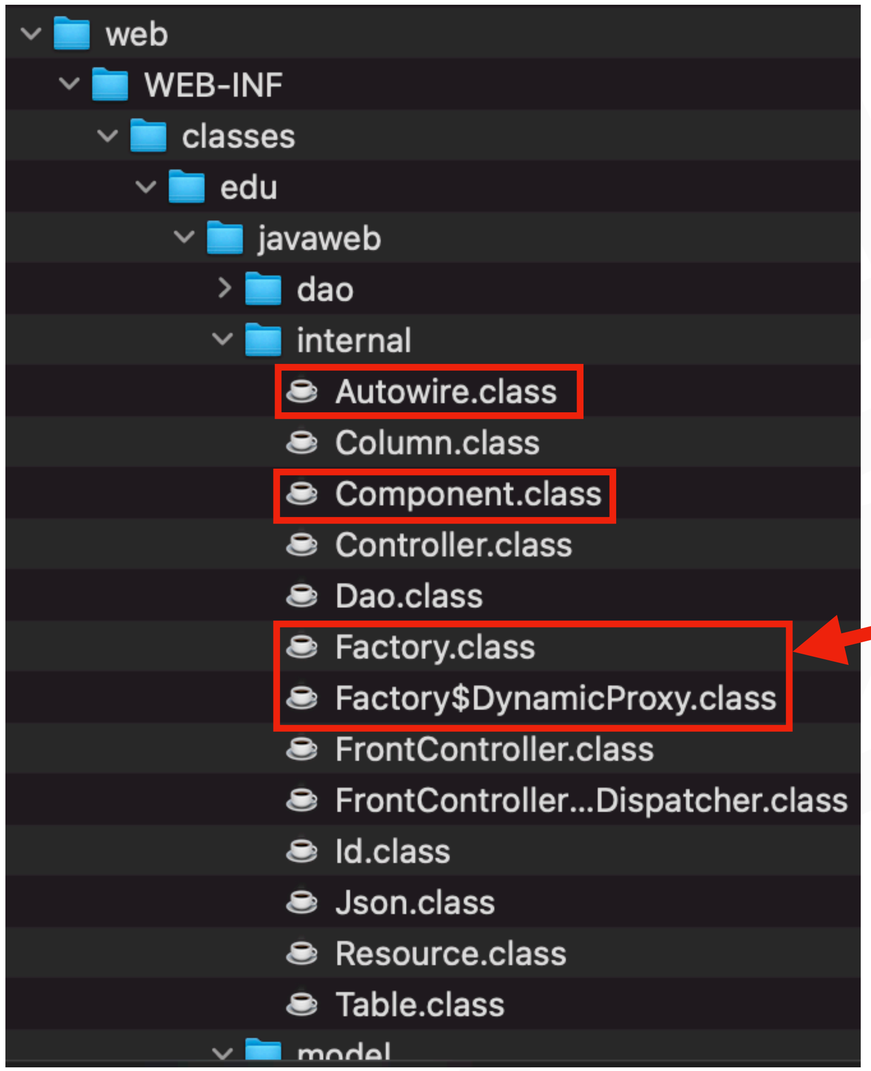
Dependency Injection




```java
@Override
public void contextInitialized(ServletContextEvent sce) {
    for(var c: classes){
        var component = c.getDeclaredAnnotation(Component.class);
        if(Objects.nonNull(component)){
            try{
                var inf = c.getInterfaces()[0];
                var o = c.getDeclaredConstructor().newInstance();
                var list = pool.get(inf);
                if(Objects.isNull(list)){
                    list = new CopyOnWriteArrayList<>();
                }
                list.add(new DynamicProxy(o));
                pool.put(inf, list);
            } catch(Exception e){
                System.out.println(c.getName()+" newInstance fail.");
                e.printStackTrace();
            }
        }
    }
    for(var c: classes){
        Arrays.asList(c.getDeclaredFields())
            .stream()
            .peek(f->f.setAccessible(true))
            .filter(f->Objects.nonNull(f.getAnnotation(Autowire.class)))
            .forEach(f->{
                var inf = c.getInterfaces()[0];
                var obj = getProxy(inf, p->p.getTarget().getClass().equals(c)).getTarget();
                var type = f.getType();
                var tObj = getProxy(type, p->true).getProxy();
                try{
                    f.set(obj, tObj);
                } catch(Exception e){
                    System.out.println(c.getName()+"."+f.getName()+" set fail.");
                    e.printStackTrace();
                }
            });
    }
}
```

架構實作

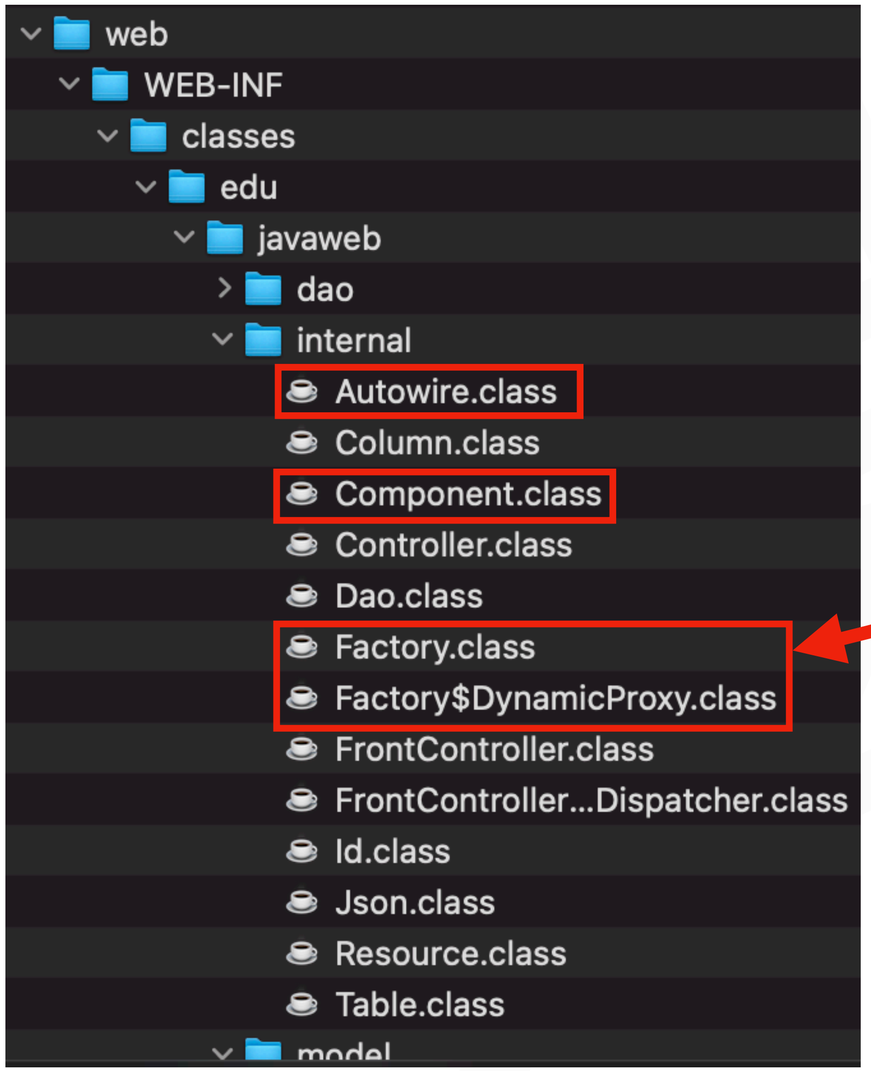
Dependency Injection




```
static DynamicProxy getProxy(Class<?> t, Predicate<DynamicProxy> p){
    var list = pool.get(t);
    if(list.size()==1){
        return list.get(0);
    } else {
        return list.stream()
            .filter(p)
            .findFirst()
            .orElseGet(()->null);
    }
}
static List<DynamicProxy> getProxies(Class<?> t){
    return pool.get(t);
}
public static <T> T getObject(Class<T> t){
    return pool.get(t)
        .stream()
        .map(p->p.getProxy())
        .findFirst()
        .map(o->t.cast(o))
        .orElseGet(()->null);
}
public static <T> List<T> getObjects(Class<T> t){
    return pool.get(t)
        .stream()
        .map(p->p.getProxy())
        .map(o->t.cast(o))
        .toList();
}
```

架構實作

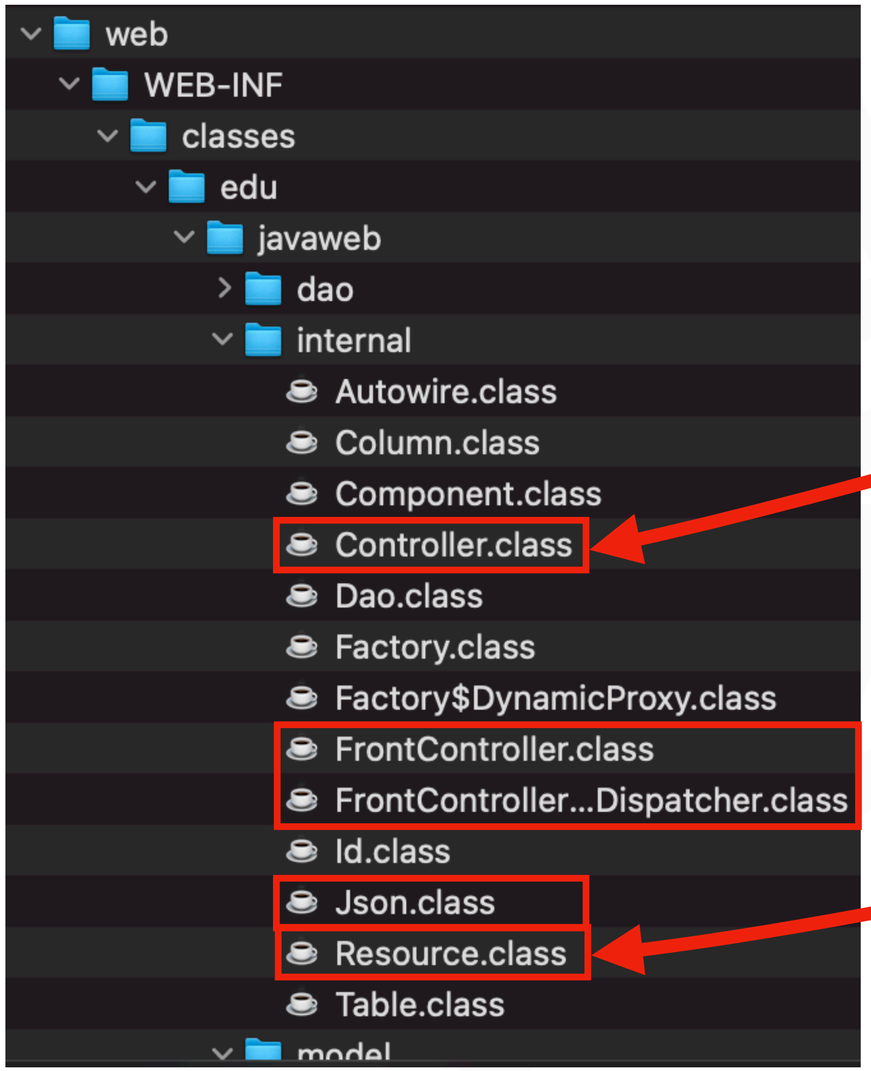
Aspect Oriented Programming




```
    public static class DynamicProxy implements InvocationHandler {
        private Object target;
        public DynamicProxy(Object t){
            this.target = t;
        }
        public Object getTarget(){
            return target;
        }
        public Object getProxy(){
            return Proxy.newProxyInstance(
                    target.getClass().getClassLoader(),
                    target.getClass().getInterfaces(),
                    this
                );
        }
        public Object invoke(Object proxy, Method m, Object[] args)
                throws Exception {
            System.out.println(target.getClass()+"."+m.getName()+"("+Arrays.toString(args)+")");
            var result = m.invoke(target, args);
            System.out.println(target.getClass()+"."+m.getName()+" return "+result);
            return result;
        }
    }
}
```

架構實作

Model View Controller

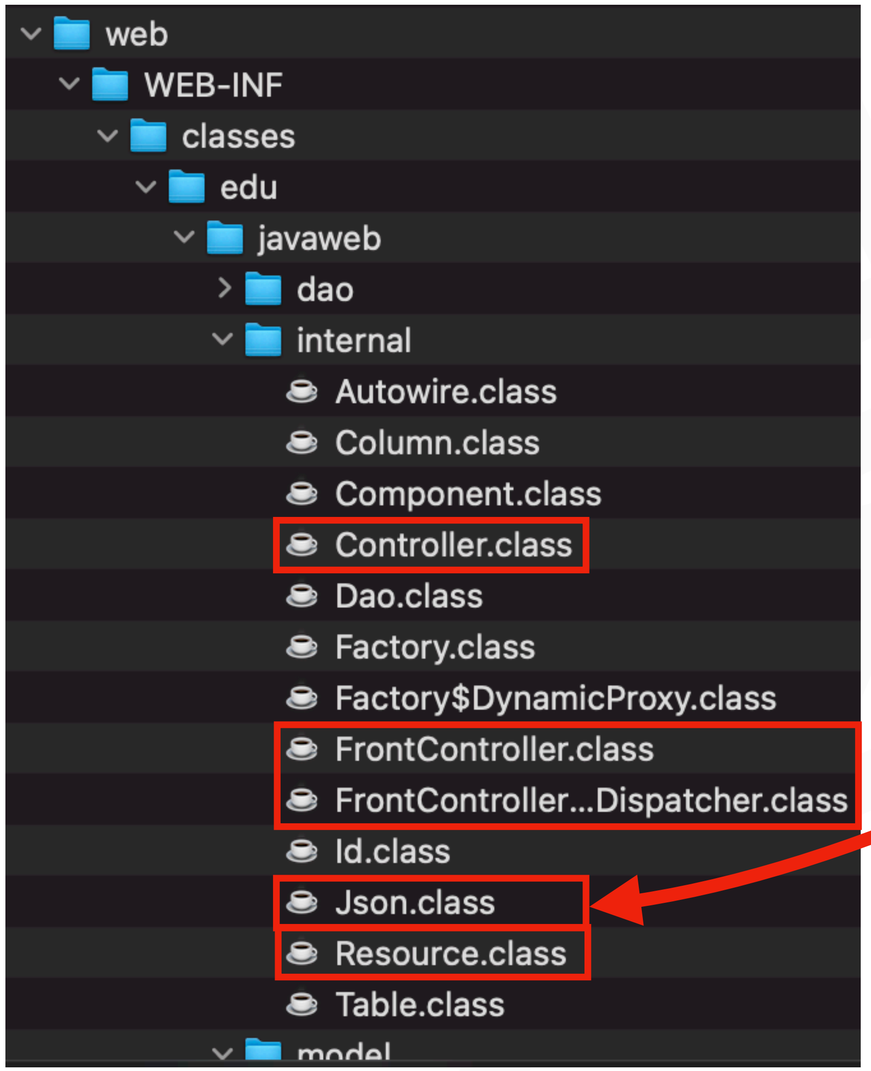



```java
package edu.javaweb.internal;
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface Controller {
    String method();
    String path();
    String forward();
}
```

```java
package edu.javaweb.internal;
public interface Resource {
}
```

架構實作

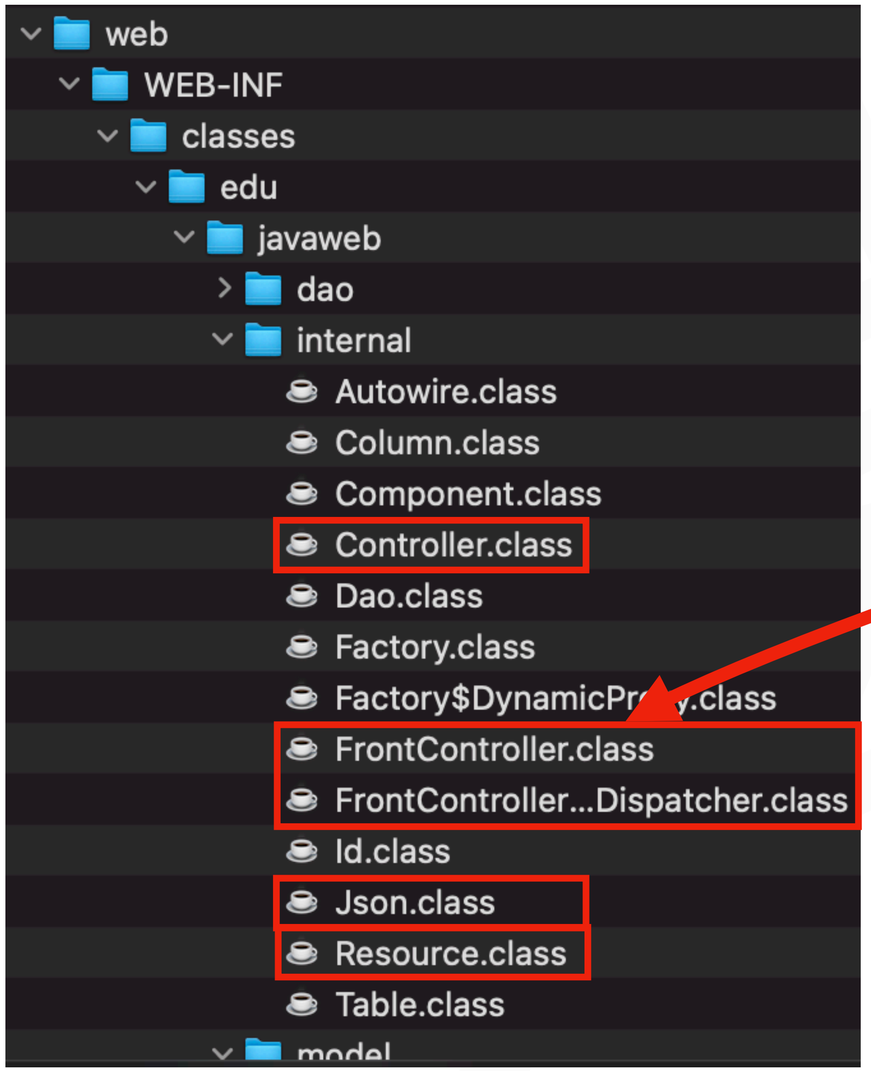
Model View Controller




```java
package edu.javaweb.internal;
import com.fasterxml.jackson.databind.ObjectMapper;
import jakarta.servlet.http.*;
import java.io.*;
import java.nio.charset.*;

public class Json {
    private static ObjectMapper mapper = new ObjectMapper();

    public static String to(Object o){
        try{
            return mapper.writeValueAsString(o);
        } catch(Exception e){
            e.printStackTrace();
            return null;
        }
    }

    public static <T> T as(String json, Class<T> t){
        try{
            return mapper.readValue(json, t);
        } catch(Exception e){
            e.printStackTrace();
            return null;
        }
    }

    public static <T> T as(HttpServletRequest request, Class<T> t){
        try{
            return as(getRequestBody(request), t);
        } catch(Exception e){
            e.printStackTrace();
            return null;
        }
    }

    private static String getRequestBody(HttpServletRequest request) throws IOException {
        StringBuilder stringBuilder = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(
                new InputStreamReader(request.getInputStream(), StandardCharsets.UTF_8))) {
            String line;
            while ((line = reader.readLine()) != null) {
                stringBuilder.append(line);
            }
        }
        return stringBuilder.toString();
    }
}
```

架構實作

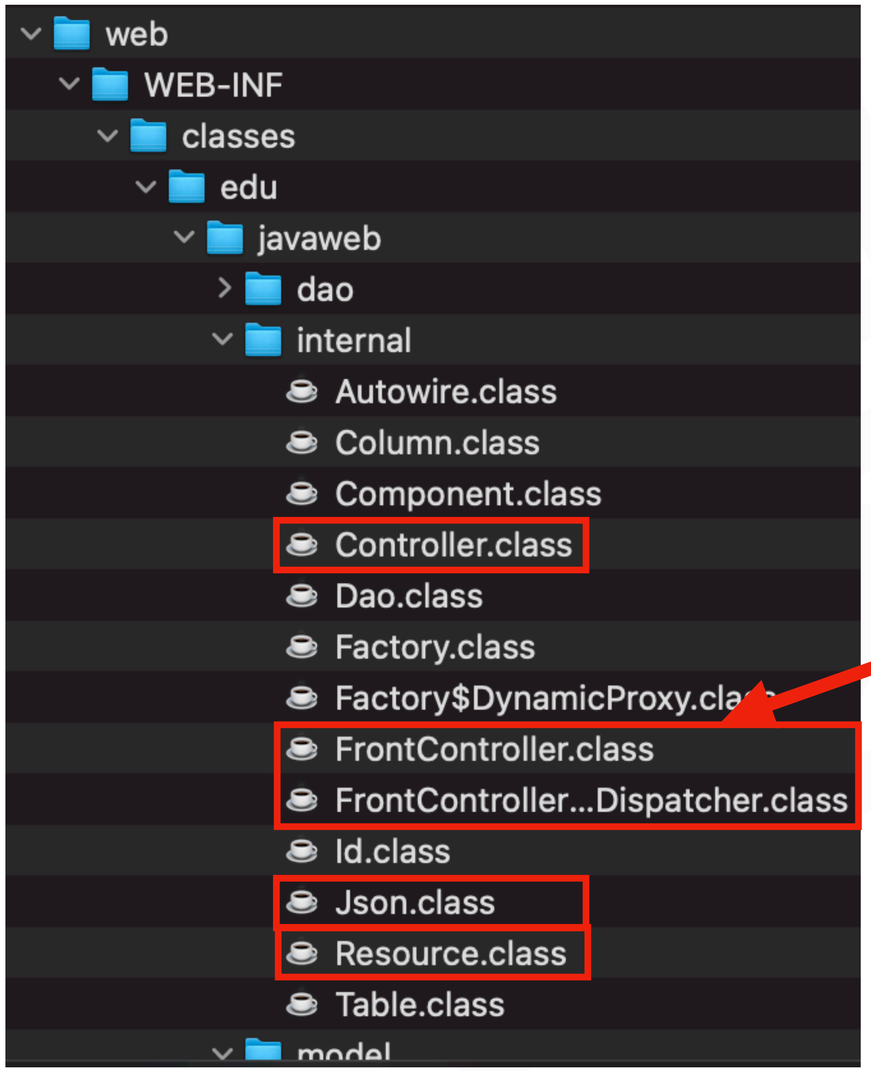
Model View Controller




```java
package edu.javaweb.internal;
import jakarta.servlet.*;
import jakarta.servlet.http.*;
import jakarta.servlet.annotation.*;
import java.io.*;
import java.util.*;
import java.util.concurrent.*;
import edu.javaweb.resource.*;
import java.lang.reflect.*;

@WebServlet(urlPatterns="/mvc/*",
    loadOnStartup=1
)
public class FrontController extends HttpServlet {
    private static final Map<String, Map<String, ControllerDispatcher>> mvc
        = Map.ofEntries(
              Map.entry("GET", new ConcurrentHashMap<String, ControllerDispatcher>()),
              Map.entry("POST", new ConcurrentHashMap<String, ControllerDispatcher>()),
              Map.entry("PUT", new ConcurrentHashMap<String, ControllerDispatcher>()),
              Map.entry("DELETE", new ConcurrentHashMap<String, ControllerDispatcher>())
          );

    private static List<Class<? extends Resource>> classes
        = List.of(
            UserResource.class,
            RoleResource.class
        );
    public void init(){
        classes.stream()
            .forEach(c->Arrays.asList(c.getDeclaredMethods())
                .stream()
                .filter(m->Objects.nonNull(m.getAnnotation(Controller.class)))
                .forEach(m->{
                    var controller = m.getAnnotation(Controller.class);
                    var httpMethod = mvc.get(controller.method());
                    var obj = getControllerObject(c);
                    httpMethod.put(controller.path(), new ControllerDispatcher(
                        obj, m, controller.forward()));
                }));
    }
```

架構實作

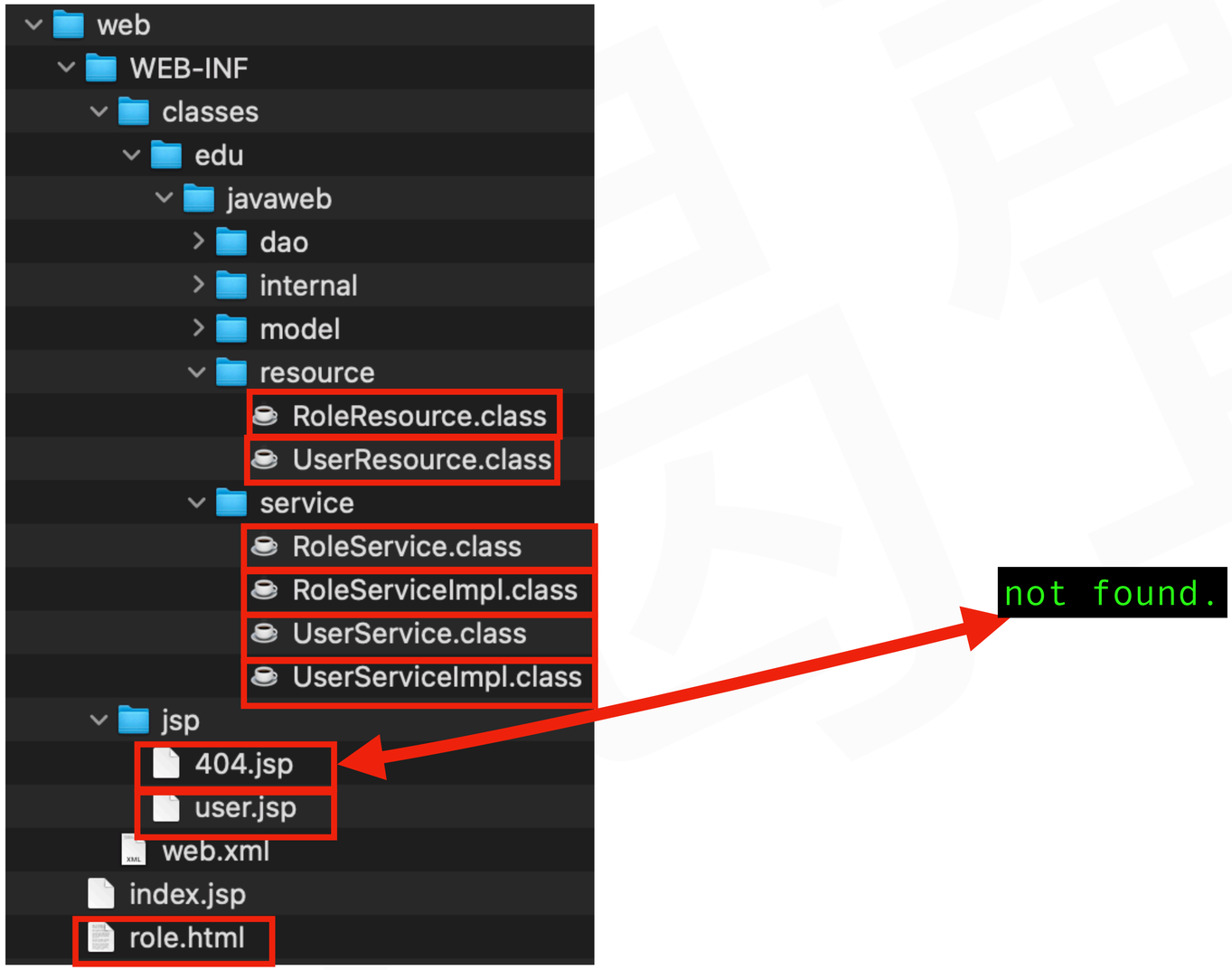
RESTful




```
    private Object getControllerObject(Class<?> c){
        return Factory.getProxies(Resource.class)
            .stream()
            .map(p->p.getTarget())
            .filter(r->c.equals(r.getClass()))
            .findFirst()
            .orElseThrow();
    }

    public void service(HttpServletRequest request,
            HttpServletResponse response)
            throws IOException, ServletException {
        var method = request.getMethod();
        var pathInfo = request.getPathInfo();
        var c = mvc.get(method).get(pathInfo);
        var forward = "/404.jsp";
        if(Objects.nonNull(c)){
            forward = c.forward();
            try{
                c.method().invoke(c.target(), request,response);
            } catch(Exception e){
                e.printStackTrace();
            }
        }
        if(forward.contains("json")){
            response.setContentType("application/json");
            var o = request.getAttribute("json");
            var r = Json.to(o);
            var out = response.getWriter();
            out.write(r);
            out.close();
        } else {
            request.getRequestDispatcher("/WEB-INF/jsp"+forward)
                .forward(request, response);
        }
    }

    private static record ControllerDispatcher(
        Object target, Method method, String forward) {
    }
}
```

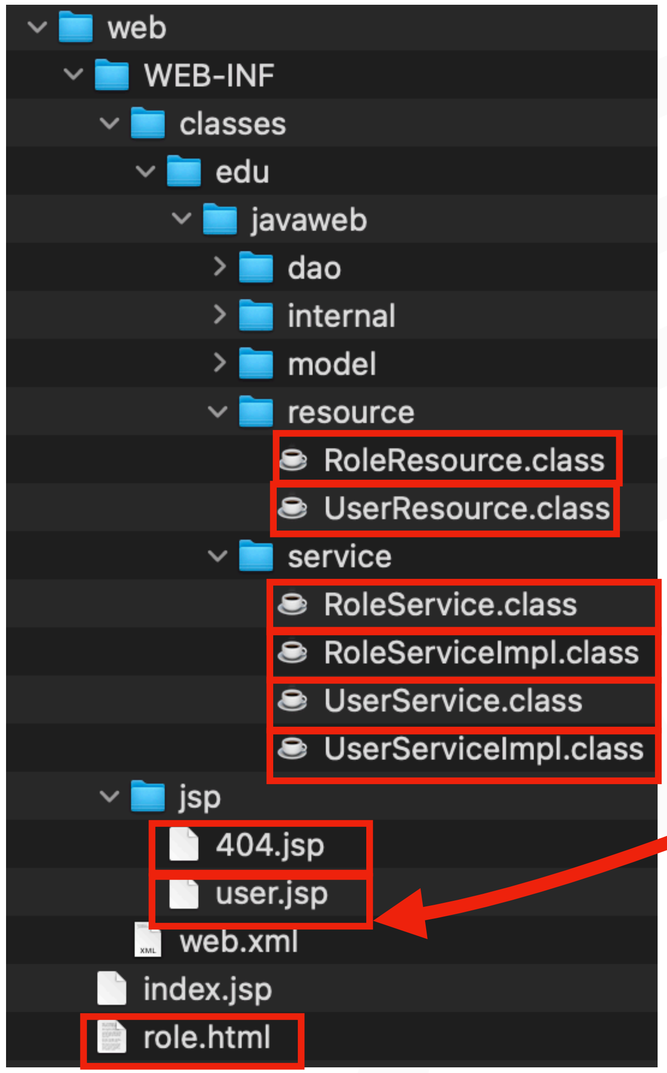
架構實作

RESTful



架構實作

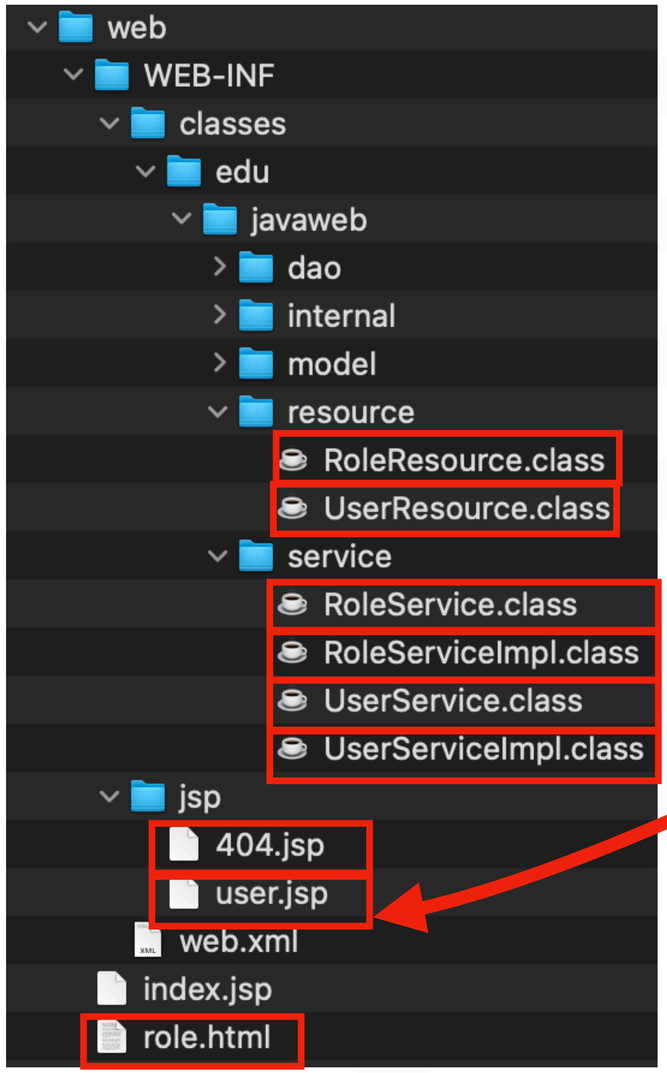
RESTful




```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"
         trimDirectiveWhitespaces="true" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<html>
<head>
    <title>User Information</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        table {
            width: 100%;
            border-collapse: collapse;
        }
        th, td {
            border: 1px solid #ccc;
            padding: 10px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
        a {
            text-decoration: none;
            color: blue;
        }
        form {
            margin-bottom: 20px;
        }
        label {
            margin-right: 10px;
        }
    </style>
</head>
```

架構實作

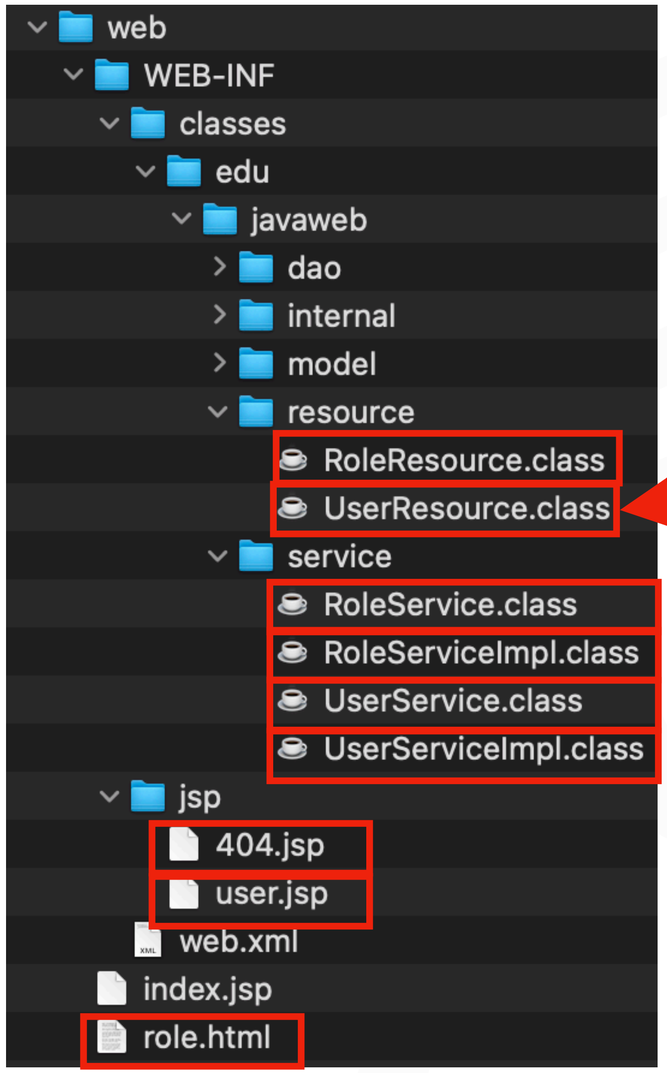
RESTful




```jsp
<body>
    <h1>User Information</h1>

    <!-- Link to Home Page -->
    <a href="<c:url value='/' />">回首頁</a>

    <!-- User Form -->
    <form action="<c:url value='/mvc/user' />" method="post">
        <label for="id">ID:</label>
        <input type="text" name="id" id="id" value="${user.id}"><br>

        <label for="name">Name:</label>
        <input type="text" name="name" id="name" value="${user.name}"><br>

        <label for="age">Age:</label>
        <input type="text" name="age" id="age" value="${user.age}"><br>

        <input type="submit" value="Submit">
    </form>

    <!-- User Table -->
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Age</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            <c:forEach var="user" items="${users}">
                <tr>
                    <td>${user.id}</td>
                    <td>${user.name}</td>
                    <td>${user.age}</td>
                    <td>
                        <a href="<c:url value='/mvc/user/modify' var='modifyUrl'>
                                    <c:param name='id' value='${user.id}' />
                                 </c:url>${modifyUrl}">修改</a> |
                        <a href="<c:url value='/mvc/user/delete' var='deleteUrl'>
                                    <c:param name='id' value='${user.id}' />
                                 </c:url>${deleteUrl}">刪除</a>
                    </td>
                </tr>
            </c:forEach>
        </tbody>
    </table>
</body>
</html>
```

架構實作

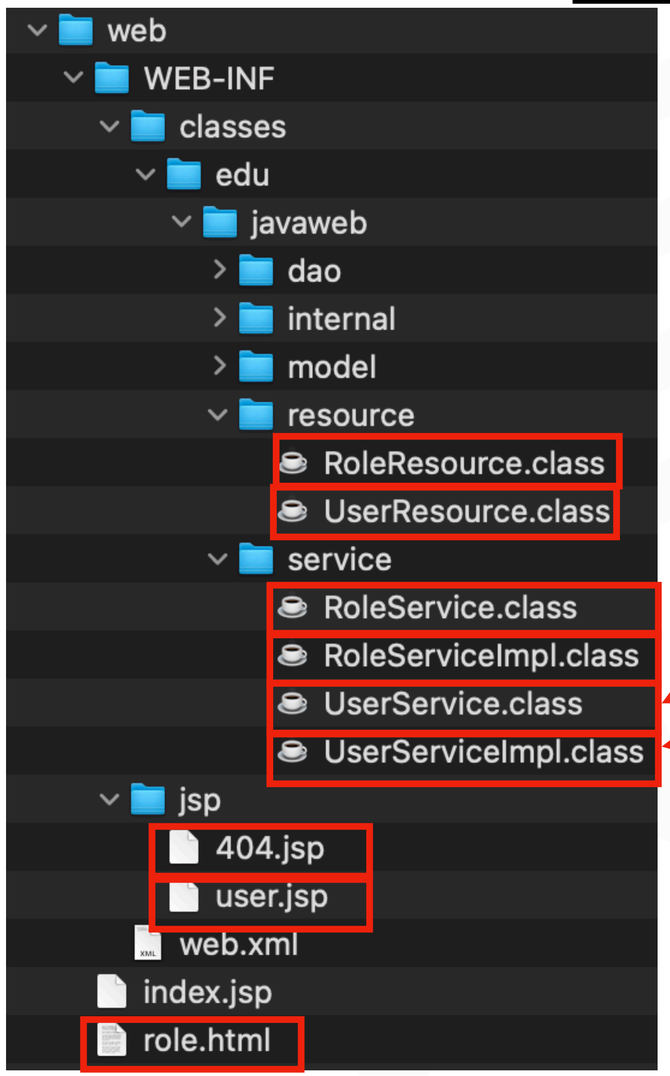
RESTful




```java
package edu.javaweb.resource;
import edu.javaweb.internal.*;
import edu.javaweb.service.*;
import jakarta.servlet.http.*;
import java.util.*;

@Component
public class UserResource implements Resource {
    @Autowire
    private UserService service;

    @Controller(method="GET", path="/user", forward="/user.jsp")
    public void getUser(HttpServletRequest request,
            HttpServletResponse response) {
        var users = service.find();
        request.setAttribute("users", users);
    }
    @Controller(method="GET", path="/user/modify", forward="/user.jsp")
    public void getModifyUser(HttpServletRequest request,
            HttpServletResponse response) {
        var users = service.find();
        request.setAttribute("users", users);
        var id = convertId(request.getParameter("id"));
        var user = service.get(id)
                .orElseThrow();
        request.setAttribute("user", user);
    }
    @Controller(method="GET", path="/user/delete", forward="/user.jsp")
    public void deleteUser(HttpServletRequest request,
            HttpServletResponse response) {
        var id = convertId(request.getParameter("id"));
        service.delete(id);
        var users = service.find();
        request.setAttribute("users", users);
    }
    @Controller(method="POST", path="/user", forward="/user.jsp")
    public void createOrModifyUser(HttpServletRequest request,
            HttpServletResponse response) {
        var id = convertId(request.getParameter("id"));
        var name = request.getParameter("name");
        var age = convertAge(request.getParameter("age"));
        service.get(id)
            .map(u->service.update(id,name,age))
            .orElseGet(()->service.create(id,name,age))
            .orElseThrow();
        var users = service.find();
        request.setAttribute("users", users);
    }
    private Long convertId(String id){
        return Long.parseLong(id);
    }
    private Integer convertAge(String age){
        return Integer.parseInt(age);
    }
}
```

架構實作

RESTful

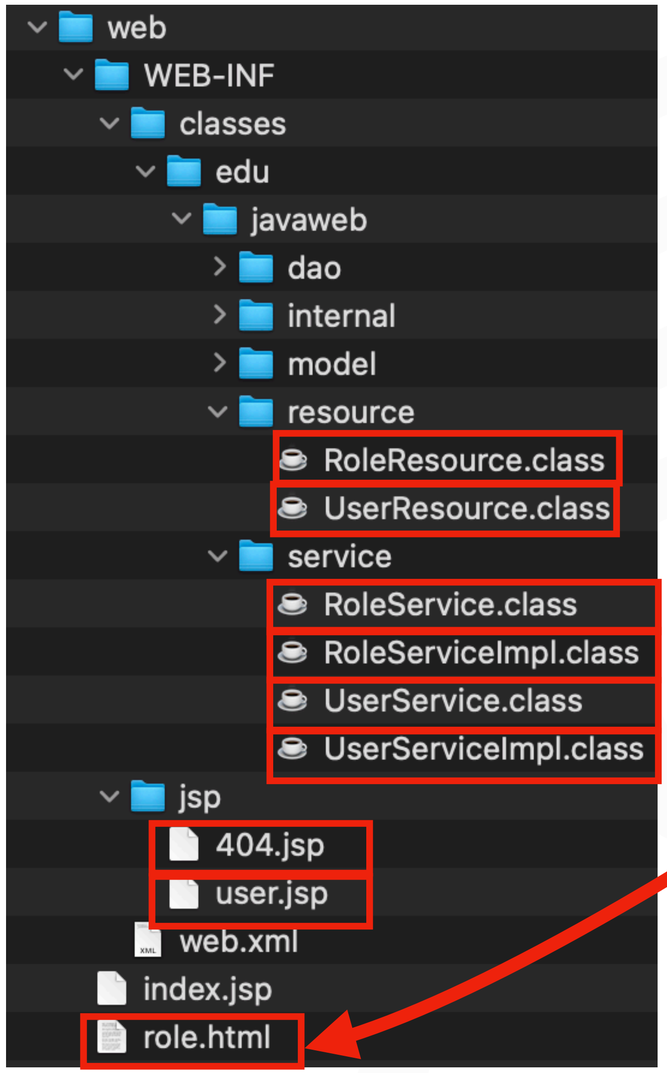



```java
package edu.javaweb.service;
import java.util.*;
import edu.javaweb.model.*;

public interface UserService {
    public Optional<User> create(Long id, String name, Integer age);
    public void delete(Long id);
    public Optional<User> update(Long id, String name, Integer age);
    public Optional<User> get(Long id);
    public List<User> find();
}
```

```java
package edu.javaweb.service;
import edu.javaweb.internal.*;
import edu.javaweb.model.*;
import edu.javaweb.dao.*;
import java.util.*;

@Component
public class UserServiceImpl implements UserService{
    @Autowire
    private UserDao dao;

    public Optional<User> create(Long id, String name, Integer age){
        var u = new User();
        u.setId(id);
        u.setName(name);
        u.setAge(age);
        return dao.create(u);
    }
    public void delete(Long id){
        dao.delete(id);
    }
    public Optional<User> update(Long id, String name, Integer age){
        var u = new User();
        u.setName(name);
        u.setAge(age);
        return dao.update(id, u);
    }
    public Optional<User> get(Long id){
        return dao.get(id);
    }
    public List<User> find(){
        return dao.find();
    }
}
```

架構實作

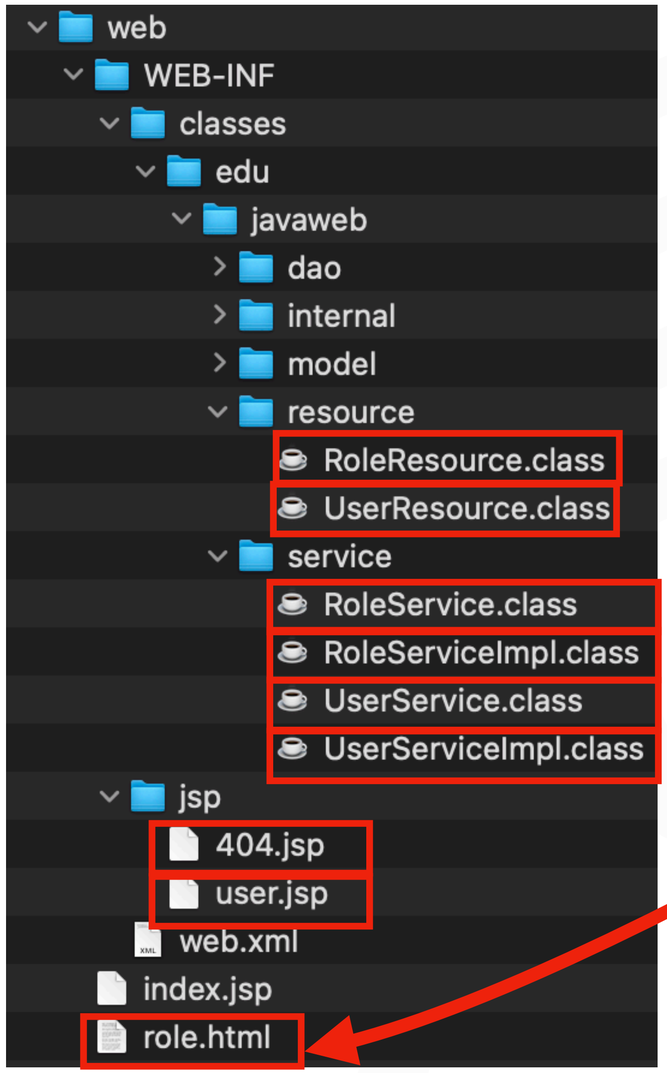
RESTful




```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Role Information</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            margin: 20px;
        }
        table {
            width: 100%;
            border-collapse: collapse;
        }
        th, td {
            border: 1px solid #ccc;
            padding: 10px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
```

架構實作

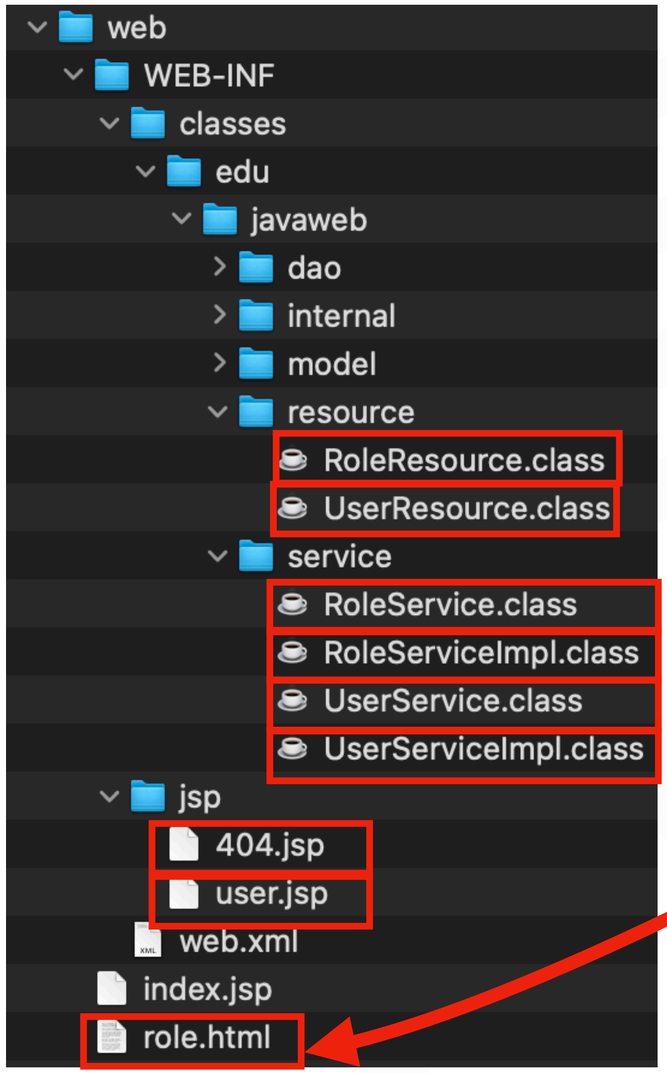
RESTful




```html
<body>
    <h1>Role Information</h1>

    <!-- Link to Home Page -->
    <a href="/web">回首頁</a>

    <!-- Role Form -->
    <form id="roleForm">
        <label for="roleId">ID:</label>
        <input type="text" name="roleId" id="roleId"><br>

        <label for="roleName">Name:</label>
        <input type="text" name="roleName" id="roleName"><br>

        <input type="button" value="Submit" onclick="addRole()">
    </form>

    <!-- Role Table -->
    <table id="roleTable">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody id="roleTableBody">
            <!-- JavaScript will populate this section -->
        </tbody>
    </table>
```

架構實作

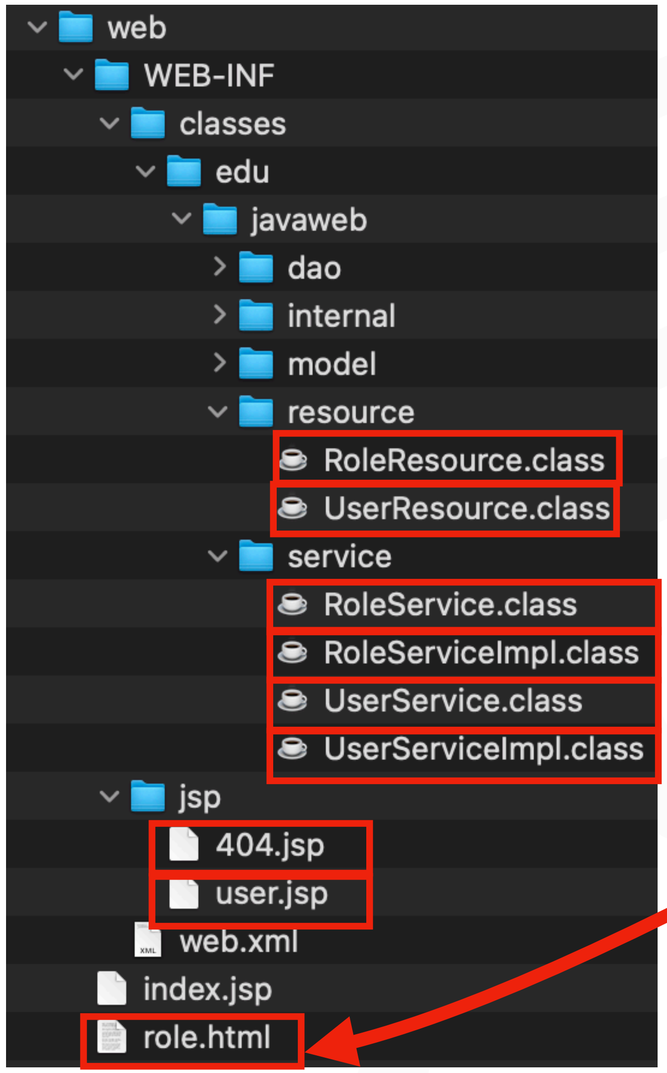
RESTful




```
<script>
    // Function to delete a role
    function deleteRole(id) {
        fetch(`/web/mvc/role?id=${id}`, {
            method: 'DELETE'
        })
        .then(response => {
            if (response.ok) {
                // Remove the role from the table
                const row = document.getElementById(`role-${id}`);
                row.remove();
            } else {
                console.error('Failed to delete role');
            }
        })
        .catch(error => {
            console.error('Error deleting role:', error);
        });
    }
```

架構實作

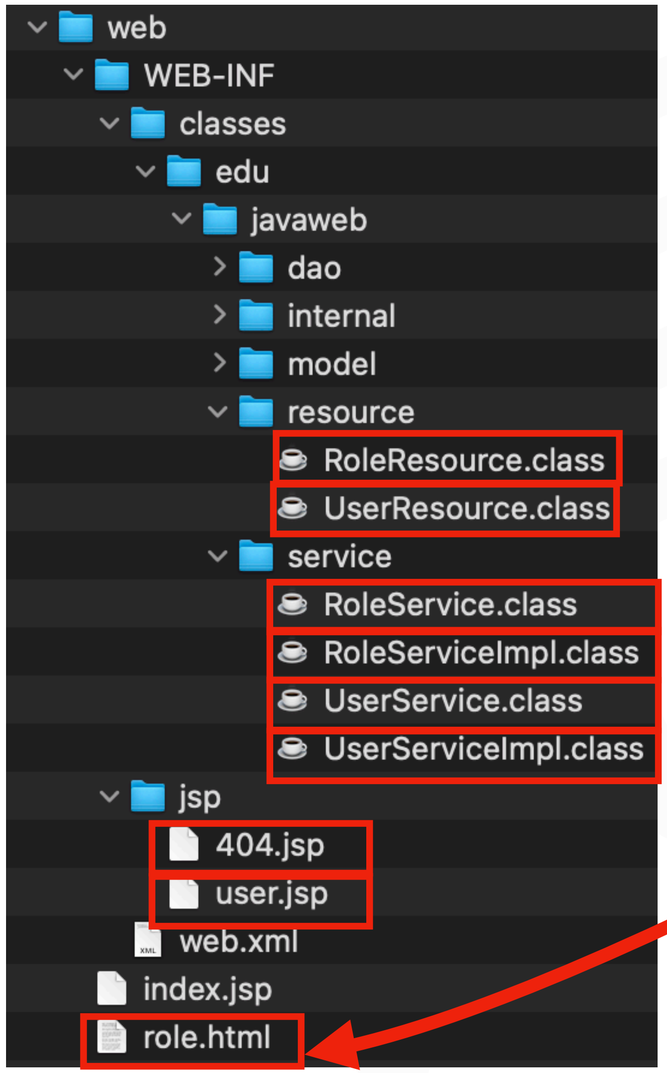
RESTful




```
// Function to edit a role name
function editRole(id) {
    const cellName = document.getElementById(`role-name-${id}`);
    const originalName = cellName.textContent;

    // Create an input element
    const input = document.createElement('input');
    input.type = 'text';
    input.value = originalName;

    // Replace cell content with input element
    cellName.innerHTML = '';
    cellName.appendChild(input);
    input.focus();
    // Handle focus out event
    input.addEventListener('blur', async () => {
        const newName = input.value;

        // Send PUT request to update role name
        try {
            const response = await fetch(`/web/mvc/role?id=${id}`, {
                method: 'PUT',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ name: newName })
            });

            if (response.ok) {
                const updatedRole = await response.json();
                console.log('Role updated:', updatedRole);

                // Update cell content with new name
                cellName.textContent = updatedRole.name;
            } else {
                console.error('Failed to update role');
                cellName.textContent = originalName;
            }
        } catch (error) {
            console.error('Error updating role:', error);
            cellName.textContent = originalName;
        }
    });
}
```

架構實作

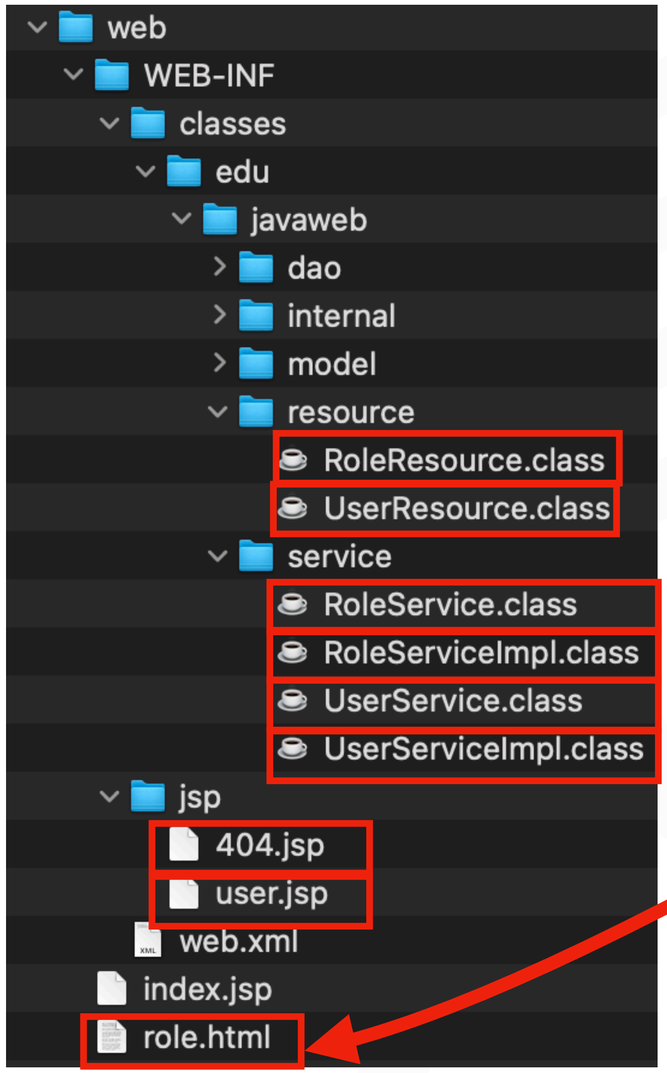
RESTful




```
        // Function to add a new role
        async function addRole() {
            const id = document.getElementById('roleId').value;
            const name = document.getElementById('roleName').value;

            const newRole = {
                id: parseInt(id, 10),
                name: name
            };

            try {
                // Send POST request to create new role
                const response = await fetch('/web/mvc/role', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify(newRole)
                });

                if (response.ok) {
                    const createdRole = await response.json();
                    console.log('Role created:', createdRole);

                    // Clear the form
                    document.getElementById('roleId').value = '';
                    document.getElementById('roleName').value = '';

                    // Reload the role list
                    loadRoles();
                } else {
                    console.error('Failed to create role');
                }
            } catch (error) {
                console.error('Error creating role:', error);
            }
        }
```

架構實作

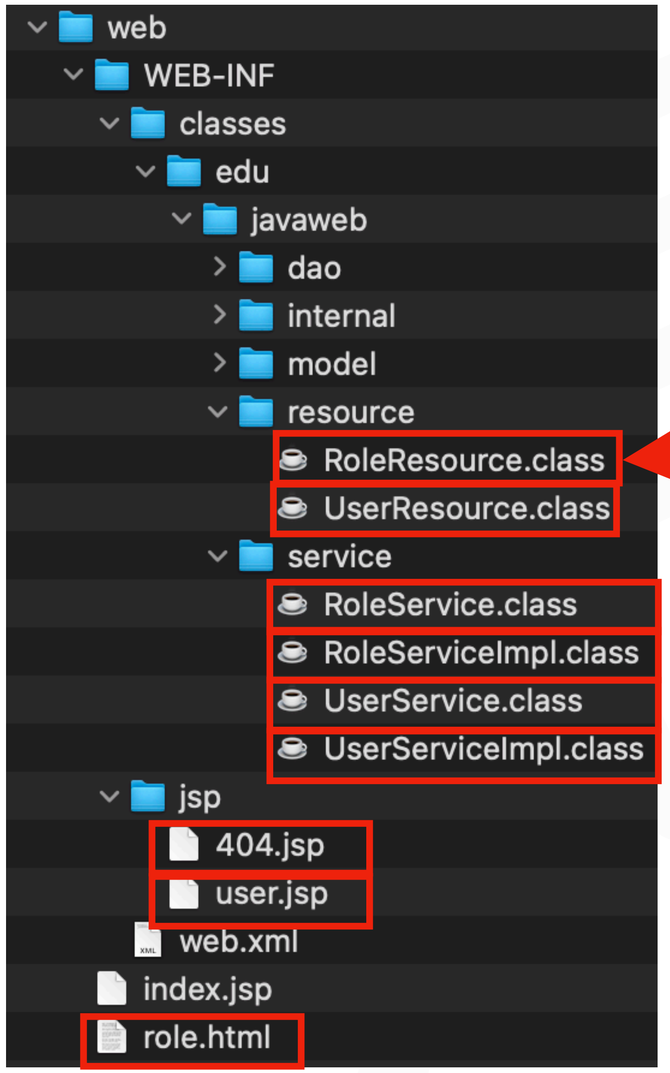
RESTful




```
        // Function to load roles into the table
        async function loadRoles() {
            try {
                const response = await fetch('/web/mvc/role');
                const data = await response.json();
                const roles = data.elements;
                const tableBody = document.getElementById('roleTableBody');

                // Clear existing rows
                tableBody.innerHTML = '';

                roles.forEach(role => {
                    const row = tableBody.insertRow();
                    row.id = `role-${role.id}`;
                    const cellId = row.insertCell(0);
                    const cellName = row.insertCell(1);
                    const cellAction = row.insertCell(2);

                    cellId.textContent = role.id;
                    cellName.textContent = role.name;

                    // Add a unique ID to the name cell
                    cellName.id = `role-name-${role.id}`;

                    cellAction.innerHTML = `
                        <a href="javascript:editRole(${role.id})">修改</a> |
                        <a href="javascript:deleteRole(${role.id})">刪除</a>
                    `;
                });
            } catch (error) {
                console.error('Error fetching roles:', error);
            }
        }

        // Call loadRoles on page load
        window.addEventListener('load', () => {
            loadRoles();
        });

    </script>
</body>
</html>
```

架構實作

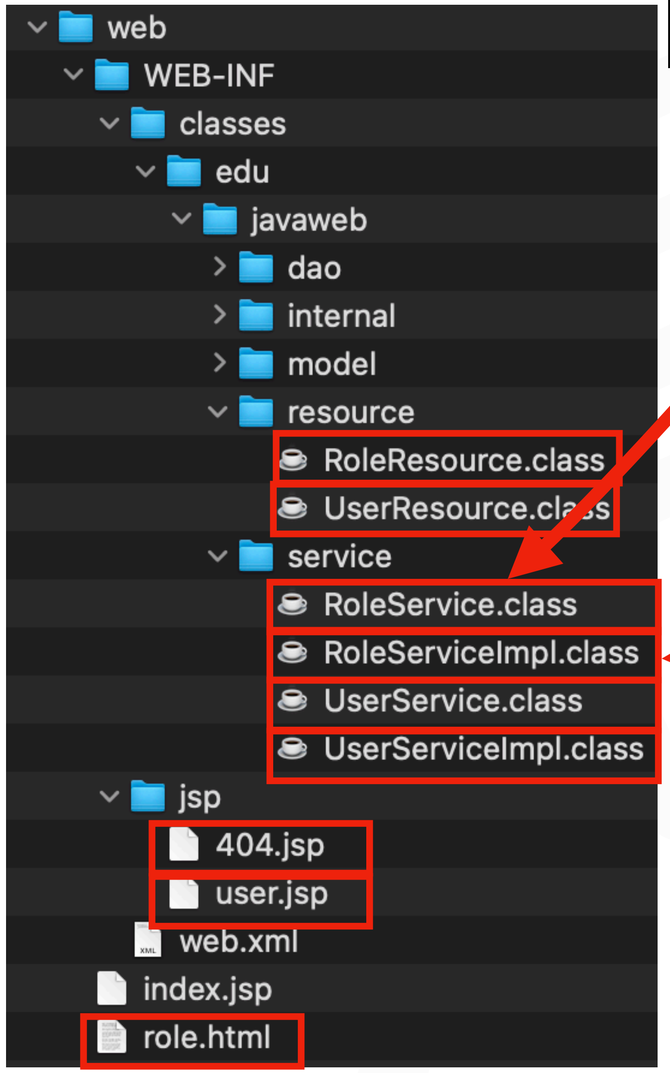
RESTful




```java
package edu.javaweb.resource;
import edu.javaweb.internal.*;
import edu.javaweb.service.*;
import edu.javaweb.model.*;
import jakarta.servlet.http.*;
import java.util.*;

@Component
public class RoleResource implements Resource {
    @Autowire
    private RoleService service;

    @Controller(method="GET", path="/role", forward="/json.jsp")
    public void getRole(HttpServletRequest request,
            HttpServletResponse response) {
        var roles = service.find();
        var json = Map.of("elements",roles, "total", roles.size());
        request.setAttribute("json", json);
    }
    @Controller(method="POST", path="/role", forward="/json.jsp")
    public void createRole(HttpServletRequest request,
            HttpServletResponse response) {
        var role = Json.as(request, Role.class);
        var r = service.create(role)
                    .orElseGet(()->null);
        request.setAttribute("json", r);
    }
    @Controller(method="PUT", path="/role", forward="/json.jsp")
    public void updateRole(HttpServletRequest request,
            HttpServletResponse response) {
        var id = request.getParameter("id");
        var role = Json.as(request, Role.class);
        var r = service.update(convertId(id),role)
                    .orElseGet(()->null);
        request.setAttribute("json", r);
    }
    @Controller(method="DELETE", path="/role", forward="/json.jsp")
    public void deleteRole(HttpServletRequest request,
            HttpServletResponse response) {
        var id = request.getParameter("id");
        service.delete(convertId(id));
        request.setAttribute("json", Map.of());
    }
    private Long convertId(String id){
        return Long.parseLong(id);
    }
}
```

架構實作

RESTful

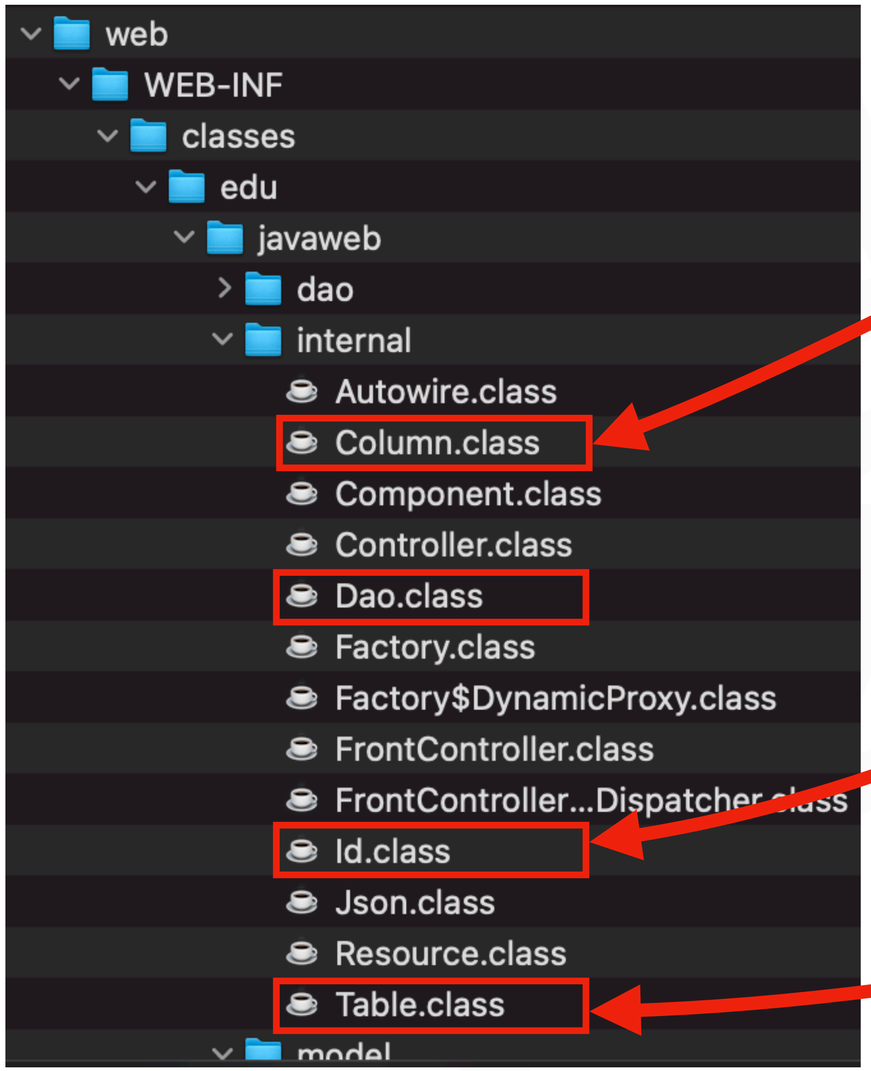



```java
package edu.javaweb.service;
import java.util.*;
import edu.javaweb.model.*;

public interface RoleService {
    public Optional<Role> create(Role r);
    public void delete(Long id);
    public Optional<Role> update(Long id, Role r);
    public Optional<Role> get(Long id);
    public List<Role> find();
}
```

```java
package edu.javaweb.service;
import edu.javaweb.internal.*;
import edu.javaweb.model.*;
import edu.javaweb.dao.*;
import java.util.*;

@Component
public class RoleServiceImpl implements RoleService{
    @Autowire
    private RoleDao dao;

    public Optional<Role> create(Role r){
        return dao.create(r);
    }
    public void delete(Long id){
        dao.delete(id);
    }
    public Optional<Role> update(Long id, Role r){
        return dao.update(id, r);
    }
    public Optional<Role> get(Long id){
        return dao.get(id);
    }
    public List<Role> find(){
        return dao.find();
    }
}
```

架構實作

Object Relational Mapping



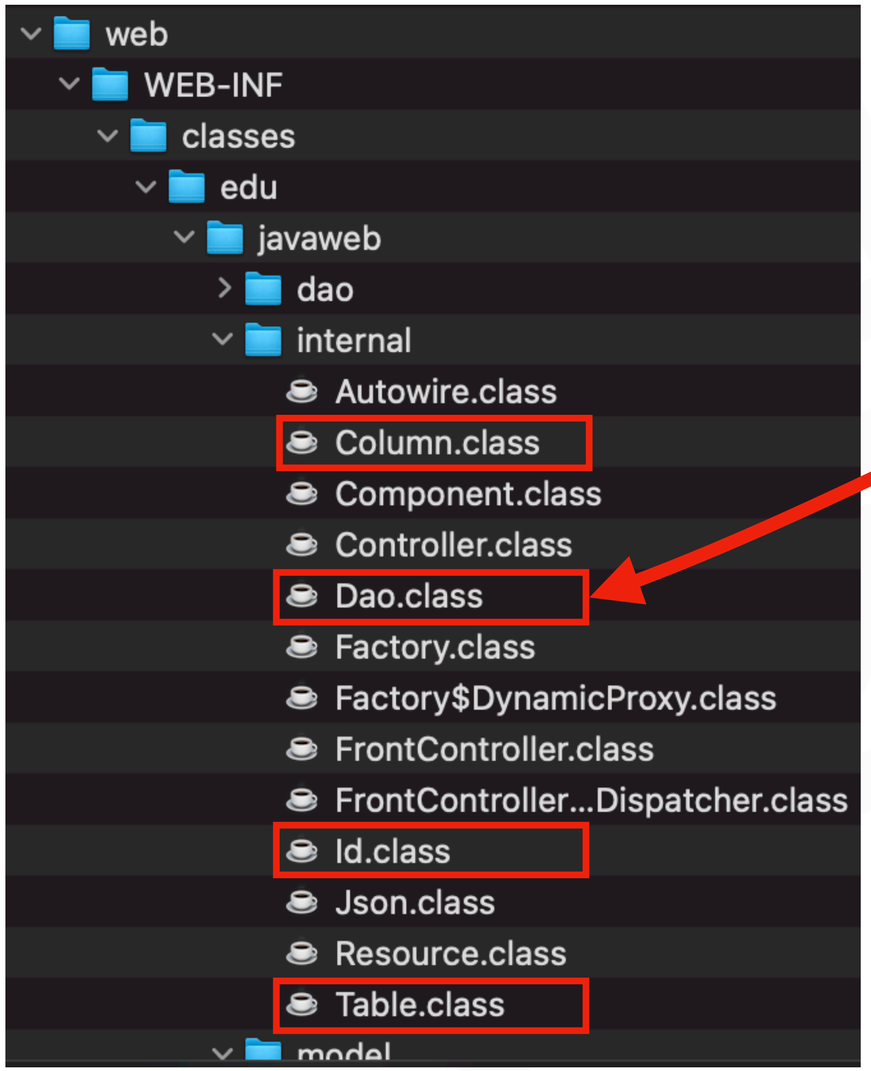

```java
package edu.javaweb.internal;
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
public @interface Column {
    String value();
}
```

```java
package edu.javaweb.internal;
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.FIELD)
public @interface Id {
}
```

```java
package edu.javaweb.internal;
import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.TYPE)
public @interface Table {
    String value();
}
```

架構實作

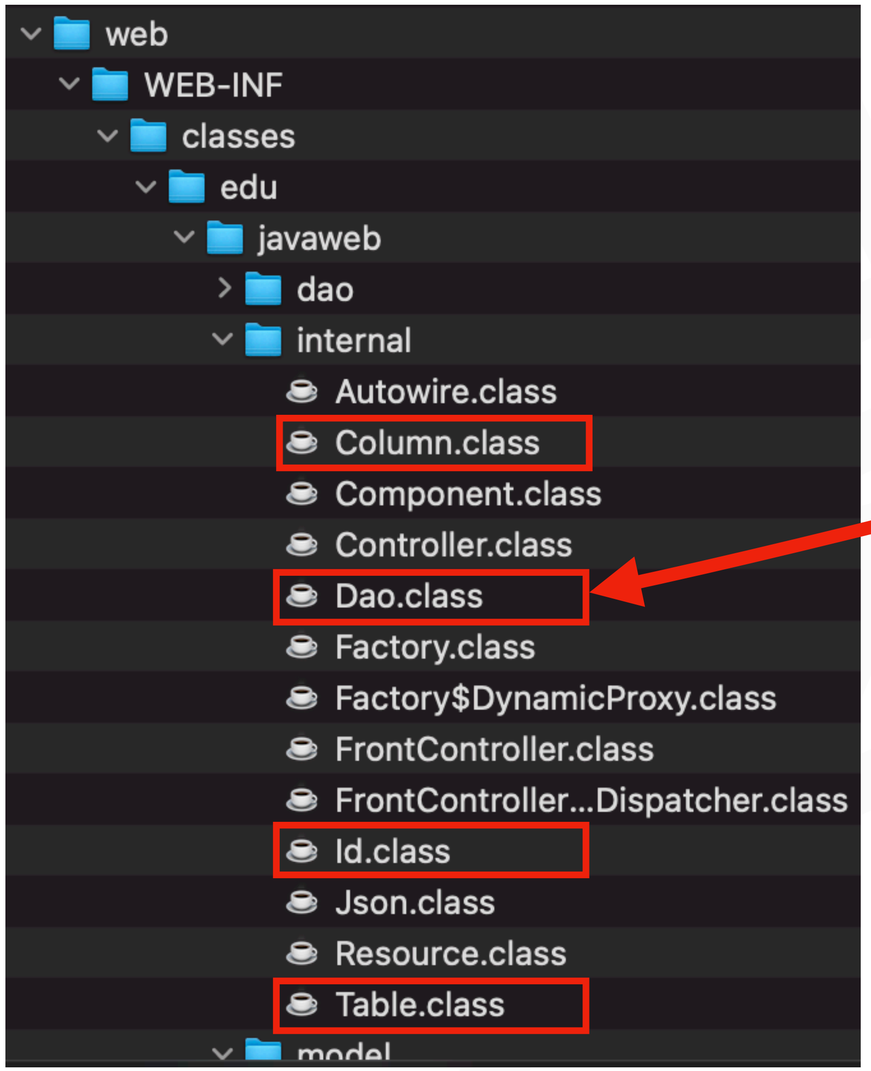
Object Relational Mapping




```java
package edu.javaweb.internal;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.lang.reflect.*;
import java.sql.*;

@SuppressWarnings("unchecked")
public interface Dao<PK, T>{
```

架構實作

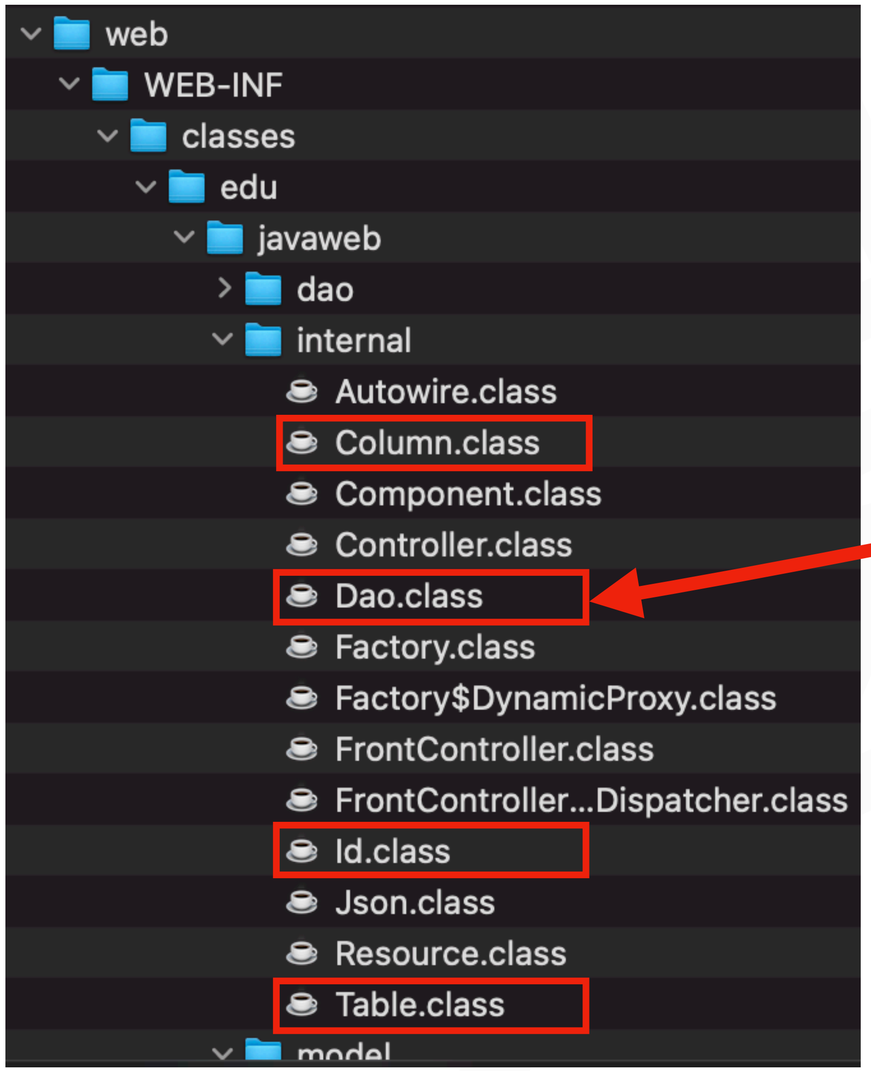
Object Relational Mapping




```java
public default Optional<T> update(PK pk, T t){
    var clazz = getModelType();
    var pkClazz = getPkType();
    var ff = Arrays.asList(clazz.getDeclaredFields());
    var k = new StringJoiner(",");
    String idColumn = null;
    var list = new ArrayList<>();
    for(var f: ff){
        f.setAccessible(true);
        var column = f.getAnnotation(Column.class);
        var id = f.getAnnotation(Id.class);
        if(Objects.isNull(id)){
            k.add(column.value()+"=?");
            try{
                list.add(f.get(t));
            } catch(Exception e){
                e.printStackTrace();
            }
        } else {
            idColumn = column.value();
        }
    }
    list.add(pk);
    var table = clazz.getAnnotation(Table.class).value();
    var sql = String.format("update %s set %s where %s=?",
                    table, k.toString(), idColumn);
    var result = connect(c->{
        try{
            var statement = c.prepareStatement(sql);
            for(var i=0;i<list.size();i++){
                statement.setObject(i+1, list.get(i));
            }
            statement.executeUpdate();
            return null;
        }catch(SQLException e){
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    });
    return get(pkClazz.cast(pk));
}
```

架構實作

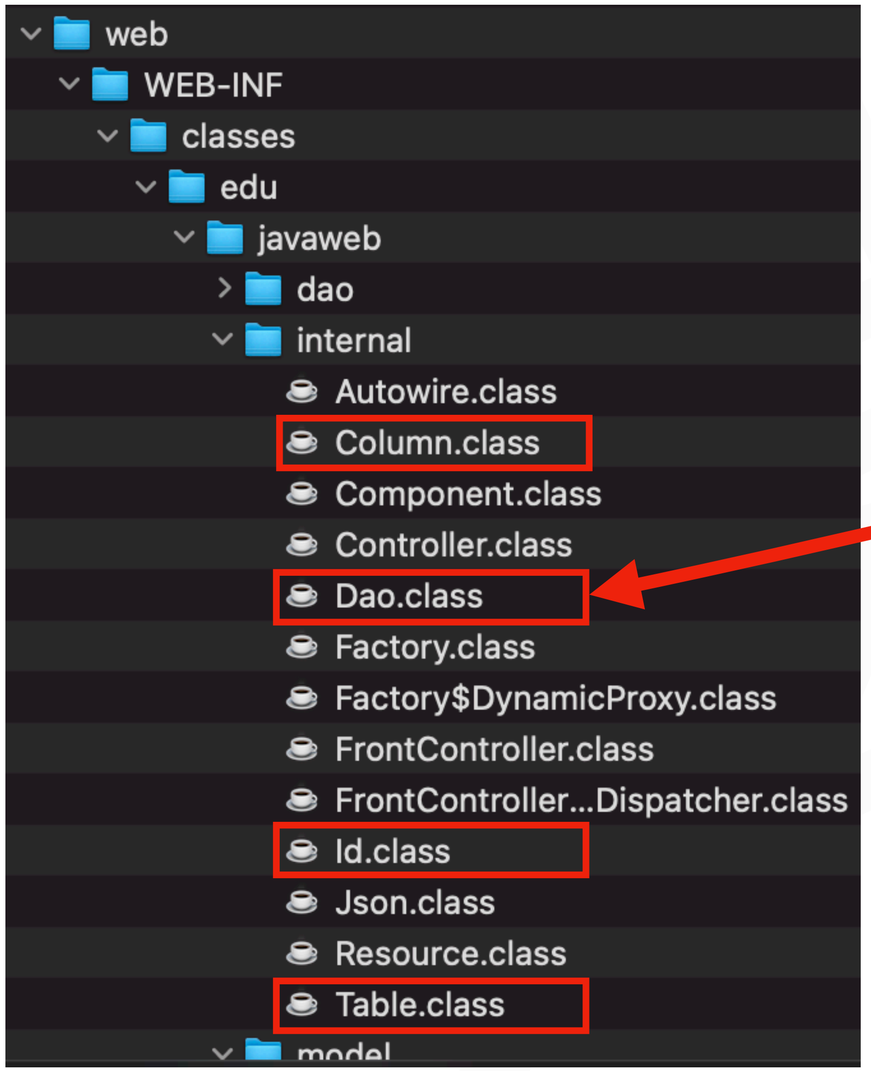
Object Relational Mapping




```
public default void delete(PK pk){
    var clazz = getModelType();
    var ff = Arrays.asList(clazz.getDeclaredFields());
    var idField = ff.stream()
                    .filter(f->Objects.nonNull(
                        f.getAnnotation(Id.class)))
                    .findFirst()
                    .orElseThrow();
    var idColumn = idField.getAnnotation(Column.class);
    var table = clazz.getAnnotation(Table.class).value();
    final Object _pk = getPkType().equals(String.class)?
                    "'"+pk.toString()+"'":pk;
    var sql = String.format("delete from %s where %s=?",table,idColumn.value());
    var result = connect(c->{
        try{
            var statement = c.prepareStatement(sql);
            statement.setObject(1, _pk);
            statement.executeUpdate();
            return null;
        }catch(SQLException e){
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    });
}
```

架構實作

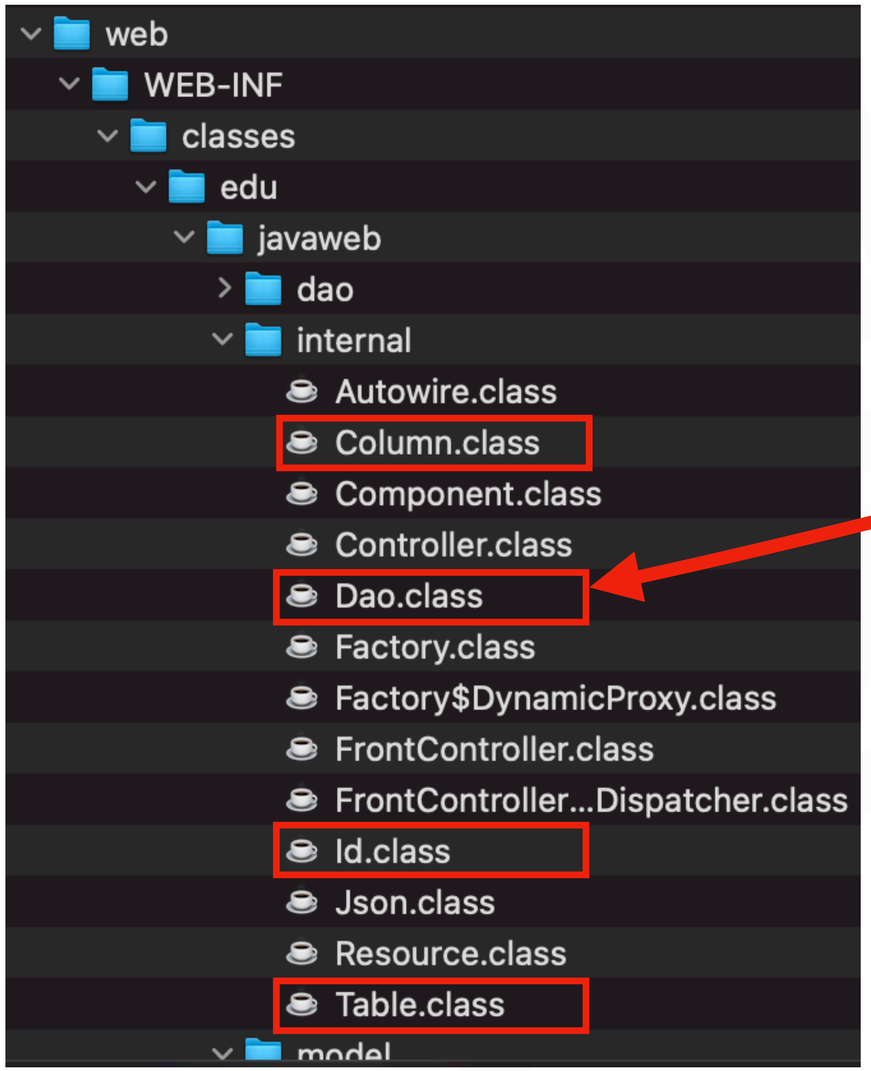
Object Relational Mapping




```java
public default Optional<T> create(T t){
    var clazz = getModelType();
    var pk = getPkType();
    var ff = Arrays.asList(clazz.getDeclaredFields());
    var k = new StringJoiner(",");
    var v = new StringJoiner(",");
    Object idValue = null;
    var list = new ArrayList<>();
    for(var f: ff){
        var c = f.getAnnotation(Column.class);
        var i = f.getAnnotation(Id.class);
        if(Objects.nonNull(c)){
            k.add(c.value());
            v.add("?");
            f.setAccessible(true);
            try{
                var o = f.get(t);
                list.add(o);
                if(Objects.nonNull(i)){
                    idValue = o;
                }
            } catch(Exception e){
                e.printStackTrace();
            }
        }
    }
    var table = clazz.getAnnotation(Table.class).value();
    var sql = String.format("insert into %s (%s) values (%s)",
                    table, k.toString(), v.toString());
    var result = connect(c->{
        try{
            var statement = c.prepareStatement(sql);
            for(var i=0;i<list.size();i++){
                statement.setObject(i+1, list.get(i));
            }
            statement.executeUpdate();
            return null;
        }catch(SQLException e){
            e.printStackTrace();
            throw new RuntimeException(e);
        }
    });
    return get(pk.cast(idValue));
}
```

架構實作

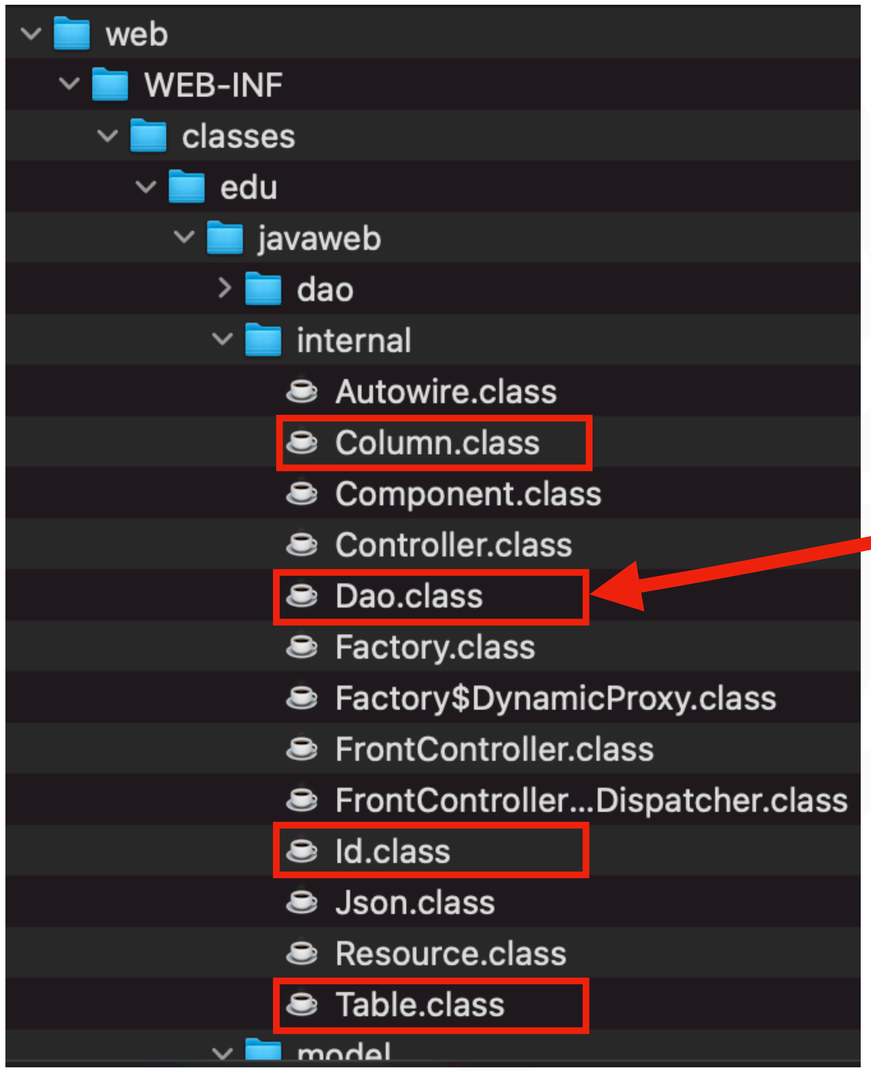
Object Relational Mapping




```
public default Optional<T> get(PK pk){
    var clazz = getModelType();
    var ff = Arrays.asList(clazz.getDeclaredFields());
    var column = ff.stream()
                   .map(f->f.getAnnotation(Column.class))
                   .filter(a->Objects.nonNull(a))
                   .map(a->a.value())
                   .reduce((s,e)->s+","+e)
                   .orElse("*");
    var idField = ff.stream()
                    .filter(f->Objects.nonNull(
                        f.getAnnotation(Id.class)))
                    .findFirst()
                    .orElseThrow();
    var idColumn = idField.getAnnotation(Column.class);
    var t = clazz.getAnnotation(Table.class).value();
    final Object _pk = getPkType().equals(String.class)?
                    "'"+pk.toString()+"'":pk;
    var sql = String.format("select %s from %s where %s=?",
                     column, t, idColumn.value());
    var result = connect(c->{
        try{
            var statement = c.prepareStatement(sql);
            statement.setObject(1, _pk);
            return statement.executeQuery();
        } catch(SQLException e){
            e.printStackTrace();
            throw new RuntimeException();
        }
    }).orElseThrow();
    try{
        if(result.next()){
            var o = clazz.getConstructor()
                         .newInstance();
            ff.stream()
              .forEach(f->setValue(result,f,o));
            return Optional.ofNullable(o);
        } else {
            return Optional.empty();
        }
    } catch(Exception e){
        e.printStackTrace();
        return Optional.empty();
    }
}
```

架構實作

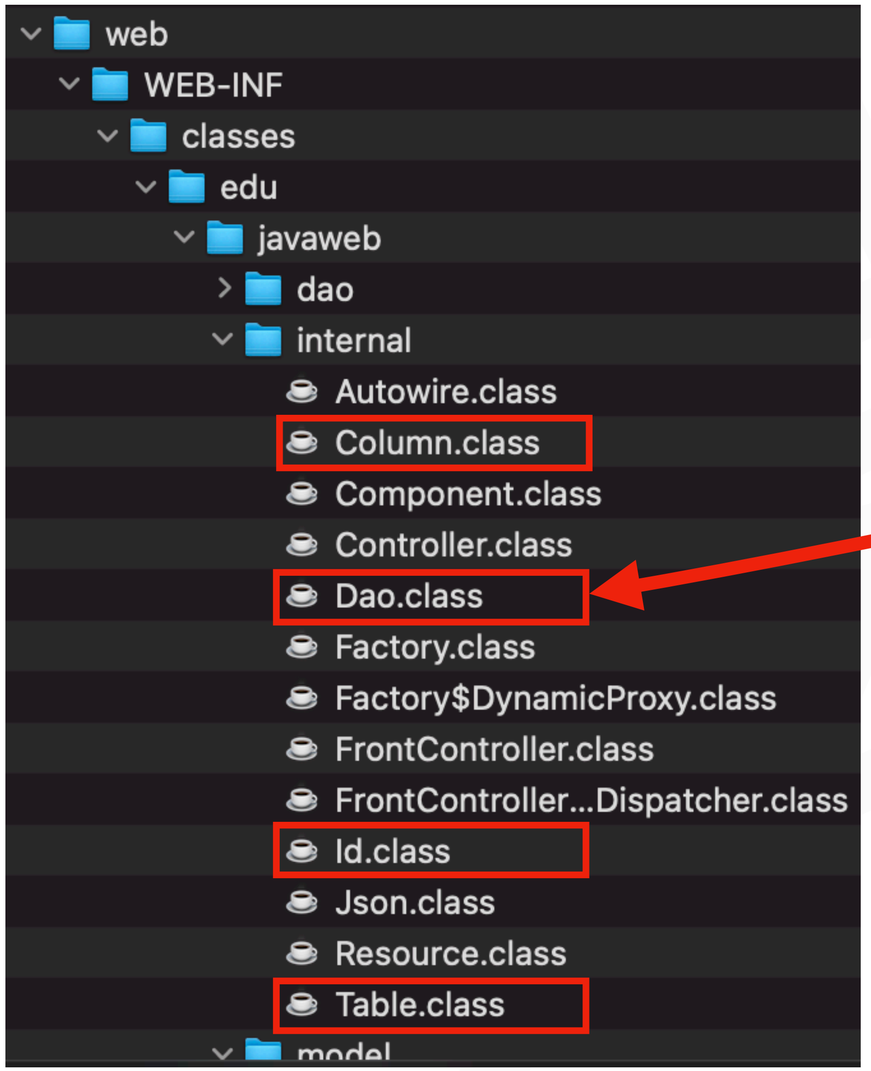
Object Relational Mapping




```
public default List<T> find(){
    var clazz = getModelType();
    var ff = Arrays.asList(clazz.getDeclaredFields());
    var column = ff.stream()
                   .map(f->f.getAnnotation(Column.class))
                   .filter(a->Objects.nonNull(a))
                   .map(a->a.value())
                   .reduce((s,e)->s+","+e)
                   .orElse("*");
    var t = clazz.getAnnotation(Table.class).value();
    var sql = String.format("select %s from %s", column, t);
    var result = connect(c->{
        try{
            var statement = c.prepareStatement(sql);
            return statement.executeQuery();
        } catch(SQLException e){
            e.printStackTrace();
            throw new RuntimeException();
        }
    }).orElseThrow();
    List<T> r = new CopyOnWriteArrayList<>();
    try{
        while(result.next()){
            var o = clazz.getConstructor()
                         .newInstance();
            ff.stream()
              .forEach(f->setValue(result,f,o));
            r.add(o);
        }
    } catch(Exception e){
        e.printStackTrace();
    }
    return r;
}
```

架構實作

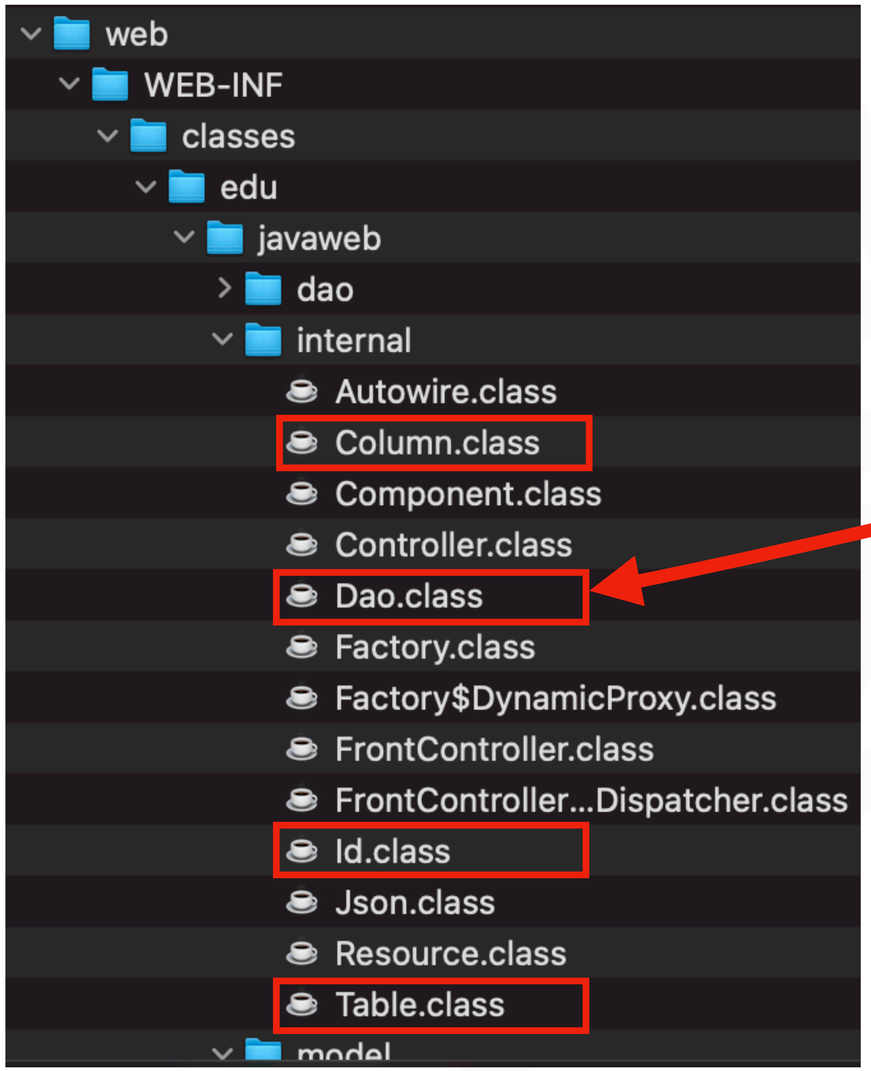
Object Relational Mapping




```
private void setValue(ResultSet result, Field f, Object model){
    try{
        var a = f.getAnnotation(Column.class);
        var v = result.getObject(a.value());
        f.setAccessible(true);
        f.set(model, v);
    } catch(Exception e){
        e.printStackTrace();
    }
}
private Optional<ResultSet> connect(Function<Connection, ResultSet> f) {
    try{
        var conn = DriverManager.getConnection(
              "jdbc:postgresql://localhost:5432/web","postgres","1qaz2wsx");
        try(conn){
            conn.setAutoCommit(false);
            conn.setTransactionIsolation(Connection.TRANSACTION_READ_UNCOMMITTED);
            var result = f.apply(conn);
            conn.commit();
            return Optional.ofNullable(result);
        }catch(RuntimeException e){
            e.printStackTrace();
            conn.rollback();
        }
    }catch(SQLException e){
        e.printStackTrace();
    }
    return Optional.empty();
}
private Class<PK> getPkType(){
    var t = getGenericType();
    if(t.length==2){
        return (Class<PK>) t[0];
    } else {
        return null;
    }
}
```

架構實作

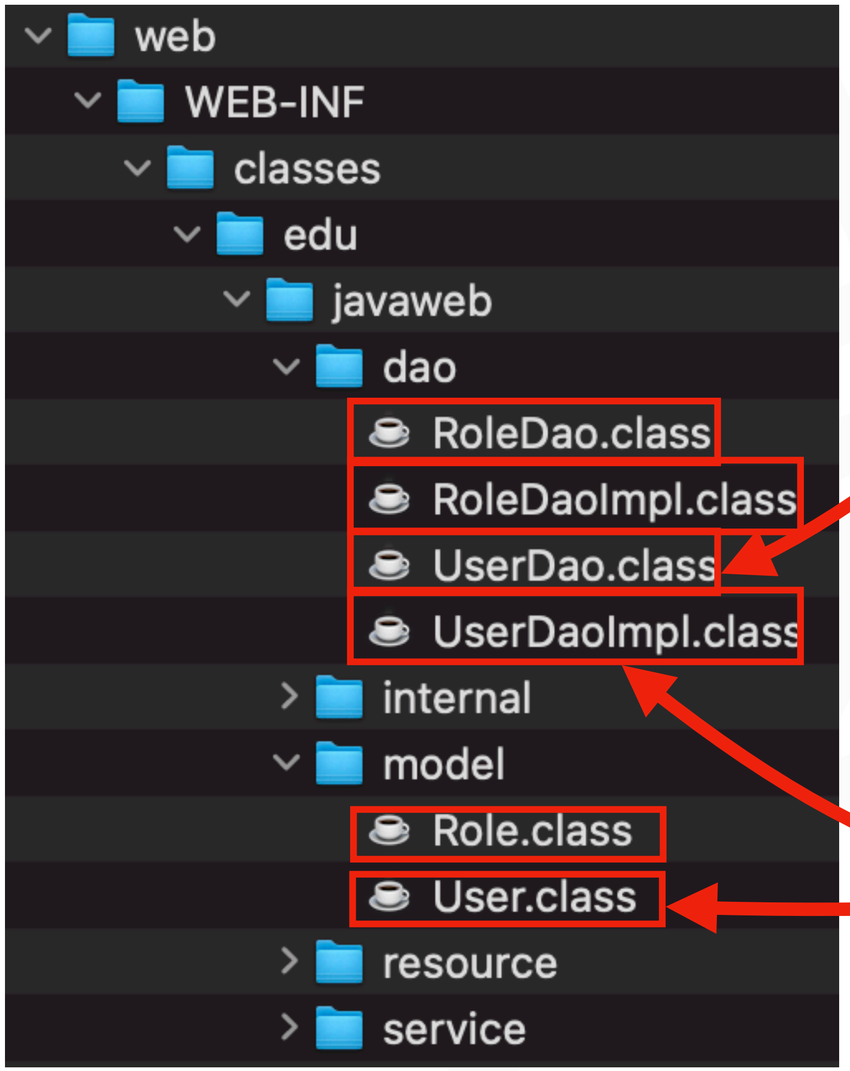
Object Relational Mapping




```
    private Class<T> getModelType(){
        var t = getGenericType();
        if(t.length==2){
            return (Class<T>) t[1];
        } else {
            return null;
        }
    }
    private Type[] getGenericType(){
        var c = getClass().getGenericInterfaces()[0];
        var type = ((Class<?>)c).getGenericInterfaces()[0];
        if (type instanceof ParameterizedType t) {
            return t.getActualTypeArguments();
        } else {
            return new Type[]{};
        }
    }
}
```

架構實作

Object Relational Mapping



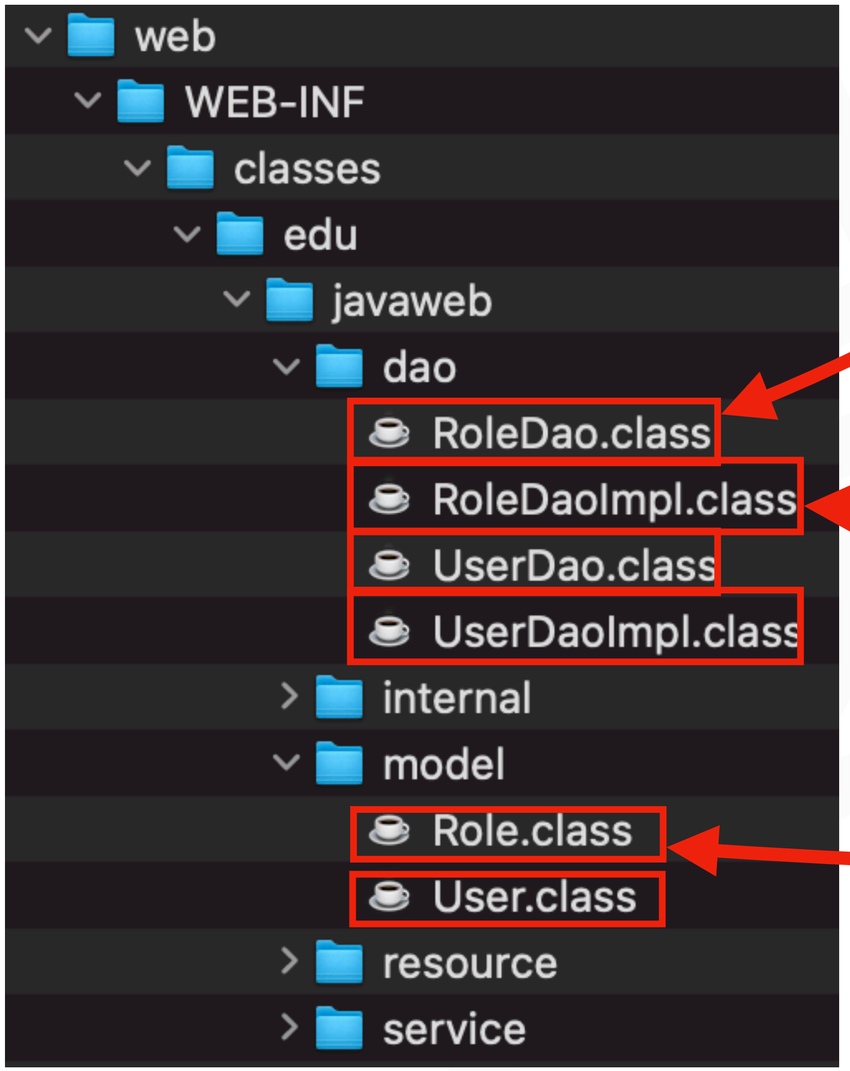

```java
package edu.javaweb.dao;
import java.util.*;
import edu.javaweb.model.*;
import edu.javaweb.internal.*;

public interface UserDao extends Dao<Long, User> {
}
```

```java
package edu.javaweb.dao;
import edu.javaweb.internal.*;
import edu.javaweb.model.*;
import java.util.*;
import java.sql.*;
import java.util.function.*;

@Component
public class UserDaoImpl implements UserDao {
}
```

```java
package edu.javaweb.model;
import edu.javaweb.internal.*;

@Table("users")
public class User {
    @Id
    @Column("id")
    private Long id;

    @Column("name")
    private String name;

    @Column("age")
    private Integer age;

    public Long getId(){
        return this.id;
    }

    public String getName(){
        return this.name;
    }

    public Integer getAge(){
        return this.age;
    }

    public void setId(Long id){
        this.id = id;
    }

    public void setName(String name){
        this.name = name;
    }

    public void setAge(Integer age){
        this.age = age;
    }

    public String toString(){
        var s = "User [id=%s, name=%s, age=%s]";
        return String.format(s, id, name, age);
    }
}
```

架構實作

Object Relational Mapping




```java
package edu.javaweb.dao;
import java.util.*;
import edu.javaweb.model.*;
import edu.javaweb.internal.*;

public interface RoleDao extends Dao<Long, Role> {
}
```

```java
package edu.javaweb.dao;
import edu.javaweb.internal.*;
import edu.javaweb.model.*;
import java.util.*;
import java.sql.*;
import java.util.function.*;

@Component
public class RoleDaoImpl implements RoleDao {
}
```

```java
package edu.javaweb.model;
import edu.javaweb.internal.*;

@Table("role")
public class Role {
    @Id
    @Column("id")
    private Long id;

    @Column("name")
    private String name;

    public Long getId(){
        return this.id;
    }

    public String getName(){
        return this.name;
    }

    public void setId(Long id){
        this.id = id;
    }

    public void setName(String name){
        this.name = name;
    }

    public String toString(){
        var s = "Role [id=%s, name=%s]";
        return String.format(s, id, name);
    }
}
```

Java Web

應用與實務

- 開發工具
  - maven, gradle
    - nexus repository
  - git & git flow
  - liquibase
  - jenkins
  - docker
- 測試工具
  - TDD
    - junit, mockito
  - BDD
    - cucumber
  - jmeter
- 專案工具
  - 專案管理
    - 敏捷開發
    - agile, scrum
  - issue tracking
    - Redmine, JIRA, Trello