# Spring MVC

## Spring MVC

### The MVC pattern

* Mẫu MVC chia chia ứng dụng thành 3 phần:
* **Model**: The domain object model / service layer
* **View**: Template code / markup
* **Controller**: Presentation logic / action classes
* MVC định nghĩa sự tương tác giữa các thành phần để thúc đẩy sự tách biệt các quan hệ và sự kết hợp lỏng lẻo
* Mỗi file có một trách nhiệm.
* Cho phép phân chia lao động giữa lập trình viên và nhà thiết kế.
* Tạo điều kiện cho unit testing.
* Dễ dàng để hiểu, thay đổi và debug.

### Những lợi ích của mẫu MVC

Sự phân chia logic ứng dụng và thiết kế web thông qua mẫu MVC

* Tích hợp với template languages.
* Một số MVC frameworks cung cấp built-in components.
* Những lợi ích khác bao gồm:
* Form validation
* Error handling
* Request parameter type conversion
* Internationalization
* IDE integration

### Spring's Web MVC Framework

* **Spring Web MVC framework** được xây dựng dựa trên **generic Servlet** mà được biết như là **DispatcherServlet** class (**Front Controller**).
* Lớp **DispatcherServlet** gửi yêu cầu đến handlers với các configurable handler mappings, theme resolution, locale và view resolution cùng với file uploading.
* **handleRequest(request, response) method** được cung cấp bởi **default handler** được gọi là **Controller interface**.
* Những **application controller implementation classes** của **controller interface**:
* AbstractController
* AbstractCommandController
* SimpleFormController

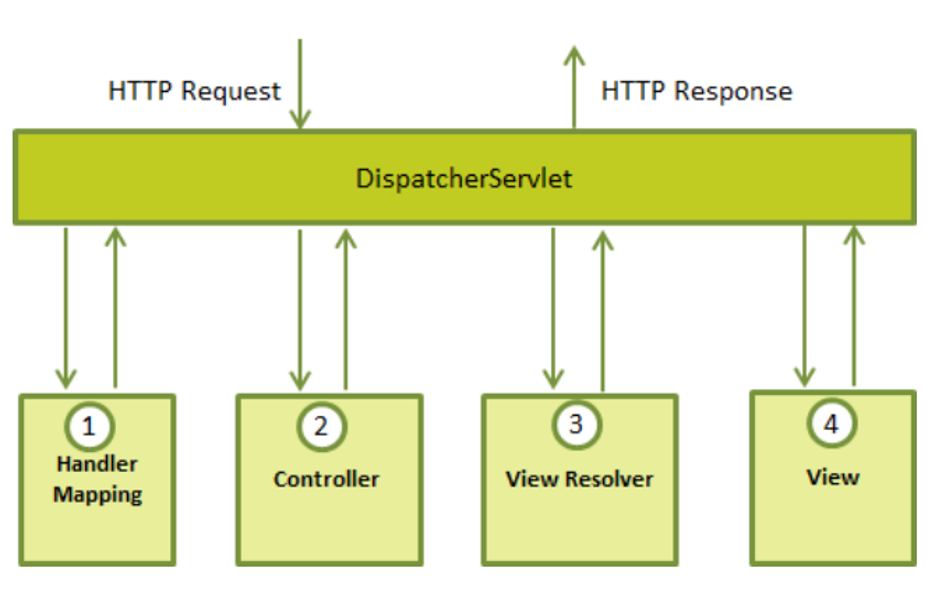
### Những tính năng của Spring MVC

* **Sự cấu hình mạnh mẽ** của cả framework và application classes.
* **Phân chia những vai trò**.
* **Linh loạt** trong việc chọn những subclasses.
* **Model transfer flexibility**.
* **Không cần duplication code**.
* **Validation** và **binding** một cách rõ ràng.
* **Specific local và theme resolution.**
* **Những tiện ích** của **JSP** từ **tag library**.

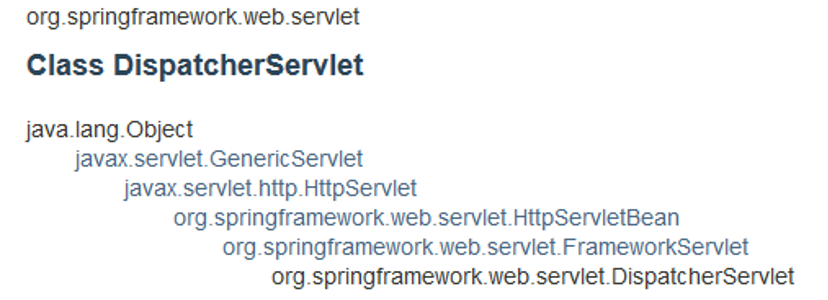
### Spring Web MVC

* Spring Web MVC là original web framework được xây dựng dựa trên Servlet API và đã được bao gồm trong Spring Framework ngay từ đầu.
* Spring Web MVC được biết đến phổ biến hơn là **Spring MVC**
* Các khái niệm trong Spring Web MVC:
* DispatcherServlet
* Filters
* Annotated Controllers
* Functional Endpoints
* URI Links
* Asynchronous Requests
* CORS
* Error Responses
* Web Security
* HTTP Caching
* View Technologies
* MVC Config
* HTTP/2

### DispatcherServlet



* Sau khi nhận một HTTP request, **DispatcherServlet** sẽ dựa vào **HandlerMapping** để gọi **Controller** phù hợp.
* **Controller** nhận **request đó**, gọi đến các **class service** thích hợp để xử lý yêu cầu. Sau khi xử lý xong Controller sẽ nhận được **model** từ tầng service. Controller gửi **model** vừa nhận được đến **DispatcherServlet**.
* DispatcherSerlet dựa vào **View Resolver** để tìm ra **view thích hợp** và truyền **model** vào nó.
* **View hoàn chỉnh được built** và DispatcherSerlet gửi nó cho người dùng.



**DispatcherServlet class** là phần quan trọng của spring Web MVC framework. Nó được sử dụng để **dispatching request** đến những **controller**.

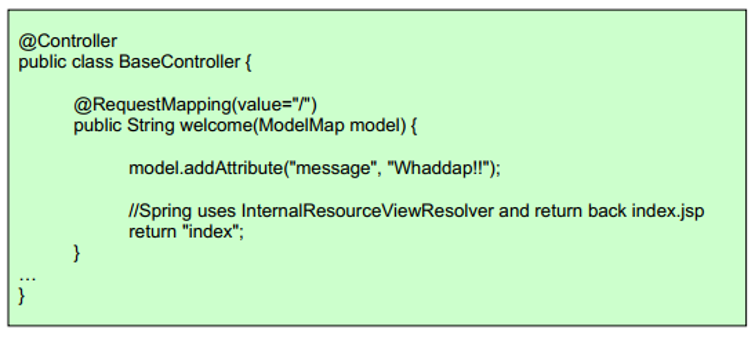
DispatcherServlet class được cấu hình trong file **web.xml** của một web app.

### HandlerMapping/Controller/ViewResolver

* **Handler mapping**: Quản lý sự thực thi của những controller, miễn là chúng phù hợp với các tiêu chí đã chỉ định.
* **Controller**: Xử lý các request của client.
* **View resolver**: giải quyết view names thành view được sử dụng bởi DispatcherServlet.
* Việc ánh xạ giữa **Logical name** và **Physical View Location** được xử lý bởi **View Resolver object**.
* Spring đi kèm với một bộ **Built-In Spring Resolvers**.
* Chúng ta có thể viết một Custom View Resolvers bằng cách implementing org.springframework.web.servlet.ViewResolver interface

### Controllers

* Những controllers là những thành phần trung tâm của MVC.
* Để đánh dấu một class là controller ta thêm **@Controller** annotation trên class đó và sử dụng **@RequestMapping** để map method trong class đến url:



* **@Controller** annotation định nghĩa class như một **Spring MVC controller**.
* **@RequestMapping** annotation được sử dụng để map URLs như ‘/hello’ trên **toàn bộ class** hoặc trên **một phương thức cụ thể** của class.
* **@RequestMapping(method = RequestMethod.GET)** được sử dụng để khai báo một phương thức như là controller's default service method để xử lý **HTTP GET request**.

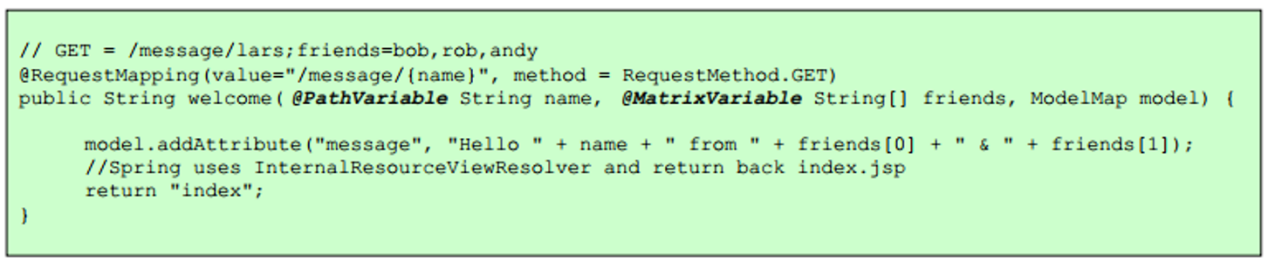
### Controller arguments

* @RequestParam, @PathVariable
* @Valid
* Model, ModelMap, Map<String, ?>
* @RequestHeader, @RequestBody

### Url Templates trong Controllers

**@PathVariable** - để map những biến trong URL paths.

**path variables** cũng có thể là biểu thức chính qui.



### Model

Controllers và view chia sẻ một đối tượng Java được tham chiếu như là **model**, (M trong MVC).

**Controller**:

@GetMapping("/hello")

public String hello(Model model) {

model.addAttribute("ten", "Nguyễn Văn A");

return "hello";

}

**HTML**:

<span th:text="${ten}">Giá trị mặc định</span>

### @ModelAttribute trong Controller

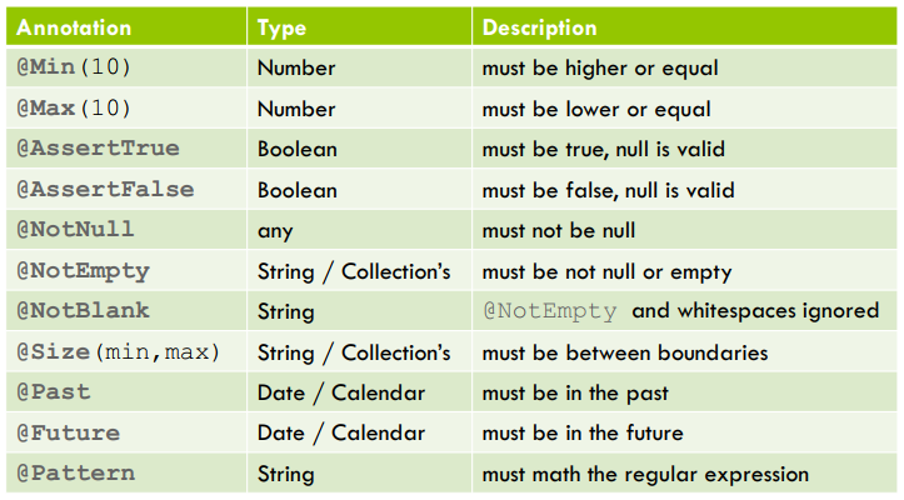
Ta có thể sử dụng **@ModelAttribute** trong controller để **load trực tiếp URL value** vào trong model.

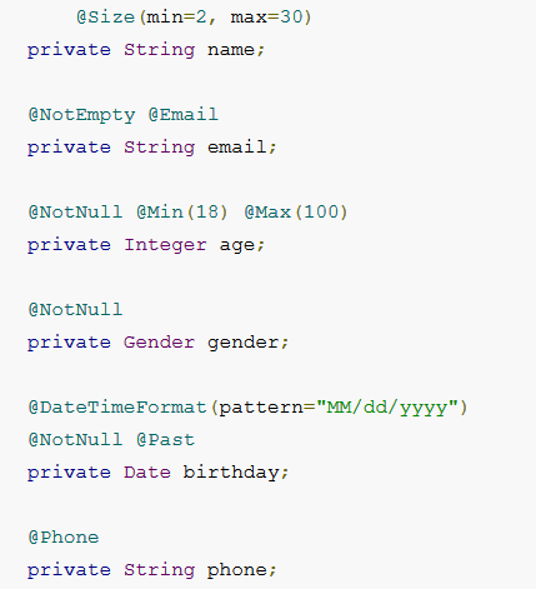
Một **Model** có thể đại diện cho những **objects** mà có thể được lấy ra từ **database** hoặc **file**.

Model **không nên có logic**, đúng hơn là controller nên nhận model và “transform” model dựa vào request, trong khi gửi nó đến view.

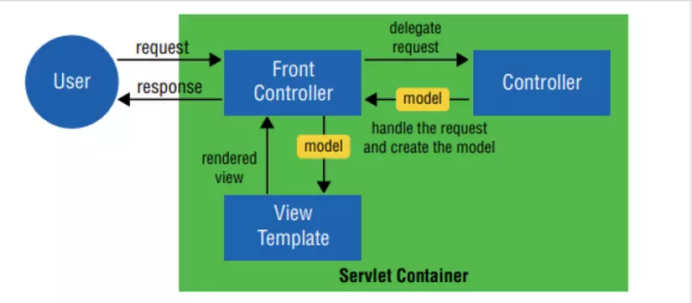
### Spring Validation

* Bean Validation:
  + Standard Constraints:





### Flow trong Spring MVC:

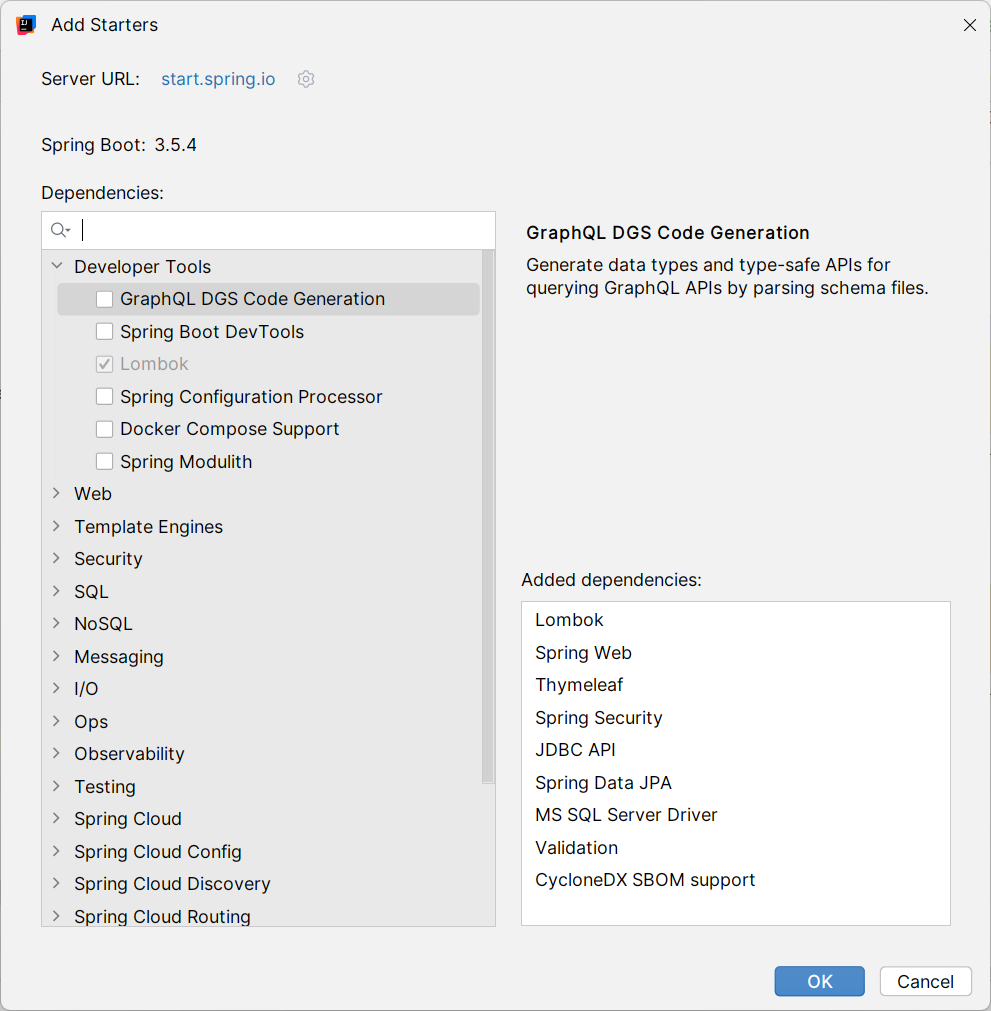


* Bất kỳ request nào tới ứng dụng web đều sẽ được gửi tới Front Controller (Dispatcher Servlet)
* Front Controller sẽ sử dụng Handler Mapping để biết được controller nào sẽ xử lý request đó
* Controller nhận request, gọi tới các class service thích hợp để xử lý yêu cầu.
* Sau khi xử lý xong, Controller sẽ nhận được model từ tầng Service hoặc tầng DAO.
* Controller gửi model vừa nhận được tới Front Controller (Dispatcher Servlet)
* Dispatcher Servlet sẽ tìm các mẫu view, sử dụng view resolver và truyền model vào nó.
* View template, model, view page được build và gửi trả lại Front Controller
* Front Controller gửi một page view tới trình duyệt để hiển thị nó cho người dùng.

## Demo SpringMVC

- Tạo Spring Boot project: **video-collection** (Quản lý bộ sưu tập video) và chọn Maven làm công cụ quản lý project

- Chọn các **dependencies** sau:



<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.microsoft.sqlserver</groupId>

<artifactId>mssql-jdbc</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-validation</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-test</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.thymeleaf.extras</groupId>

<artifactId>thymeleaf-extras-springsecurity6</artifactId>

</dependency>

</dependencies>

- Cấu hình dự án trong file **application.properties**:

spring.application.name=video-collection

server.port=6969

server.servlet.context-path=/

spring.datasource.url=jdbc:sqlserver://BTYLAP:1433;databaseName=VideoCollectionDB;encrypt=true;trustServerCertificate=true

spring.datasource.username=sa

spring.datasource.password=123

spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver

spring.jpa.hibernate.ddl-auto=create

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

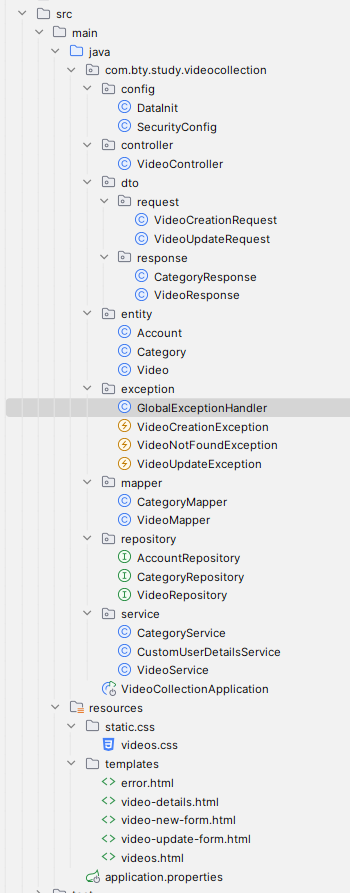
spring.jpa.properties.hibernate.physical\_naming\_strategy = org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

spring.thymeleaf.cache=false

spring.thymeleaf.prefix=classpath:/templates/

spring.thymeleaf.suffix=.html

- Tạo cấu trúc dự án như sau:



- Tạo 2 entity **Category** và **Video**:

+ **Category**:

package com.bty.study.videocollection.entity;

import jakarta.persistence.\*;

import lombok.\*;

import lombok.experimental.FieldDefaults;

import org.hibernate.annotations.Check;

import java.util.ArrayList;

import java.util.List;

@Entity

@Table(name = "Category")

@NoArgsConstructor

@AllArgsConstructor

@Setter

@Getter

@ToString

@FieldDefaults(level = AccessLevel.PRIVATE)

@Builder

@Check(name = "check\_len\_Name", constraints = "LEN(Name) >= 1 AND LEN(Name) <= 100")

public class Category {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "Id")

Long id;

@Column(name = "Name", columnDefinition = "NVARCHAR(100)", unique = true, nullable = false)

String name;

@Column(name = "Description", columnDefinition = "NVARCHAR(3000)", nullable = true)

String description;

@OneToMany(mappedBy = "category", fetch = FetchType.LAZY, cascade = CascadeType.ALL)

List<Video> videos = new ArrayList<>();

public void addVideo(Video video) {

videos.add(video);

video.setCategory(this);

}

public void removeVideo(Video video) {

videos.remove(video);

video.setCategory(null);

}

}

+ **Video**:

package com.bty.study.videocollection.entity;

import jakarta.persistence.\*;

import lombok.\*;

import lombok.experimental.FieldDefaults;

import org.hibernate.annotations.Check;

@Entity

@Table(name = "Video")

@NoArgsConstructor

@AllArgsConstructor

@Setter

@Getter

@ToString

@FieldDefaults(level = AccessLevel.PRIVATE)

@Builder

@Check(name = "check\_len\_Title", constraints = "LEN(Title) >= 5 AND LEN(Title) <= 250")

public class Video {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "Id")

Long id;

@Column(name = "Title", columnDefinition = "NVARCHAR(250)", unique = true, nullable = false)

String title;

@Column(name = "Description", columnDefinition = "NVARCHAR(3000)", nullable = true)

String description;

@Column(name = "Path", columnDefinition = "NVARCHAR(3000)", unique = true, nullable = false)

String path;

@ManyToOne

@JoinColumn(name = "CategoryId")

Category category;

}

- Tạo các **dto**:

+ **VideoCreationRequest**:

package com.bty.study.videocollection.dto.request;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

import lombok.\*;

import lombok.experimental.FieldDefaults;

@NoArgsConstructor

@AllArgsConstructor

@Setter

@Getter

@Builder

@FieldDefaults(level = AccessLevel.PRIVATE)

public class VideoCreationRequest {

@Size(min = 5, max = 250, message = "Min title length is 5 char and max title length is 250 char")

@NotNull(message = "Title must be not null")

String title;

String description;

@NotNull(message = "Path must be not null")

String path;

Long categoryId;

}

+ **VideoUpdateRequest**:

package com.bty.study.videocollection.dto.request;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

import lombok.\*;

import lombok.experimental.FieldDefaults;

@NoArgsConstructor

@AllArgsConstructor

@Setter

@Getter

@Builder

@FieldDefaults(level = AccessLevel.PRIVATE)

public class VideoUpdateRequest {

Long id;

@Size(min = 5, max = 250, message = "Min title length is 5 char and max title length is 250 char")

@NotNull

String title;

String description;

@NotNull

String path;

Long categoryId;

}

+ **CategoryResponse**:

package com.bty.study.videocollection.dto.response;

import lombok.\*;

import lombok.experimental.FieldDefaults;

@NoArgsConstructor

@AllArgsConstructor

@Setter

@Getter

@Builder

@FieldDefaults(level = AccessLevel.PRIVATE)

public class CategoryResponse {

Long id;

String name;

String description;

}

+ **VideoResponse**:

package com.bty.study.videocollection.dto.response;

import lombok.\*;

import lombok.experimental.FieldDefaults;

@NoArgsConstructor

@AllArgsConstructor

@Setter

@Getter

@Builder

@FieldDefaults(level = AccessLevel.PRIVATE)

public class VideoResponse {

Long id;

String title;

String description;

String path;

CategoryResponse category;

}

- Tạo 2 repository **CategoryRepository** và **VideoRepository**:

+ **CategoryRepository**:

package com.bty.study.videocollection.repository;

import com.bty.study.videocollection.entity.Category;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface CategoryRepository extends JpaRepository<Category, Long> {

}

+ **VideoRepository**:

package com.bty.study.videocollection.repository;

import com.bty.study.videocollection.entity.Video;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface VideoRepository extends JpaRepository<Video, Long> {

boolean existsByTitle(String title);

boolean existsByPath(String path);

List<Video> findAllByTitleContaining(String title);

}

- Tạo các custom **exception**:

+ **VideoCreationException**:

package com.bty.study.videocollection.exception;

public class VideoCreationException extends RuntimeException {

public VideoCreationException(String message) {

super(message);

}

}

+ **VideoNotFoundException**:

package com.bty.study.videocollection.exception;

public class VideoNotFoundException extends RuntimeException {

public VideoNotFoundException(String message){

super(message);

}

}

+ **VideoUpdateException**:

package com.bty.study.videocollection.exception;

public class VideoUpdateException extends RuntimeException {

public VideoUpdateException(String message) {

super(message);

}

}

- Tạo các **mapper**:

+ **CategoryMapper**:

package com.bty.study.videocollection.mapper;

import com.bty.study.videocollection.dto.response.CategoryResponse;

import com.bty.study.videocollection.entity.Category;

import org.springframework.stereotype.Component;

@Component

public class CategoryMapper {

public CategoryResponse toCategoryResponse(Category category){

CategoryResponse categoryResponse = CategoryResponse.builder()

.id(category.getId())

.name(category.getName())

.description(category.getDescription())

.build();

return categoryResponse;

}

}

+ **VideoMapper**:

package com.bty.study.videocollection.mapper;

import com.bty.study.videocollection.dto.request.VideoCreationRequest;

import com.bty.study.videocollection.dto.request.VideoUpdateRequest;

import com.bty.study.videocollection.dto.response.VideoResponse;

import com.bty.study.videocollection.entity.Category;

import com.bty.study.videocollection.entity.Video;

import com.bty.study.videocollection.exception.VideoCreationException;

import com.bty.study.videocollection.exception.VideoUpdateException;

import com.bty.study.videocollection.repository.CategoryRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Component;

@Component

@RequiredArgsConstructor

public class VideoMapper {

private final CategoryMapper categoryMapper;

private final CategoryRepository categoryRepository;

public VideoResponse toVideoResponse(Video video){

VideoResponse videoResponse = VideoResponse.builder()

.id(video.getId())

.title(video.getTitle())

.description(video.getDescription())

.path(video.getPath())

.category(categoryMapper.toCategoryResponse(video.getCategory()))

.build();

return videoResponse;

}

public Video toVideo(VideoCreationRequest videoCreationRequest){

Category category = null;

if(videoCreationRequest.getCategoryId() != null) {

category = categoryRepository.findById(videoCreationRequest.getCategoryId())

.orElseThrow(() -> new VideoCreationException("Category with id = " + videoCreationRequest.getCategoryId() + " not found"));

}

Video video = Video.builder()

.title(videoCreationRequest.getTitle())

.description(videoCreationRequest.getDescription())

.path(videoCreationRequest.getPath())

.category(category)

.build();

return video;

}

public void updateVideo(Video video, VideoUpdateRequest videoUpdateRequest){

if(!video.getId().equals(videoUpdateRequest.getId())){

throw new VideoUpdateException("Cannot update");

}

Category category = null;

if(videoUpdateRequest.getCategoryId() != null) {

category = categoryRepository.findById(videoUpdateRequest.getCategoryId())

.orElseThrow(() -> new VideoUpdateException("Category with id = " + videoUpdateRequest.getCategoryId() + " not found"));

}

video.setTitle(videoUpdateRequest.getTitle());

video.setDescription(videoUpdateRequest.getDescription());

video.setPath(videoUpdateRequest.getPath());

video.setCategory(category);

}

public VideoUpdateRequest toVideoUpdateRequest(VideoResponse videoResponse){

VideoUpdateRequest videoUpdateRequest = VideoUpdateRequest.builder()

.id(videoResponse.getId())

.title(videoResponse.getTitle())

.description(videoResponse.getDescription())

.path(videoResponse.getPath())

.categoryId(videoResponse.getCategory().getId())

.build();

return videoUpdateRequest;

}

}

- Tạo 2 service **CategoryService** và **VideoService**:

+ **CategoryService**:

package com.bty.study.videocollection.service;

import com.bty.study.videocollection.dto.response.CategoryResponse;

import com.bty.study.videocollection.entity.Category;

import com.bty.study.videocollection.mapper.CategoryMapper;

import com.bty.study.videocollection.repository.CategoryRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

@Service

@RequiredArgsConstructor

public class CategoryService {

private final CategoryRepository categoryRepository;

private final CategoryMapper categoryMapper;

public List<CategoryResponse> getAllCategories(){

List<Category> categories = categoryRepository.findAll();

List<CategoryResponse> categoryResponses = new ArrayList<>();

for (Category category : categories) {

categoryResponses.add(categoryMapper.toCategoryResponse(category));

}

return categoryResponses;

}

}

+ **VideoService**:

package com.bty.study.videocollection.service;

import com.bty.study.videocollection.dto.request.VideoCreationRequest;

import com.bty.study.videocollection.dto.request.VideoUpdateRequest;

import com.bty.study.videocollection.dto.response.VideoResponse;

import com.bty.study.videocollection.entity.Category;

import com.bty.study.videocollection.entity.Video;

import com.bty.study.videocollection.exception.VideoCreationException;

import com.bty.study.videocollection.exception.VideoNotFoundException;

import com.bty.study.videocollection.exception.VideoUpdateException;

import com.bty.study.videocollection.mapper.VideoMapper;

import com.bty.study.videocollection.repository.CategoryRepository;

import com.bty.study.videocollection.repository.VideoRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

@Service

@RequiredArgsConstructor

public class VideoService {

private final VideoRepository videoRepository;

private final VideoMapper videoMapper;

private final CategoryRepository categoryRepository;

public List<VideoResponse> getAllVideos(){

List<Video> videos = videoRepository.findAll();

List<VideoResponse> videoResponses = new ArrayList<>();

for(Video video : videos){

videoResponses.add(videoMapper.toVideoResponse(video));

}

return videoResponses;

}

public VideoResponse getVideoById(Long id){

Video video = videoRepository.findById(id)

.orElseThrow(() -> new VideoNotFoundException("Video with id = " + id + " not found"));

return videoMapper.toVideoResponse(video);

}

public VideoResponse createVideo(VideoCreationRequest request) {

if(videoRepository.existsByTitle(request.getTitle())) {

throw new VideoCreationException("Video with title = " + request.getTitle() + " already exists");

}

if(videoRepository.existsByPath(request.getPath())) {

throw new VideoCreationException("Video with path = " + request.getPath() + " already exists");

}

Video video = videoMapper.toVideo(request);

return videoMapper.toVideoResponse(videoRepository.save(video));

}

public VideoResponse updateVideo(VideoUpdateRequest request) {

Video v = videoRepository.findById(request.getId())

.orElseThrow(() -> new VideoNotFoundException("Video with id = " + request.getId() + " not found"));

List<Video> videos = videoRepository.findAll();

for(Video video : videos){

if(video.getId().equals(request.getId())) continue;

if(video.getTitle().equals(request.getTitle()))

throw new VideoUpdateException("Video with title = " + request.getTitle() + " already exists");

if(video.getPath().equals(request.getPath()))

throw new VideoUpdateException("Video with path = " + request.getPath() + " already exists");

}

videoMapper.updateVideo(v, request);

return videoMapper.toVideoResponse(videoRepository.save(v));

}

public void deleteVideoById(Long id){

if(!videoRepository.existsById(id)) {

throw new VideoNotFoundException("Video with id = " + id + " not found");

}

videoRepository.deleteById(id);

}

public List<VideoResponse> findAllVideosByTitleContaining(String title){

List<Video> videos = videoRepository.findAllByTitleContaining(title);

List<VideoResponse> videoResponses = new ArrayList<>();

for(Video video : videos){

videoResponses.add(videoMapper.toVideoResponse(video));

}

return videoResponses;

}

}

- Tạo controller **VideoController**:

package com.bty.study.videocollection.controller;

import com.bty.study.videocollection.dto.request.VideoCreationRequest;

import com.bty.study.videocollection.dto.request.VideoUpdateRequest;

import com.bty.study.videocollection.dto.response.VideoResponse;

import com.bty.study.videocollection.mapper.VideoMapper;

import com.bty.study.videocollection.service.CategoryService;

import com.bty.study.videocollection.service.VideoService;

import jakarta.validation.Valid;

import lombok.RequiredArgsConstructor;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.\*;

import java.util.List;

@Controller

@RequiredArgsConstructor

public class VideoController {

private final VideoService videoService;

private final CategoryService categoryService;

private final VideoMapper videoMapper;

@GetMapping("/videos")

public String showList(@RequestParam(name = "keyword", required = false) String keyword,

Model model){

List<VideoResponse> videoResponseList = null;

String mode = null;

if(keyword != null){

videoResponseList = videoService.findAllVideosByTitleContaining(keyword);

mode = "search";

} else {

videoResponseList = videoService.getAllVideos();

mode = "all";

}

model.addAttribute("videos", videoResponseList);

model.addAttribute("mode", mode);

model.addAttribute("keyword", keyword);

return "videos";

}

@GetMapping("/videos/details/{id}")

public String showDetails(@PathVariable("id") Long id, Model model){

VideoResponse videoResponse = videoService.getVideoById(id);

videoResponse.setDescription(videoResponse.getDescription().replace("\n", "<br>"));

model.addAttribute("video", videoResponse);

return "video-details";

}

@GetMapping("/videos/new-from")

public String showNewForm(Model model){

model.addAttribute("videoCreationRequest", new VideoCreationRequest());

model.addAttribute("categoryList", categoryService.getAllCategories());

return "video-new-form";

}

@PostMapping("/videos/new")

public String create(@Valid @ModelAttribute("videoCreationRequest") VideoCreationRequest request,

BindingResult bindingResult,

Model model) {

if(bindingResult.hasErrors()){

model.addAttribute("categoryList", categoryService.getAllCategories());

return "video-new-form";

}

videoService.createVideo(request);

return "redirect:/videos";

}

@GetMapping("/videos/update-form/{id}")

public String showUpdateForm(@PathVariable("id") Long id, Model model){

VideoResponse videoResponse = videoService.getVideoById(id);

VideoUpdateRequest videoUpdateRequest = videoMapper.toVideoUpdateRequest(videoResponse);

model.addAttribute("videoUpdateRequest", videoUpdateRequest);

model.addAttribute("categoryList", categoryService.getAllCategories());

return "video-update-form";

}

@PostMapping("/videos/update")

public String update(@Valid @ModelAttribute("videoUpdateRequest") VideoUpdateRequest request,

BindingResult bindingResult,

Model model){

if(bindingResult.hasErrors()){

model.addAttribute("categoryList", categoryService.getAllCategories());

return "video-update-form";

}

videoService.updateVideo(request);

return "redirect:/videos";

}

@GetMapping("/videos/delete/{id}")

public String delete(@PathVariable("id") Long id, Model model){

videoService.deleteVideoById(id);

return "redirect:/videos";

}

}

- Tạo class **GlobalExceptionHandler**:

package com.bty.study.videocollection.exception;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.bind.annotation.ExceptionHandler;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler(value = VideoNotFoundException.class)

public String handleVideoNotFoundException(VideoNotFoundException e, Model model){

model.addAttribute("videoNotFoundException",e);

return "error";

}

@ExceptionHandler(value = VideoCreationException.class)

public String handleVideoCreationException(VideoCreationException e,

Model model){

model.addAttribute("videoCreationException",e);

return "error";

}

@ExceptionHandler(value = VideoUpdateException.class)

public String handleVideoUpdateException(VideoUpdateException e,Model model){

model.addAttribute("videoUpdateException",e);

return "error";

}

}

\* Cấu hình bảo mật với **Spring Security**:

- Tạo entity **Account**:

package com.bty.study.videocollection.entity;

import jakarta.persistence.\*;

import lombok.\*;

import lombok.experimental.FieldDefaults;

import org.hibernate.annotations.Check;

@Entity

@Table(name = "Account")

@NoArgsConstructor

@AllArgsConstructor

@Setter

@Getter

@ToString

@Builder

@FieldDefaults(level = AccessLevel.PRIVATE)

@Check(name = "check\_len\_Username", constraints = "LEN(Username) >= 3 AND LEN(Username) <= 50")

@Check(name = "check\_len\_Role", constraints = "LEN(Role) >= 1 AND LEN(Role) <= 20")

public class Account {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "Id")

Long id;

@Column(name = "Username", columnDefinition = "NVARCHAR(50)", unique = true, nullable = false)

String username;

@Column(name = "Password", columnDefinition = "NVARCHAR(3000)", nullable = false)

String password;

@Column(name = "Role", columnDefinition = "NVARCHAR(20)", nullable = false)

String role;

}

- Tạo repository **AccountRepository**:

package com.bty.study.videocollection.repository;

import com.bty.study.videocollection.entity.Account;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import java.util.Optional;

@Repository

public interface AccountRepository extends JpaRepository<Account, Long> {

Optional<Account> findByUsername(String username);

}

- Tạo service **CustomUserDetailsService**:

package com.bty.study.videocollection.service;

import com.bty.study.videocollection.entity.Account;

import com.bty.study.videocollection.repository.AccountRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.security.core.userdetails.User;

import org.springframework.security.core.userdetails.UserDetails;

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Service;

@Service

@RequiredArgsConstructor

public class CustomUserDetailsService implements UserDetailsService {

private final AccountRepository accountRepository;

@Override

public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

Account account = accountRepository.findByUsername(username)

.orElseThrow(() -> new UsernameNotFoundException(username));

return User.builder()

.username(account.getUsername())

.password(account.getPassword())

.roles(account.getRole())

.build();

}

}

- Tạo class **SecurityConfig**:

package com.bty.study.videocollection.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(authorize -> authorize

.requestMatchers(HttpMethod.GET, "/css/\*\*").permitAll()

.requestMatchers(HttpMethod.GET, "/videos").hasAnyRole("ADMIN", "MEMBER")

.requestMatchers(HttpMethod.GET, "/videos/details/{id}").hasAnyRole("ADMIN", "MEMBER")

.requestMatchers(HttpMethod.GET, "/videos/new-from").hasRole("ADMIN")

.requestMatchers(HttpMethod.POST, "/videos/new").hasRole("ADMIN")

.requestMatchers(HttpMethod.GET, "/videos/update-form/{id}").hasRole("ADMIN")

.requestMatchers(HttpMethod.POST, "/videos/update").hasRole("ADMIN")

.requestMatchers(HttpMethod.GET, "/videos/delete/{id}").hasRole("ADMIN")

.anyRequest().authenticated()

)

.formLogin(form -> form

.defaultSuccessUrl("/videos", true)

.permitAll()

)

.logout(logout -> logout

.logoutUrl("/logout")

.logoutSuccessUrl("/login")

.permitAll()

)

.csrf(csrf -> csrf.disable())

;

return http.build();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

- Thêm phụ thuộc **thymeleaf-extras-springsecurity**:

<dependency>

<groupId>org.thymeleaf.extras</groupId>

<artifactId>thymeleaf-extras-springsecurity6</artifactId>

</dependency>

- Tạo các trang **.html**:

+ **error.html**:

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Error</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<h1>Error</h1>

<div th:if="${videoNotFoundException}">

<p class="text-danger" th:text="${videoNotFoundException.getMessage()}"></p>

</div>

<div th:if="${videoCreationException}">

<p class="text-danger" th:text="${videoCreationException.getMessage()}"></p>

</div>

<div th:if="${videoUpdateException}">

<p class="text-danger" th:text="${videoUpdateException.getMessage()}"></p>

</div>

</body>

</html>

+ **video-details.html**:

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title th:text="${video.title}"></title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<h1 class="text-center" id="title" th:text="${video.title}"></h1>

<p class="text-center text-secondary" id="category" th:text="${video.category.name}"></p>

<video class="w-50 d-block mx-auto" id="video" th:src="${video.path}" controls>

</video>

<p class="w-50 mx-auto mt-2 border px-2 py-2 bg-secondary-subtle" id="description" th:utext="${video.description}">

</p>

</body>

</html>

+ **video-new-form.html**:

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Create a video</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<h1 class="text-center">Create a video</h1>

<form class="w-50 mx-auto border px-2 py-2 rounded" th:object="${videoCreationRequest}" th:action="@{/videos/new}" method="post">

<p>

<label class="form-label" for="title">Title</label>

<input class="form-control" type="text" id="title" th:field="\*{title}">

<span class="text-danger fst-italic" th:errors="\*{title}"></span>

</p>

<p>

<label class="form-label" for="description">Description</label>

<textarea class="form-control" id="description" th:field="\*{description}"></textarea>

<span class="text-danger fst-italic" th:errors="\*{description}"></span>

</p>

<p>

<label class="form-label" for="path">Path</label>

<input class="form-control" type="text" id="path" th:field="\*{path}">

<span class="text-danger fst-italic" th:errors="\*{path}"></span>

</p>

<p>

<label class="form-label" for="category">Category</label>

<select class="form-select" id="category" th:field="\*{categoryId}">

<option th:each="category : ${categoryList}"

th:value="${category.id}"

th:text="${category.name}">

</option>

</select>

</p>

<p class="text-center">

<button class="btn btn-success" type="submit">New</button>

</p>

</form>

</body>

</html>

+ **video-update-form.html**:

<!doctype html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Update a video</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<h1 class="text-center">Update a video</h1>

<form class="w-50 mx-auto border rounded px-2 py-2" th:object="${videoUpdateRequest}" th:action="@{/videos/update}" method="post">

<input type="hidden" th:field="\*{id}" />

<p>

<label class="form-label" for="title">Title</label>

<input class="form-control" type="text" id="title" th:field="\*{title}" />

<span class="text-danger fst-italic" th:errors="\*{title}"></span>

</p>

<p>

<label class="form-label" for="description">Description</label>

<textarea class="form-control" id="description" th:field="\*{description}"></textarea>

<span class="text-danger fst-italic" th:errors="\*{description}"></span>

</p>

<p>

<label class="form-label" for="path">Path</label>

<input class="form-control" type="text" id="path" th:field="\*{path}" />

<span class="text-danger fst-italic" th:errors="\*{path}"></span>

</p>

<p>

<label class="form-label" for="category">Category</label>

<select class="form-select" id="category" th:field="\*{categoryId}">

<option th:each="category : ${categoryList}"

th:value="${category.id}"

th:text="${category.name}">

</option>

</select>

</p>

<p class="text-center">

<button class="btn btn-primary" type="submit">Update</button>

</p>

</form>

</body>

</html>

+ **videos.html**:

<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org"

xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<head>

<meta charset="UTF-8">

<title>Videos</title>

<link rel="stylesheet" th:href="@{/css/videos.css}">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body>

<div>

Username: <span class="fw-bold" sec:authentication="principal.username"></span>

</div>

<p>

<a th:href="@{/logout}">Logout</a>

</p>

<h1 class="text-center" id="title"><a th:href="@{/videos}">The list of videos</a></h1>

<form class="w-75 mx-auto text-center row justify-content-md-center" id="video-search-form" th:action="@{/videos}" method="get">

<p class="col-7">

<input class="form-control" type="search" name="keyword" placeholder="Enter a title here..." th:value="${keyword != null}? ${keyword} : ''">

</p>

<p class="col-auto">

<button class="btn btn-primary" type="submit">Search</button>

</p>

</form>

<div class="text-center" id="mode">

<p class="d-inline-block border bg-body-tertiary">Mode: <span th:text="${mode}"></span></p>

</div>

<p class="text-center" sec:authorize="hasRole('ADMIN')">

<a th:href="@{/videos/new-from}">Create a video</a>

</p>

<table class="table table-striped table-hover table-bordered w-75 mx-auto" id="video-list">

<thead class="table-light">

<tr>

<th>Id</th>

<th>Title</th>

<th>Category</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr th:each="video : ${videos}">

<td th:text="${video.id}"></td>

<td th:text="${video.title}"></td>

<td th:text="${video.category.name}"></td>

<td>

<a th:href="@{'/videos/details/' + ${video.id}}">Watch</a>

<a sec:authorize="hasRole('ADMIN')" th:href="@{'/videos/update-form/' + ${video.id}}">Update</a>

<a class="text-danger" sec:authorize="hasRole('ADMIN')" th:href="@{'/videos/delete/' + ${video.id}}"

onclick="return confirm('Are you sure?');">Delete</a>

</td>

</tr>

</tbody>

</table>

</body>

</html>

- Tạo class **DataInit**:

package com.bty.study.videocollection.config;

import com.bty.study.videocollection.entity.Account;

import com.bty.study.videocollection.entity.Category;

import com.bty.study.videocollection.entity.Video;

import com.bty.study.videocollection.repository.AccountRepository;

import com.bty.study.videocollection.repository.CategoryRepository;

import com.bty.study.videocollection.repository.VideoRepository;

import lombok.RequiredArgsConstructor;

import org.springframework.boot.CommandLineRunner;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.stereotype.Component;

import java.util.ArrayList;

@Component

@RequiredArgsConstructor

public class DataInit implements CommandLineRunner {

private final CategoryRepository categoryRepository;

private final VideoRepository videoRepository;

private final AccountRepository accountRepository;

private final PasswordEncoder passwordEncoder;

@Override

public void run(String... args) throws Exception {

Category category1 = Category.builder()

.name("Music")

.description("...")

.videos(new ArrayList<>())

.build();

Category category2 = Category.builder()

.name("Game")

.description("...")

.videos(new ArrayList<>())

.build();

Category category3 = Category.builder()

.name("News")

.description("...")

.videos(new ArrayList<>())

.build();

Video video1 = Video.builder()

.title("Modern Talking - Cheri Cheri Lady")

.description(

"""

"Cheri, Cheri Lady" is a song by German pop duo Modern Talking from their second studio album, Let's Talk About Love (1985).

Released: 2 September 1985

Genre: Dance-pop

"""

)

.path("https://ia804602.us.archive.org/1/items/music-video\_202112/Modern%20Talking%20-%20Cheri%20Cheri%20Lady.mp4")

.build();

Video video2 = Video.builder()

.title("Modern Talking - Brother Louie")

.description(

"""

"Brother Louie" is a song by German pop duo Modern Talking, released as the lead single from their third studio album, Ready for Romance (1986).

"Brother Louie" was released on 27 January 1986 and it reached No. 1 on 3 March 1986 in Germany.

It is believed that the song was written by Dieter Bohlen about producer Luis Rodríguez, as he was a close associate of Bohlen and worked on many of the Modern Talking songs.

"""

)

.path("https://ia600701.us.archive.org/5/items/Modern\_Talking\_Brother\_Louie/Modern\_Talking\_Brother\_Louie.mp4")

.build();

Video video3 = Video.builder()

.title("Michael Jackson - Billie Jean")

.description(

"""

"Billie Jean" is a song by the American singer Michael Jackson, released by Epic Records on January 3, 1983, as the second single from his sixth studio album, Thriller (1982).

Released: January 3, 1983

"""

)

.path("https://ia600808.us.archive.org/35/items/MichaelJacksonBillieJeanOfficialMusicVideo/Michael%20Jackson%20-%20Billie%20Jean%20%28Official%20Music%20Video%29.mp4")

.build();

category1.addVideo(video1);

category1.addVideo(video2);

category1.addVideo(video3);

categoryRepository.save(category1);

categoryRepository.save(category2);

categoryRepository.save(category3);

Account admin = Account.builder()

.username("admin")

.password(passwordEncoder.encode("admin"))

.role("ADMIN")

.build();

Account member = Account.builder()

.username("member")

.password(passwordEncoder.encode("member"))

.role("MEMBER")

.build();

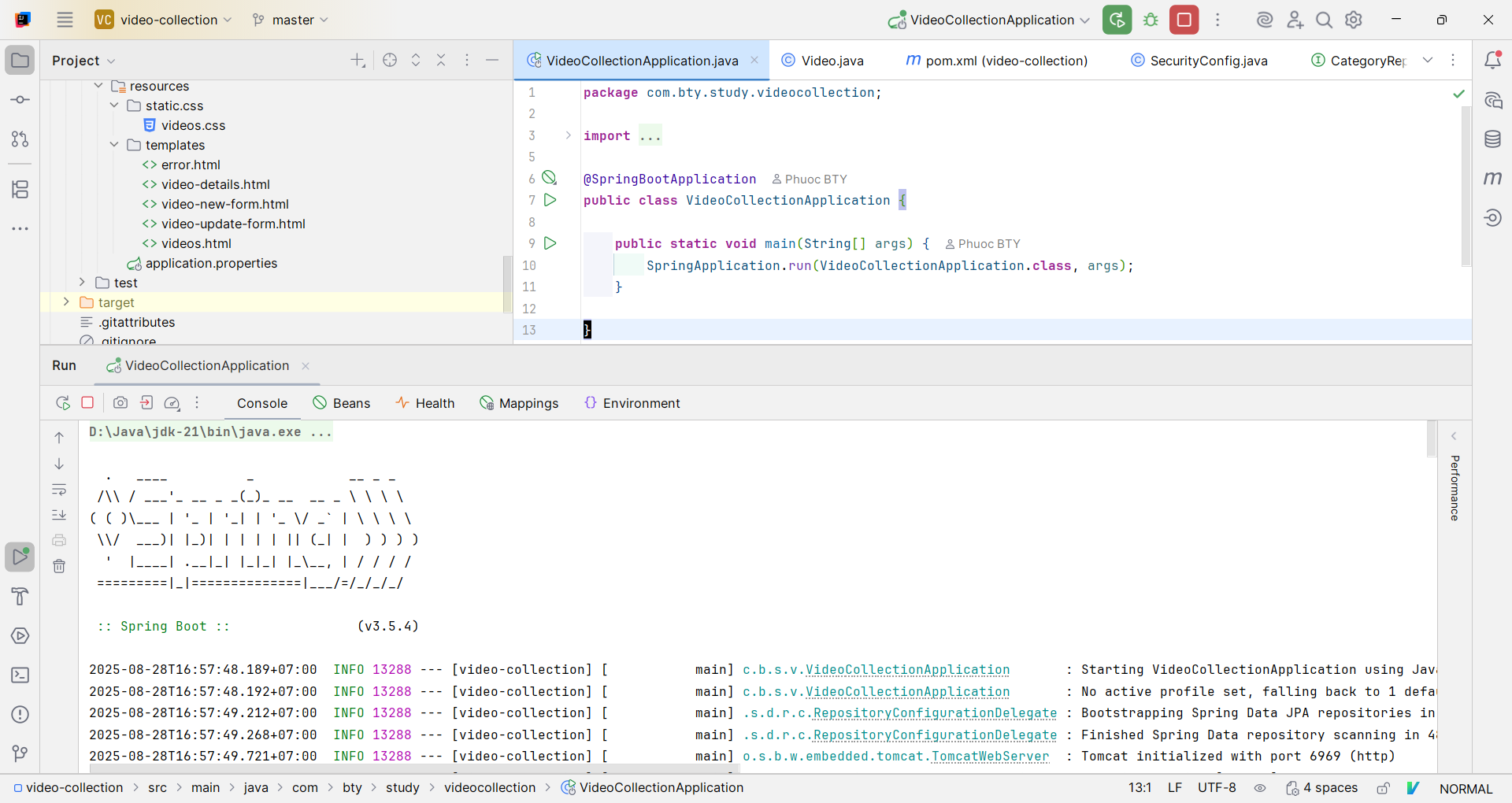
accountRepository.save(admin);

accountRepository.save(member);

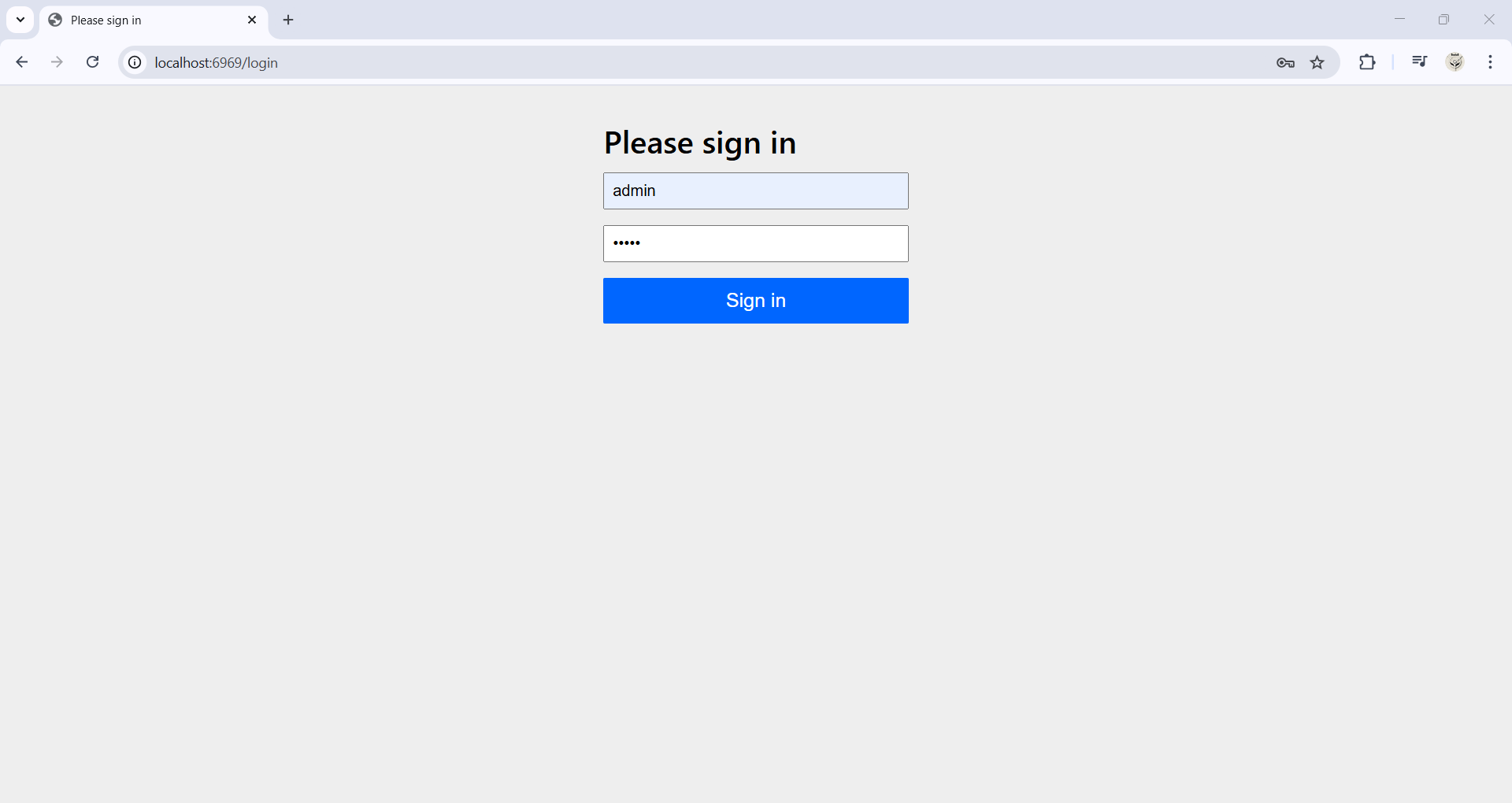
}

}

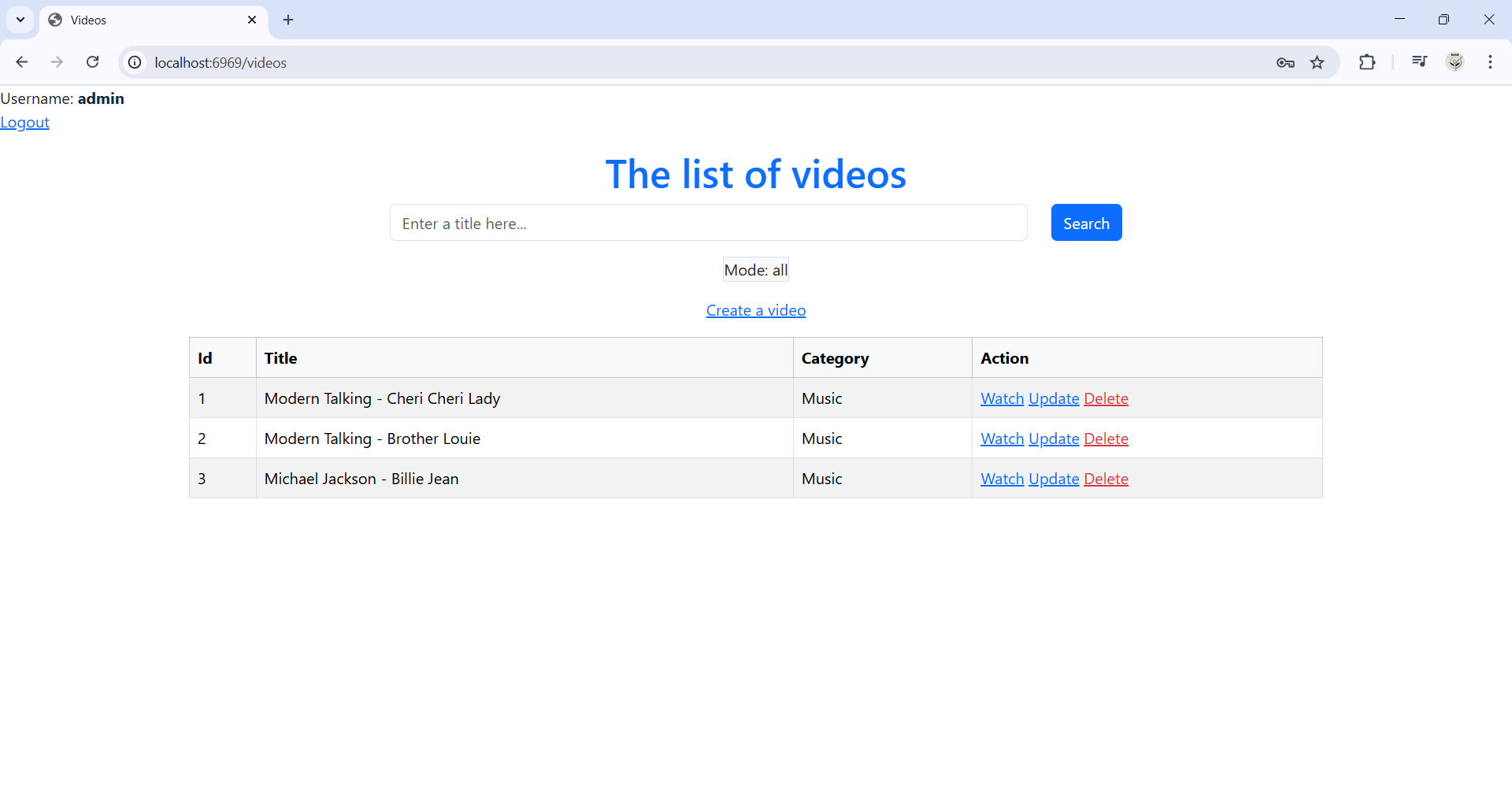
- Chạy ứng dụng:



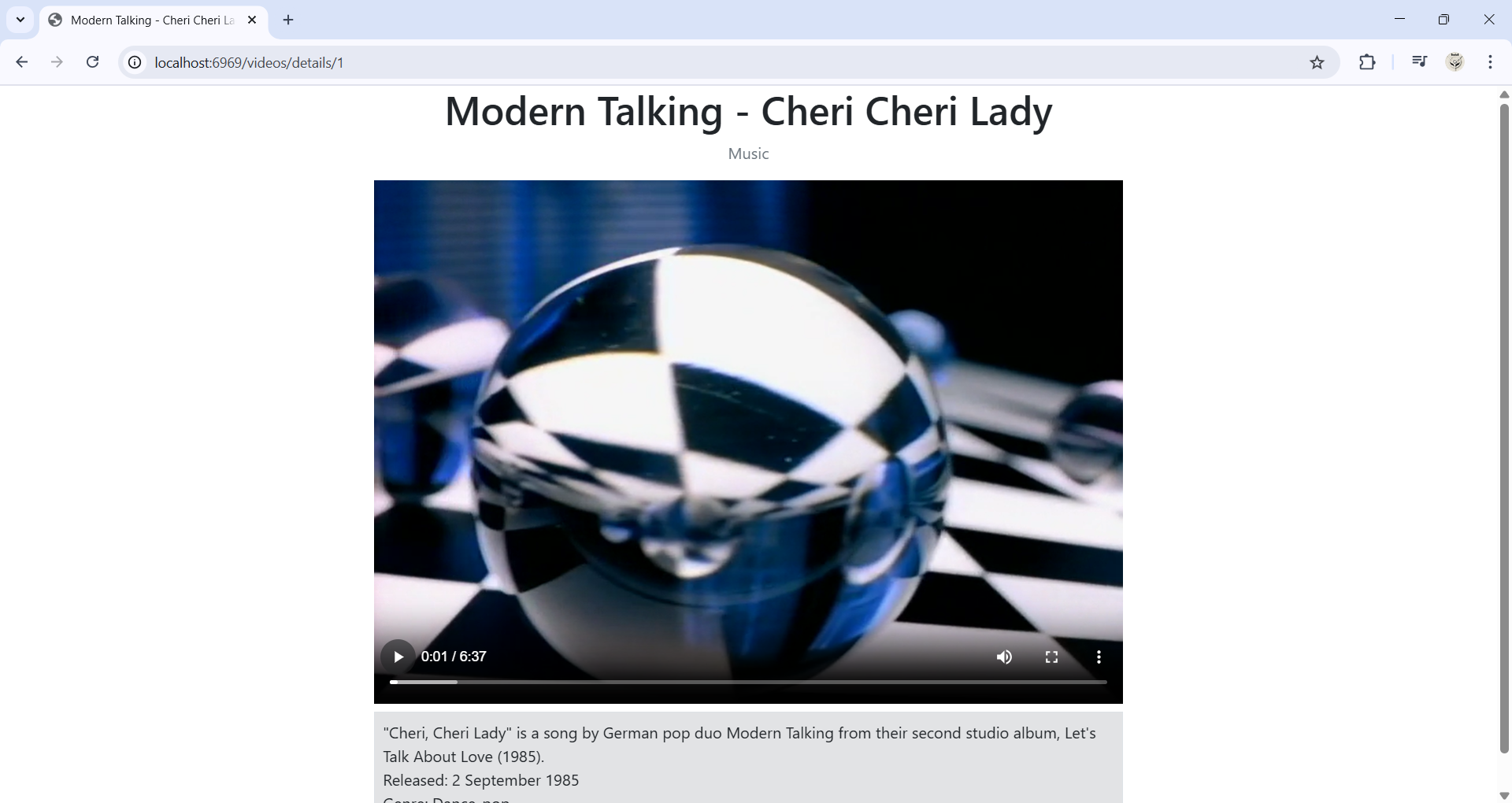
\* Đăng nhập với tài khoản **admin**:



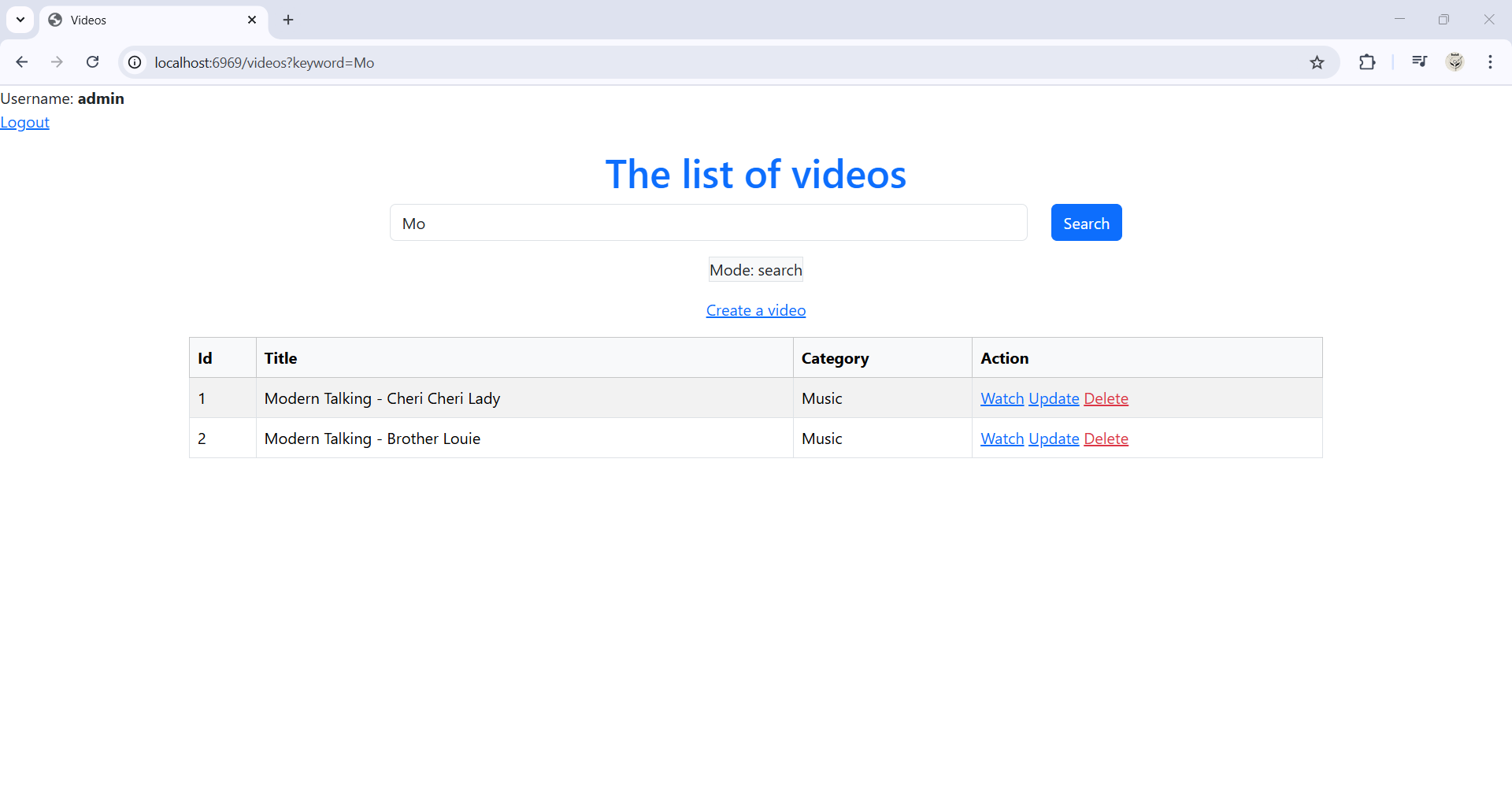
- xem danh sách tất cả video:



- Xem chi tiết một video:

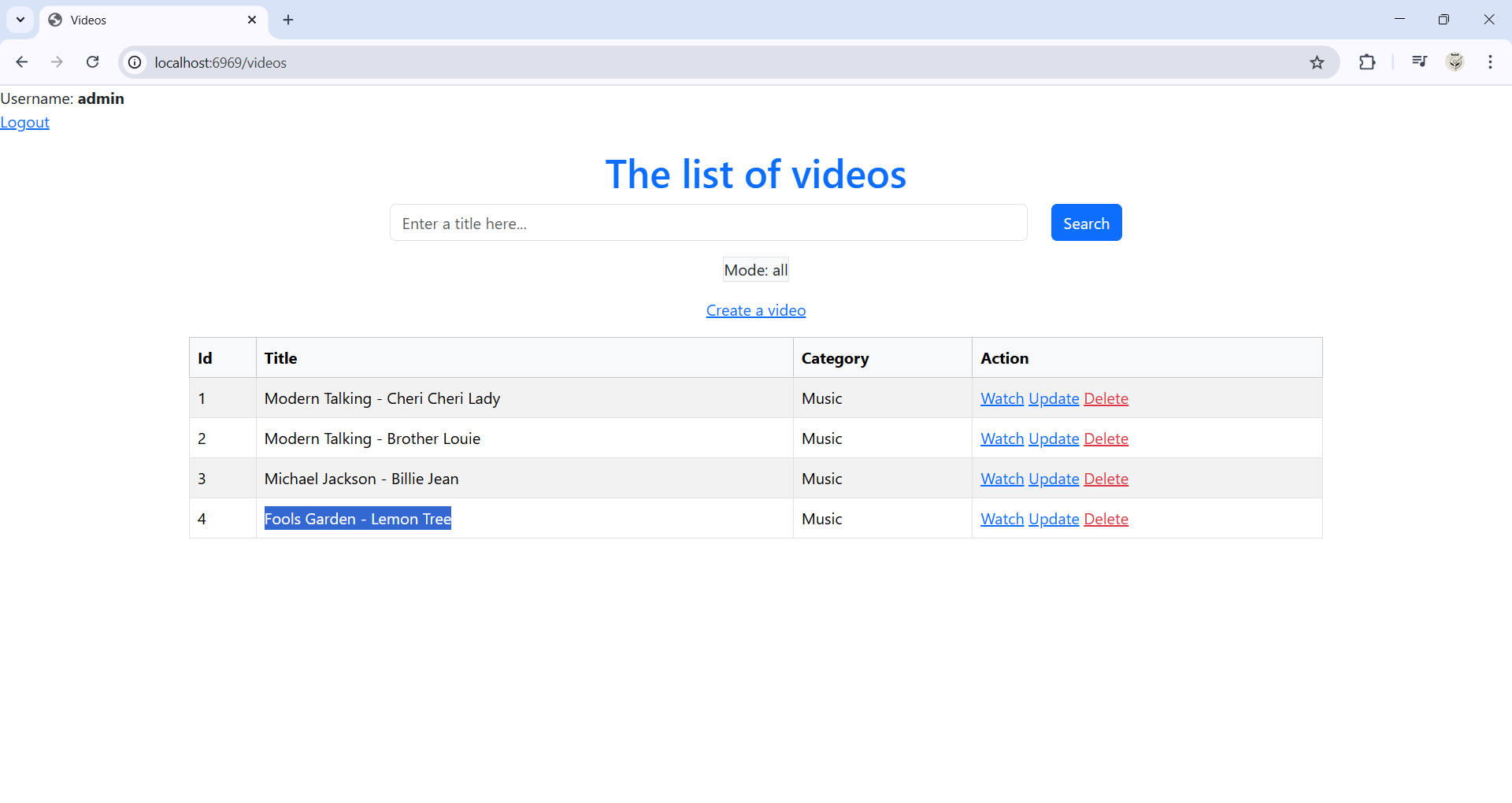


- Tìm kiếm video:

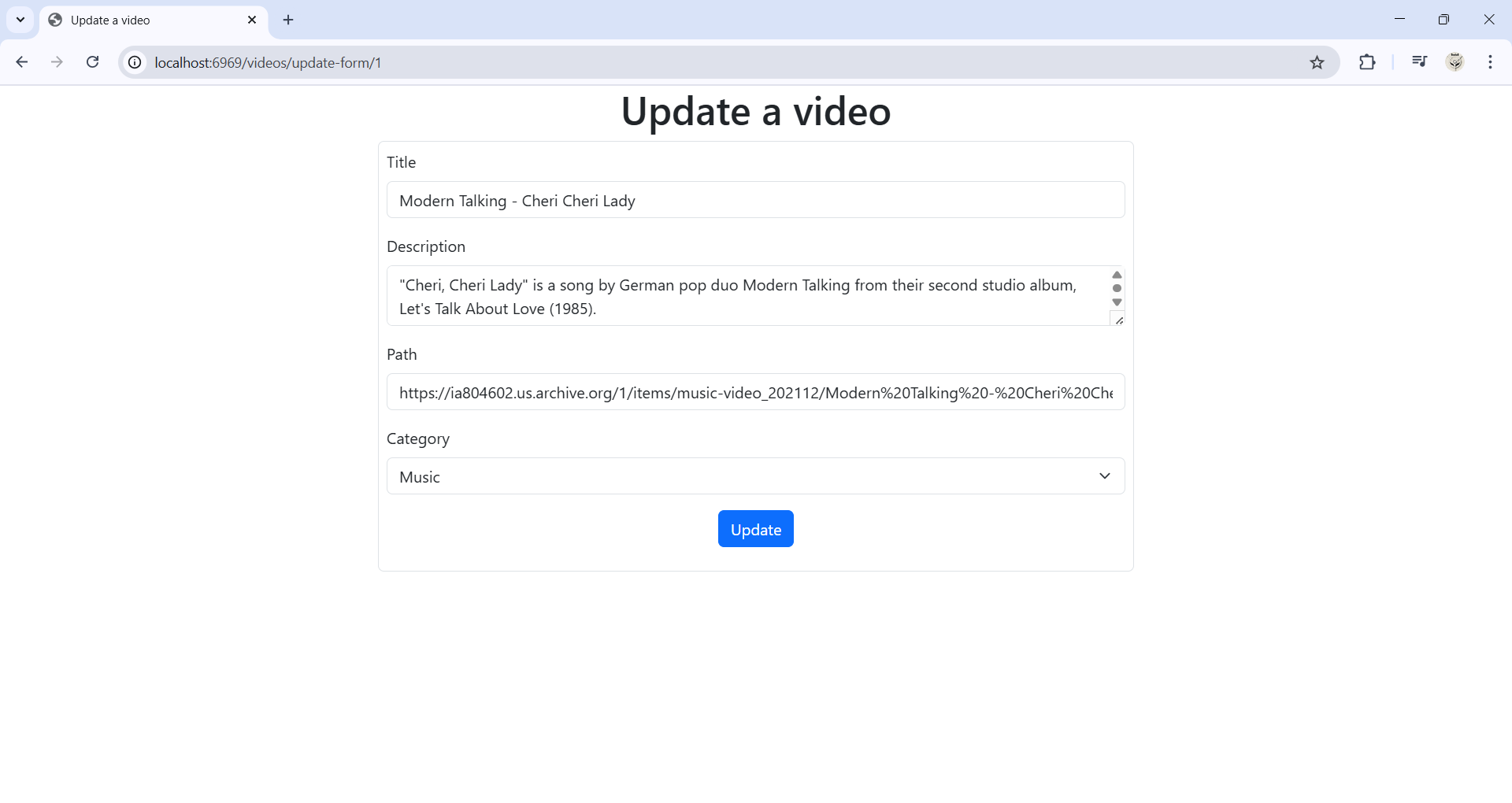


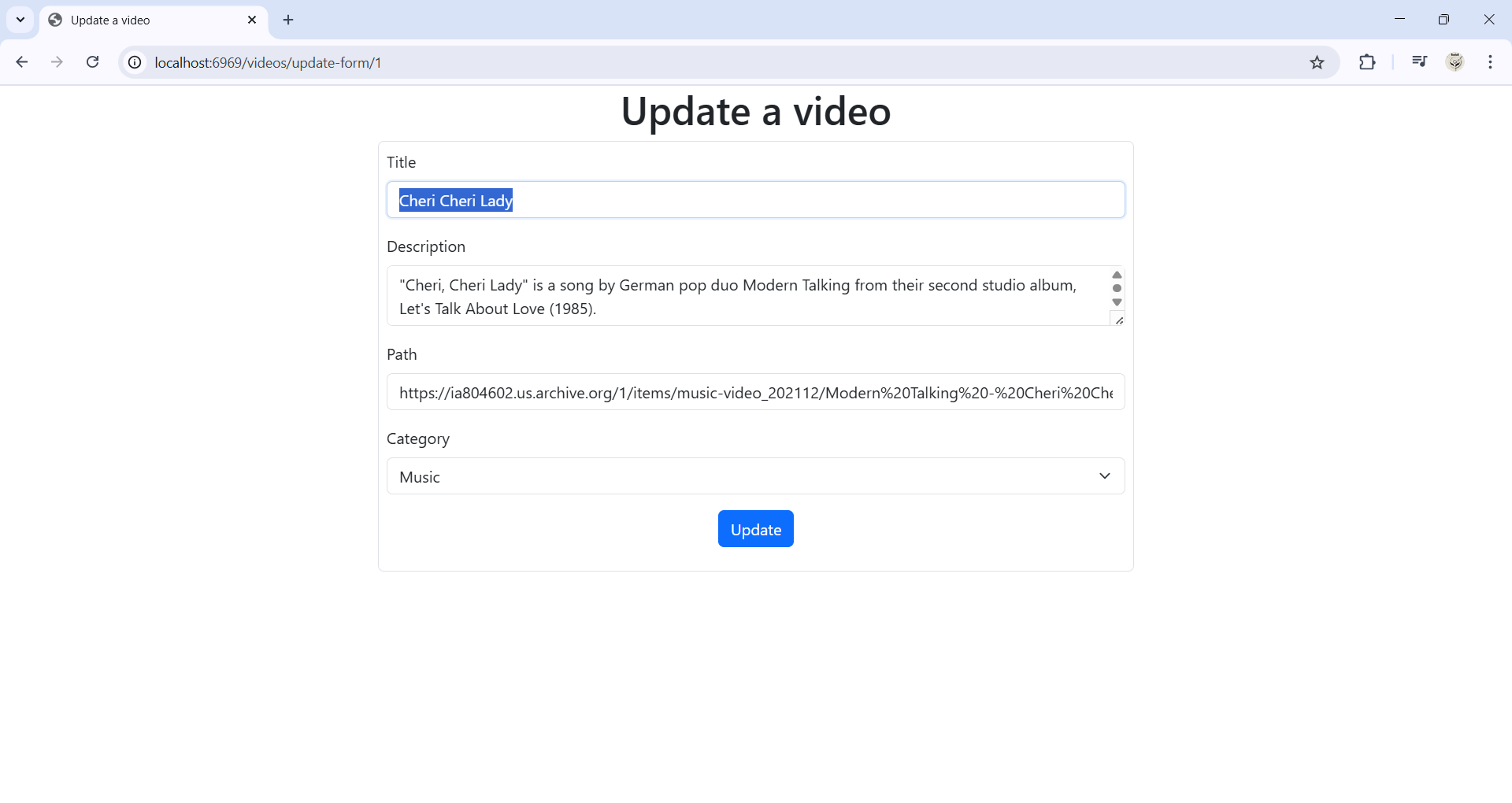
- Thêm video:

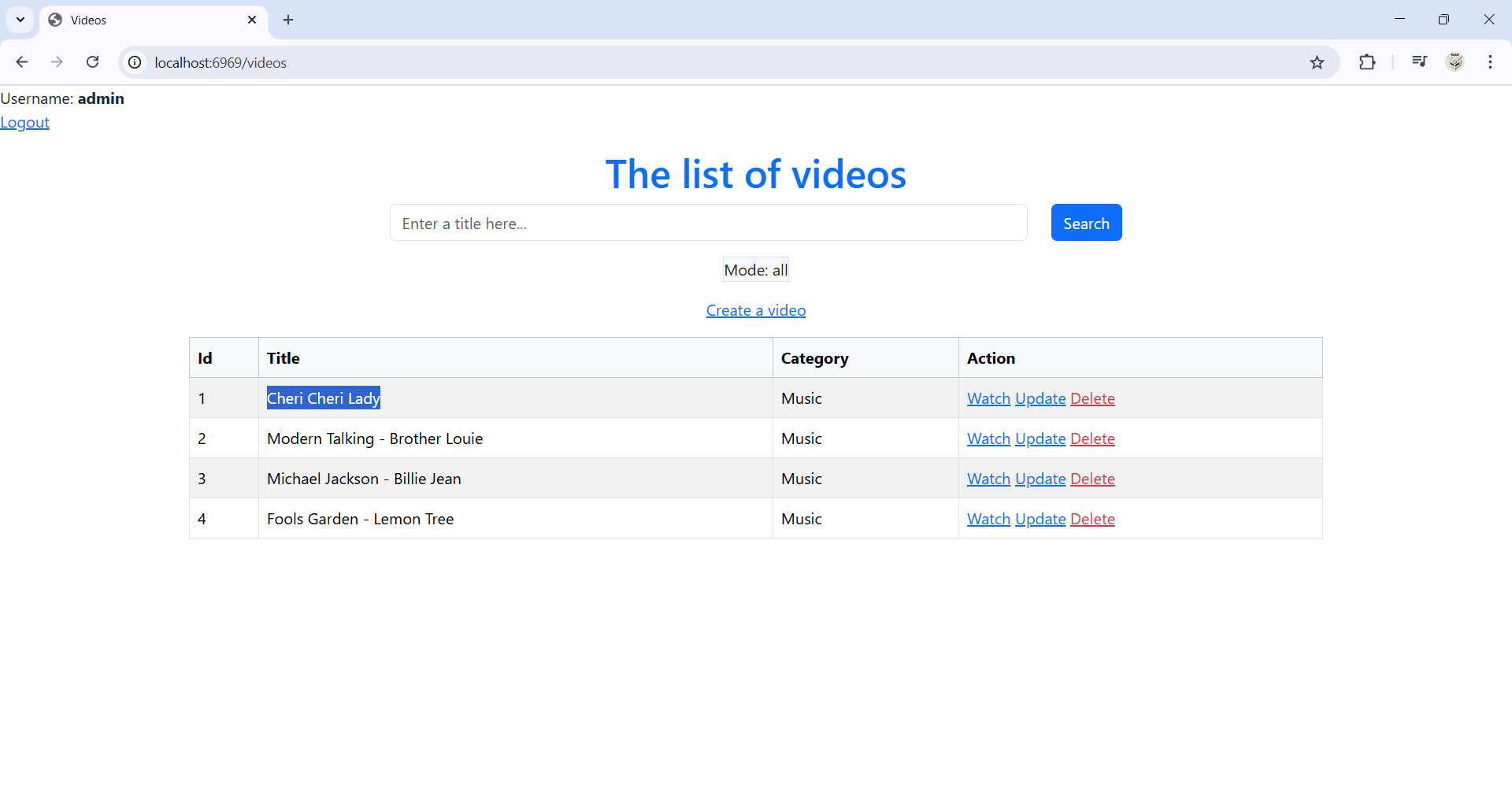




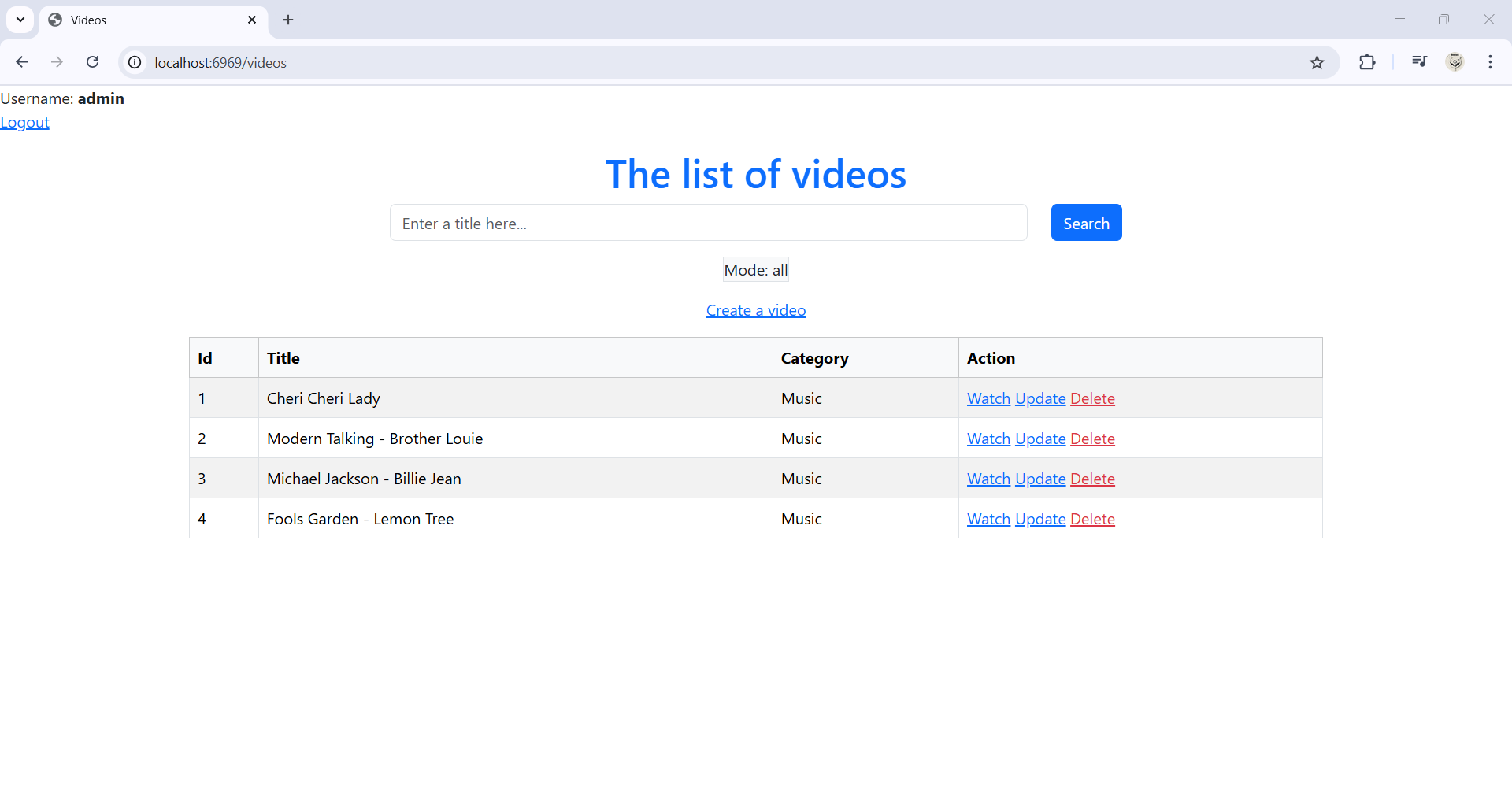
- Cập nhật video:



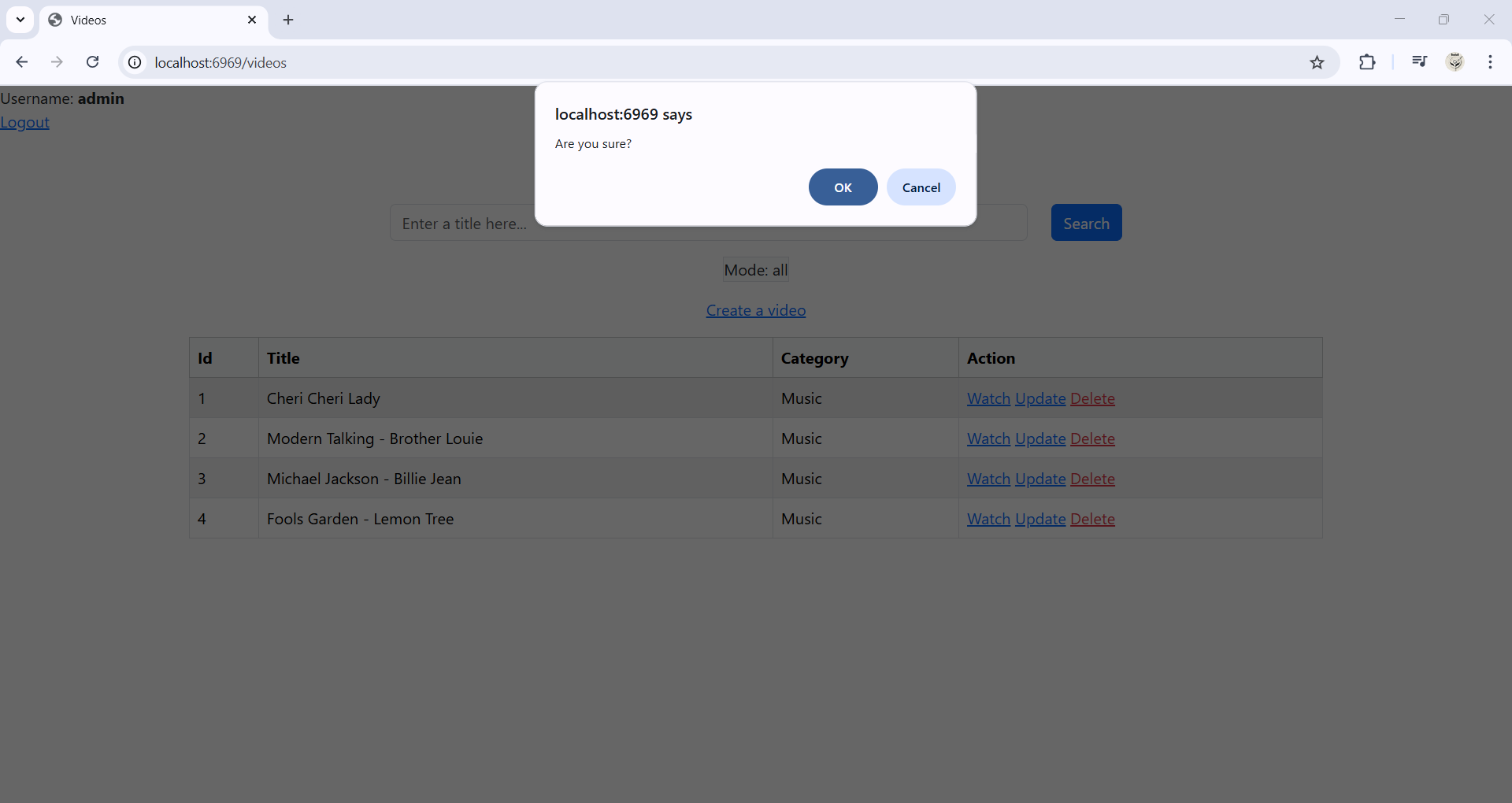


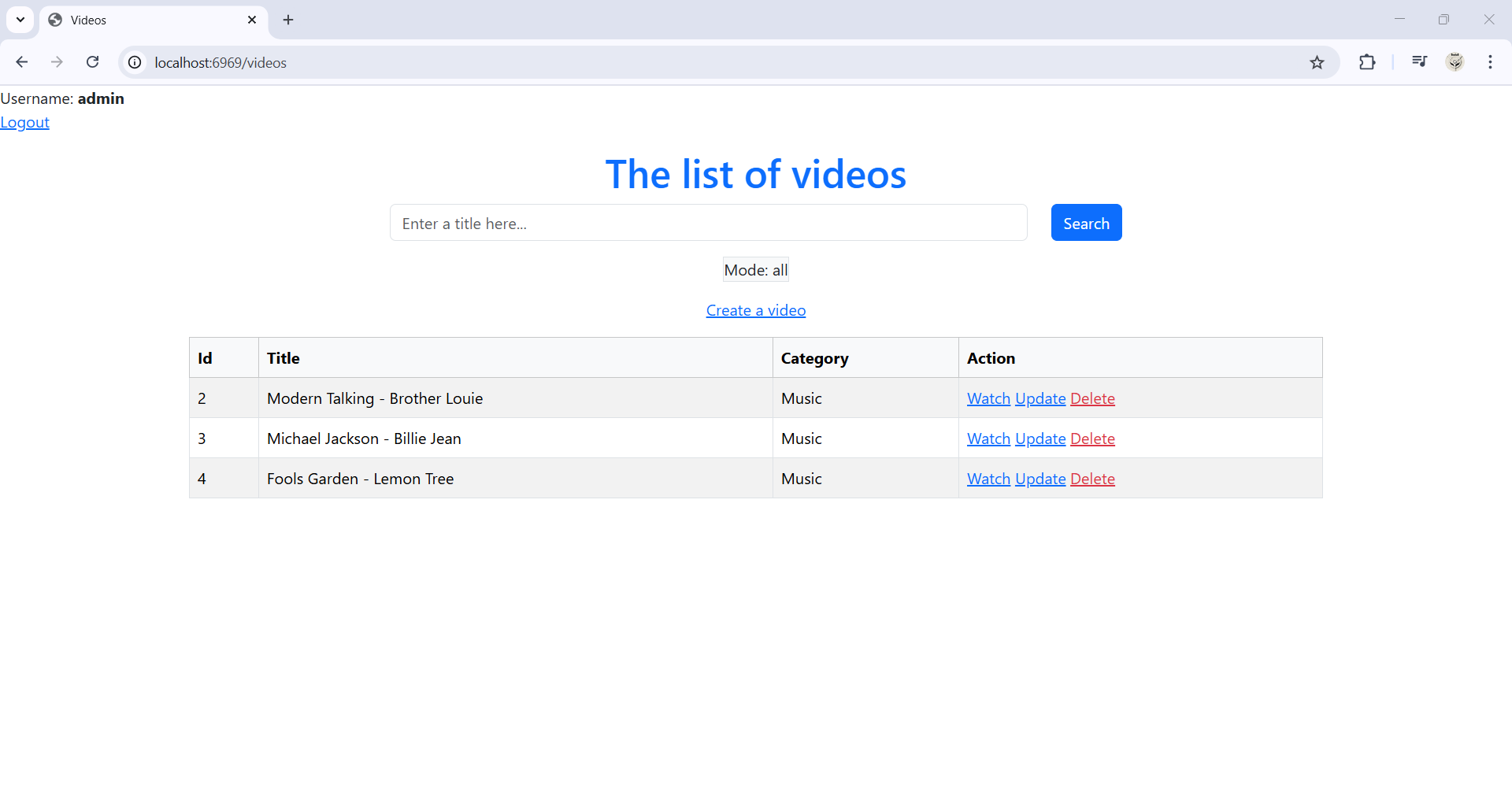


- Xóa video:

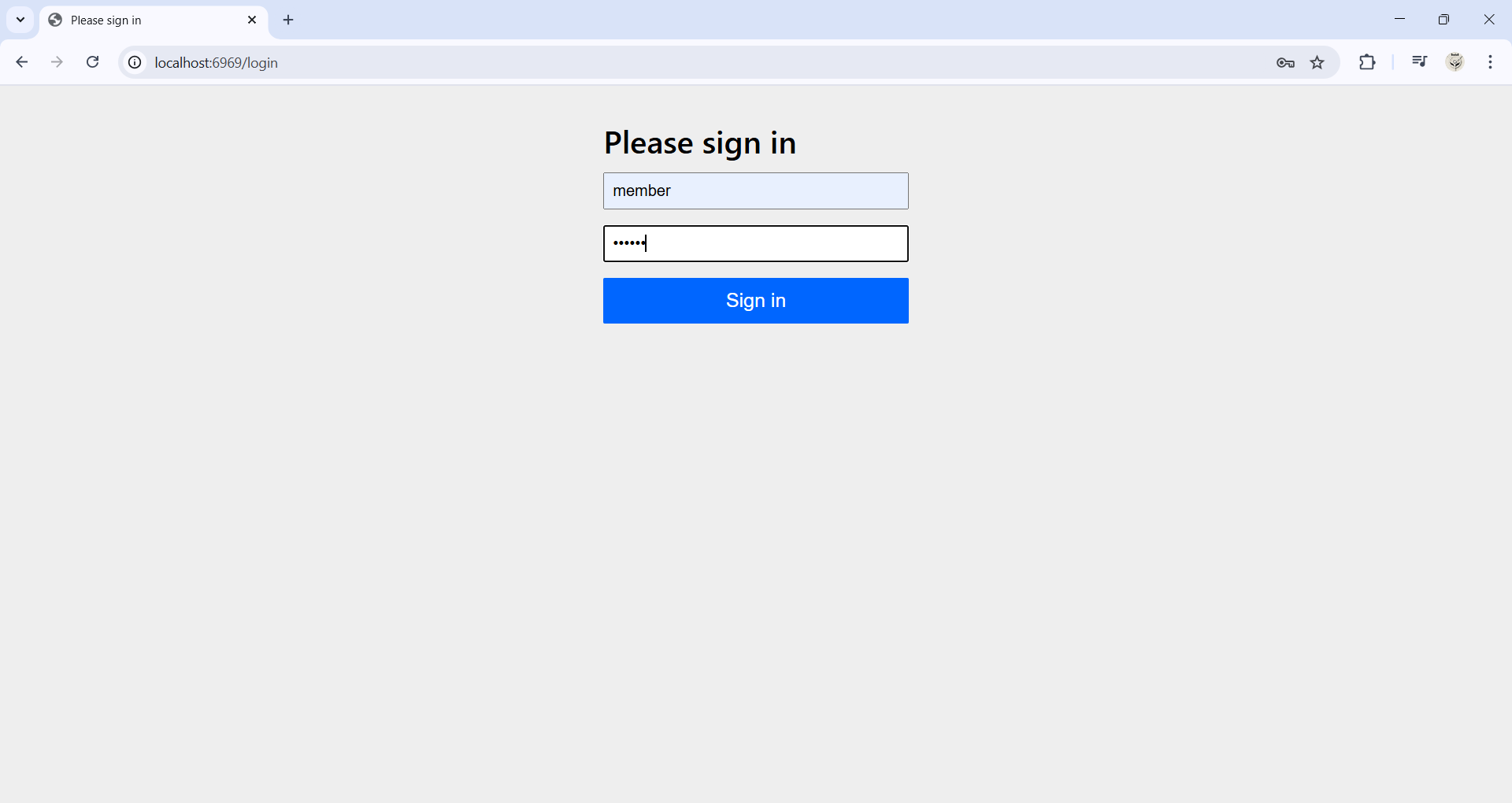


Xóa video với Id = 1:

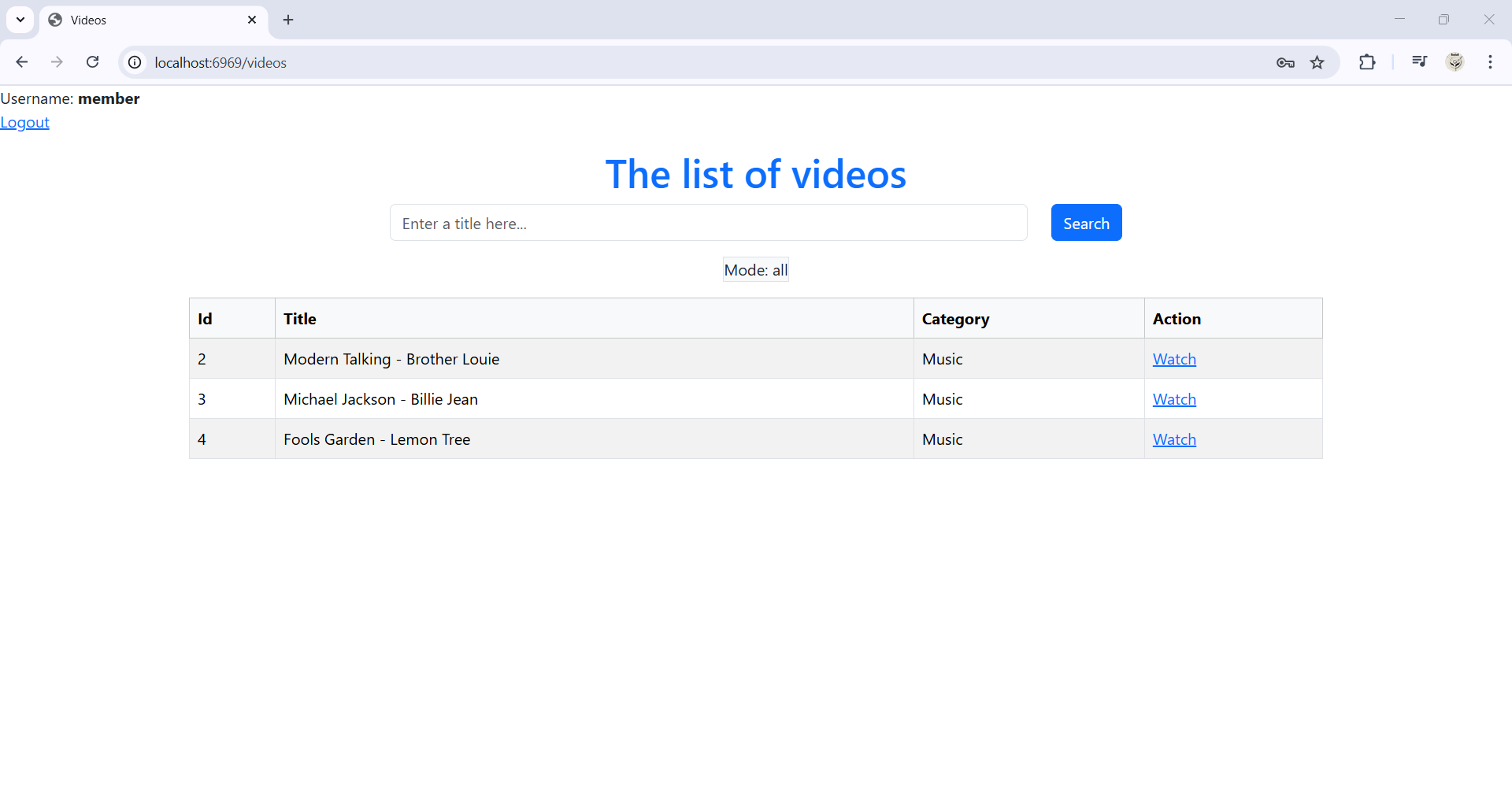




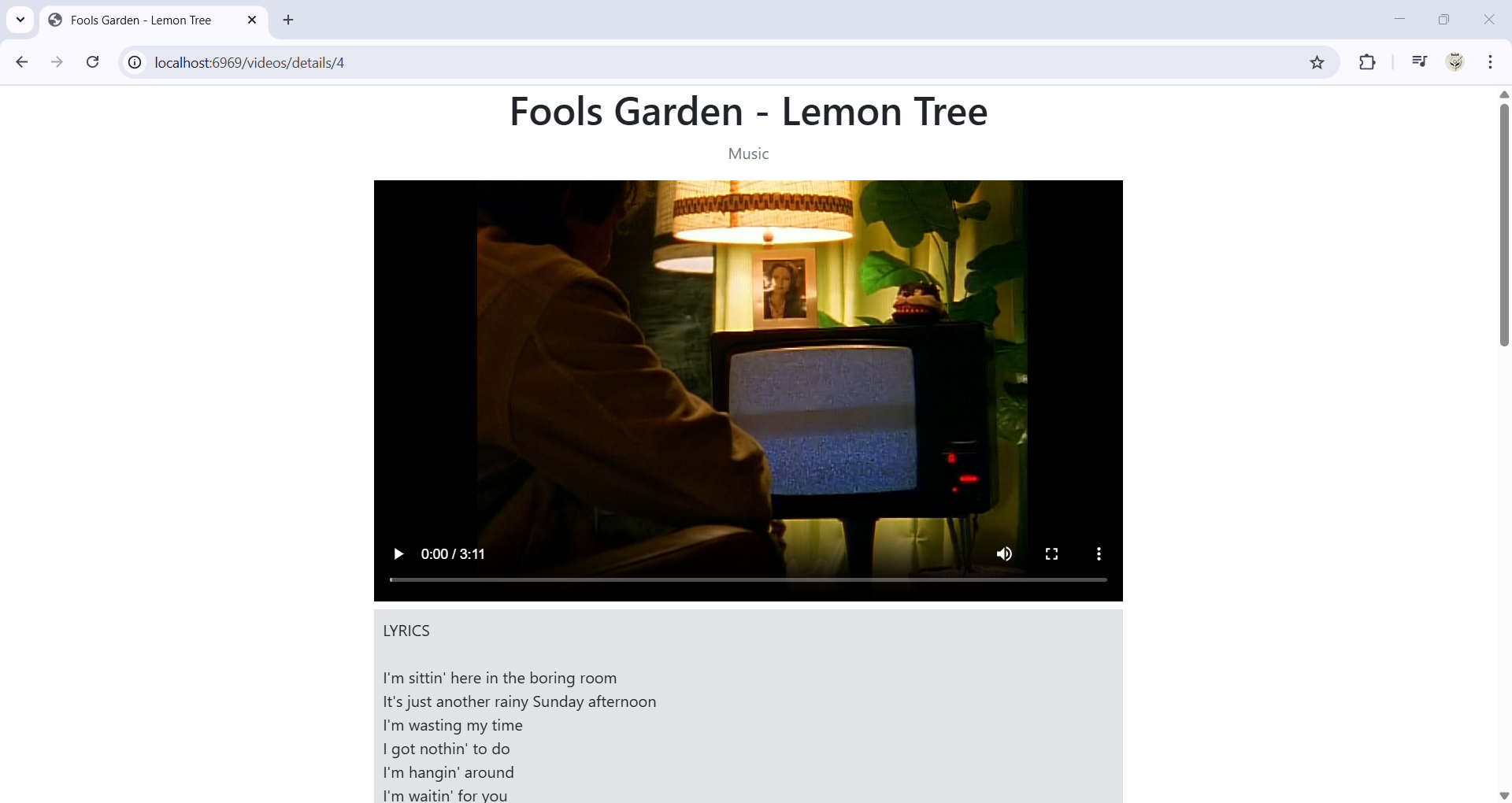
\* Đăng nhập với tài khoản **member**:



- Xem danh sách tất cả video:



- Xem chi tiết một video:



- Tìm kiếm video:

