

6차 수업 과제 - 프로젝트

2019156037 최혜민

프로젝트 개요

YouTube > IFrame Player API

검색

한국어

...

👤

참조 샘플 지원

필터

홈 > 제품 > YouTube > IFrame Player API

도움이 되었나요? ⤵ ⤶

iframe 삽입에 대한 YouTube Player API 참조 문서

IFrame Player API를 통해 웹사이트에 YouTube 동영상 플레이어를 퍼가고 JavaScript를 사용하여 플레이어를 제어할 수 있습니다. 웹페이지에 Flash 개체를 삽입하는 것과 관련된 Flash 및 JavaScript Player API와 달리 IFrame API는 콘텐츠를 페이지의 `<iframe>` 태그에 게시합니다. 이 방법은 YouTube에서 Flash를 지원하지 않는 모바일 기기에 Flash 대신 HTML5 플레이어를 게재할 수 있도록 하므로 기존에 제공된 API보다 좀 더 유연성을 제공합니다.

API의 JavaScript 함수를 사용하여 재생을 위해 동영상을 대기열에 넣거나, 이러한 동영상을 재생, 일시중지 또는 중지하거나, 플레이어 볼륨을 조정하거나, 재생 중인 동영상에 대한 정보를 가져올 수 있습니다. 또한 플레이어 상태 변경 또는 동영상 재생 품질 변경과 같은 특정 플레이어 이벤트에 대한 응답으로 실행되는 이벤트 리스너를 추가할 수 있습니다.

이 가이드에서는 IFrame API를 사용하는 방법을 설명합니다. API에서 보낼 수 있는 이벤트의 여러 유형을 식별하고 이벤트 리스너를 작성하여 이러한 이벤트에 응답하는 방법을 설명합니다. 또한 동영상 플레이어를 제어하기 위해 호출할 수 있는 여러 JavaScript 함수 및 플레이어를 좀 더 맞춤설정하는데 사용할 수 있는 플레이어 매개변수를 자세히 설명합니다.

요구사항

최종 사용자는 HTML5 `postMessage` 기능을 지원하는 브라우저를 사용해야 합니다. 대부분의 최신 브라우저가 `postMessage`를 지원하지만 Internet Explorer 7은 지원하지 않습니다.

내자 플레이어에는 200x200픽셀 이상의 표시 영역이 있어야 합니다. 플레이어에 커트로이 표시되는 경우에는 표시 영역이 최소 크

이 페이지의 내용

요구사항

시작하기

동영상 플레이어 로드

연산

함수

대기열 함수

재생 컨트롤 및 플레이어 설정

재생 상태

재생 품질

동영상 정보 가져오기

재생목록 정보 가져오기

이벤트 리스너 추가 및 삭제

DOM 노드 액세스 및 수정

이벤트

모바일 고려사항

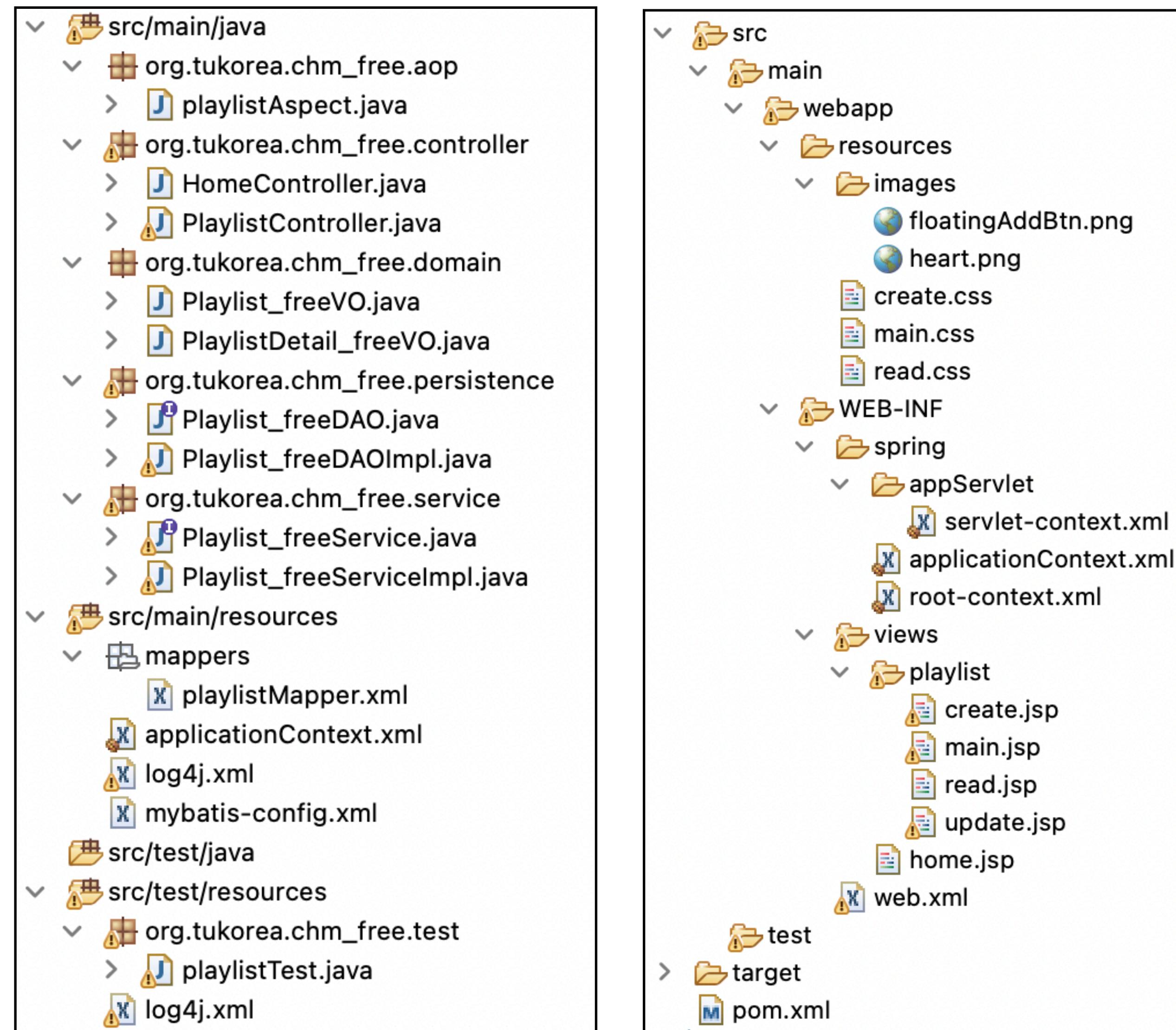
자동 재생 및 스크립트 재생

예

YouTube 플레이어 개체 만들기

업데이트 기록

프로젝트 구성



프로젝트 구성

- `create table playlist_free (`
 `playlistNumber int(10) not null AUTO_INCREMENT primary key,`
 `playlistName char(20) not null,`
 `playlistDescribe char(20),`
 `playlistLikes int(10),`
 `playlistPassword char(100),`
 `playlistPhoto char(100)`
 `);`
- `create table playlistDetail_free (`
 `playlistNumber int(10),`
 `FOREIGN KEY (`playlistNumber`) REFERENCES `playlist_free`(`playlistNumber`),`
 `playlistDetailSource char(100),`
 `primary key(playlistNumber, playlistDetailSource)`
 `);`

DI 설정

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
    <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
    <property name="url" value="jdbc:mysql://127.0.0.1:3306/springdb?allowPublicKeyRetrieval=true&useUnicode=true&characterEncoding=utf8&useSSL=false" />
    <property name="username" value="root" />
    <property name="password" value="password" />
    <property name="maxActive" value="5" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
    <constructor-arg ref="dataSource" />
</bean>

<bean class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">
    <constructor-arg ref="dataSource" />
</bean>

<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource" />
</bean>

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="configLocation" value="classpath:/mybatis-config.xml" />
    <property name="mapperLocations" value="classpath:mappers/*Mapper.xml" />
</bean>

<bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate" destroy-method="clearCache">
    <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactory" />
</bean>

<tx:annotation-driven transaction-manager="transactionManager" />

<context:component-scan base-package="org.tukorea.chm_free.aop" />
<context:component-scan base-package="org.tukorea.chm_free.service" />
<context:component-scan base-package="org.tukorea.chm_free.persistence" />

<aop:aspectj-autoproxy></aop:aspectj-autoproxy>
```

AOP 설정

```
@Before("execution(* readList())")
public void beforeRead(JoinPoint jp) {
    System.out.println("[before readList]");
    System.out.println("플레이리스트 전체를 읽어오고 있습니다. 조금만 기다려주세요... ");
    Signature sig = jp.getSignature();
    System.out.println(sig.getName() + " 함수 호출 중");
}

@After("execution(* readList())")
public void afterRead(JoinPoint jp) {
    System.out.println("[after readList]");
    System.out.println("플레이리스트를 전부 읽어왔습니다.");
    System.out.println("방문해주셔서 감사합니다 :)");
}

@AfterReturning(value="execution(* selectById(Integer))", returning="playlist")
public void afterReturningSelectById(JoinPoint jp, Playlist_freeVO playlist) {
    System.out.println("[AfterReturning selectById]");
    System.out.println(playlist.getPlaylistName() + "를 불러왔습니다.");
}

@After("execution(* heart(org.tukorea.chm_free.domain.Playlist_freeVO))")
public void afterHeart() {
    System.out.println("[@After heart]");
    System.out.println("하트 추가 완료! 하트는 계속 추가할 수 있습니다");
    System.out.println("플레이리스트가 좋았던 만큼 하트를 눌러 주세요!");
}

@AfterThrowing(value="execution(* add(org.tukorea.chm_free.domain.PlaylistDetail_freeVO))", throwing="ex")
public void afterThrowingAddDetail(Throwable ex) {
    System.out.println("[@AfterThrowing addDetail]");
    System.out.println(ex.toString());
    System.out.println("같은 링크를 2번 이상 입력하셨나요?");
}
```

트랜잭션 설정

```
@Override  
@Transactional(propagation=Propagation.REQUIRED, isolation=Isolation.READ_COMMITTED, timeout=10) // annotation Transaction  
public void addALL(Playlist_freeVO playlist, List<String> Dvo) throws Exception {  
    // 1  
    Integer playlistNumber = playlist_freeDAO.add(playlist);  
  
    // 2  
    System.out.println(playlistNumber);  
    for (int t = 0; t < Dvo.size(); t++){  
        PlaylistDetail_freeVO dao = new PlaylistDetail_freeVO();  
        dao.setPlaylistDetailSource(Dvo.get(t));  
        dao.setPlaylistNumber(playlistNumber);  
        playlist_freeDAO.add(dao);  
    }  
}  
  
@Override  
@Transactional(propagation=Propagation.REQUIRED, isolation=Isolation.READ_COMMITTED, timeout=10) // annotation Transaction  
public void deleteAll(Integer playlistNumber) throws Exception {  
    // 1  
    playlist_freeDAO.deleteDetail(playlistNumber);  
  
    // 2  
    playlist_freeDAO.delete(playlistNumber);  
}
```

실행 결과 화면 - Main

Playlist List

5 선선한 새벽, 혼자 걷고 싶을 때 수정 삭제

#Pink Sweat, #Flower, #Face to Face

43 저녁에 듣기 좋은 잔잔한 재즈 수정 삭제

#Wendy Marcini #Oakwood Station

89 빈티지한 크리스마스 재즈 캐롤 수정 삭제

#White Christmas, #Let it Snow!, #A Masrshmallow World

112 조성진의 클래식 플레이리스트 수정 삭제

#Clair de lune, #The Little Shepherd, #Lento placido

+

실행 결과 화면 - Add

Playlist Info

playlistName	공백 없이 입력하세요
playlistDescribe	공백 없이 입력하세요
playlistPassword	공백 없이 입력하세요(정수형 숫자)
playlistPhoto	공백 없이 입력하세요
playlistDetailSource	
	공백 없이 입력하세요

[행추가](#) [행삭제](#)

[보내기](#)

실행 결과 화면 - Read

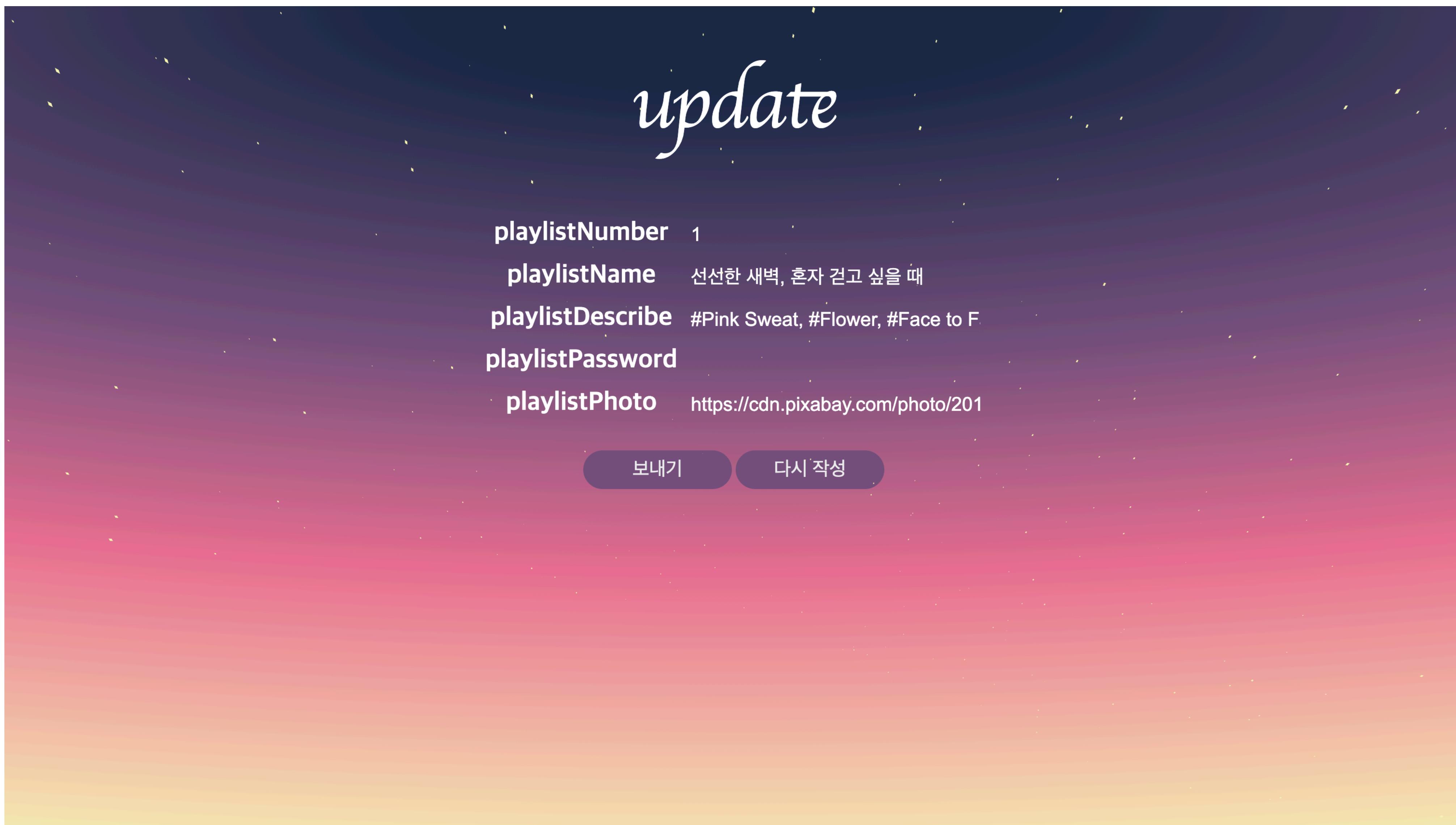


https://www.youtube.com/watch?v=8CEJoCr_9UI

<https://www.youtube.com/watch?v=EcaUV-mrKEA>

https://www.youtube.com/watch?v=r5Z741PC9_c

실행 결과 화면 - Update



실행 결과 화면 - Delete

The screenshot shows a web application interface with a background gradient from dark blue at the top to orange at the bottom. On the left, there is a list of songs with their names, heart icons, and counts (e.g., 5, 43, 89, 112). Each song entry includes hashtags and edit/delete buttons. On the right, a modal dialog box is displayed with the following text:
localhost:8080 내용:
삭제를 위해서 비밀번호를 입력해주세요.
password
취소 확인

5 선선한 새벽, 혼자 걷고 싶을 때 수정 삭제
#Pink Sweat, #Flower, #Face to Face

43 저녁에 듣기 좋은 잔잔한 재즈 수정 삭제
#Wendy Marcin #Oakwood Station

89 빈티지한 크리스마스 재즈 캐롤 수정 삭제
#White Christmas, #Let it Snow!, #A Marshmallow World

112 조성진의 클래식 플레이리스트 수정 삭제
#Clair de lune, #The Little Shepherd, #Lento placido

A large blue circular button with a white plus sign is located in the bottom right corner.