

Update a Song

Update a Song

- As well as being able to add and delete songs, we should be able to edit/change them also.
- We can update a song by modifying it's object element in the *songs* array.







Update a Song

Playlist 4

Dashboard

About

Beethoven Sonatas

Song	Artist		
<input type="text" value="Piano Sonata No. 8"/>	<input type="text" value="Beethoven"/>		
<input type="text" value="Piano Sonata No. 10"/>	<input type="text" value="Beethoven"/>		
<input type="text" value="Piano Sonata No. 12"/>	<input type="text" value="Beethoven"/>		

Title

Artist

Add Song

Update a Song – *listsongs.hbs*

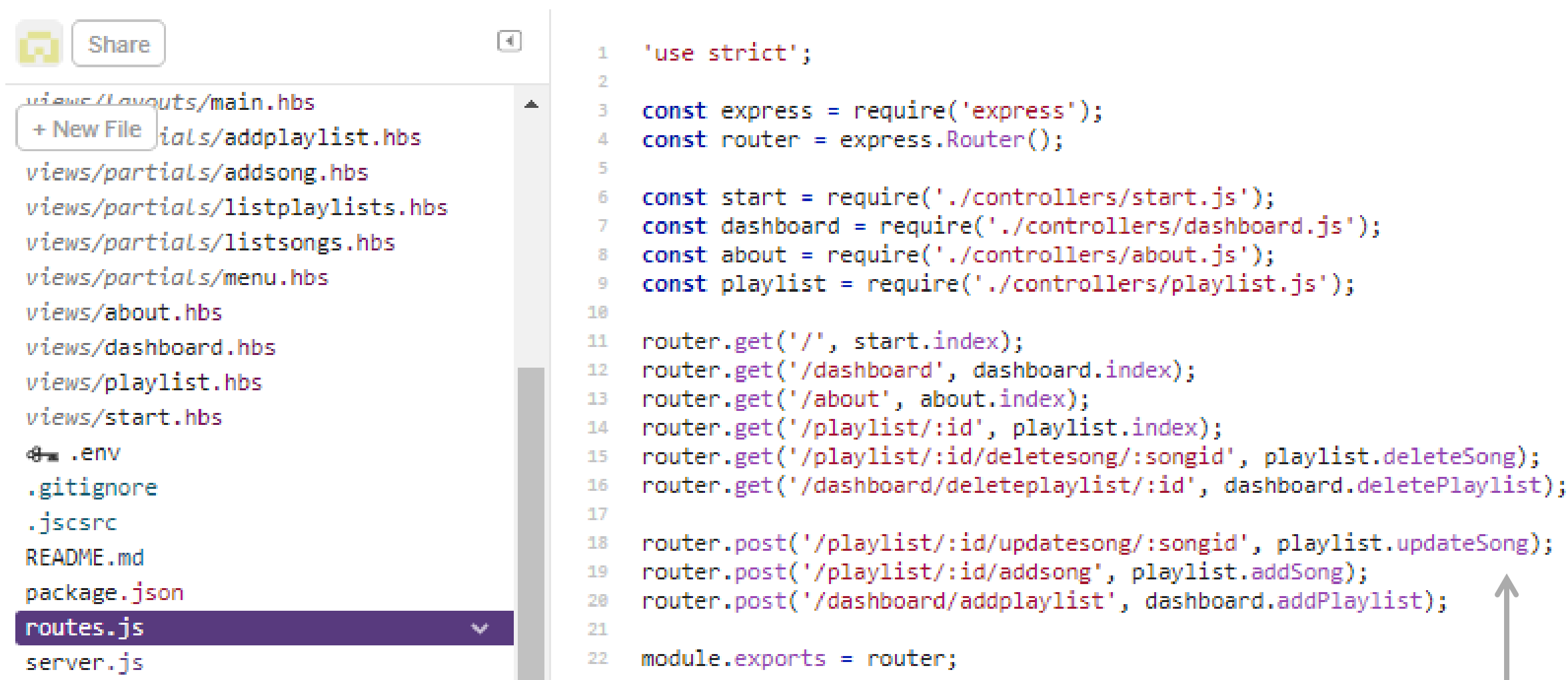
action



```
<form class = "ui form" action="/playlist/{{../playlist.id}}/updatesong/{{id}}" method="POST">
<tr class="fields">
  <td class="field">
    <span class="ui input">
      <input placeholder="Title" type="text" size = "30" name="title" autofocus required value = "{{title}}">
    </span>
  </td>
  <td class="field">
    <span class="ui input">
      <input placeholder="Artist" type="text" size = "30" name="artist" required value = "{{artist}}">
    </span>
  </td>
  <td>
    <button class="ui submit icon button"><i class="icon pencil alternate"></i></button>
  </td>
  <td>
    <a href="/playlist/{{../playlist.id}}/deletesong/{{id}}" class="ui icon button delsong">
      <i class="icon trash"></i>
    </a>
  </td>
</tr>
</form>
... ..
```

Each field element is displayed in an input text form control

Update a Song – *routes.js*

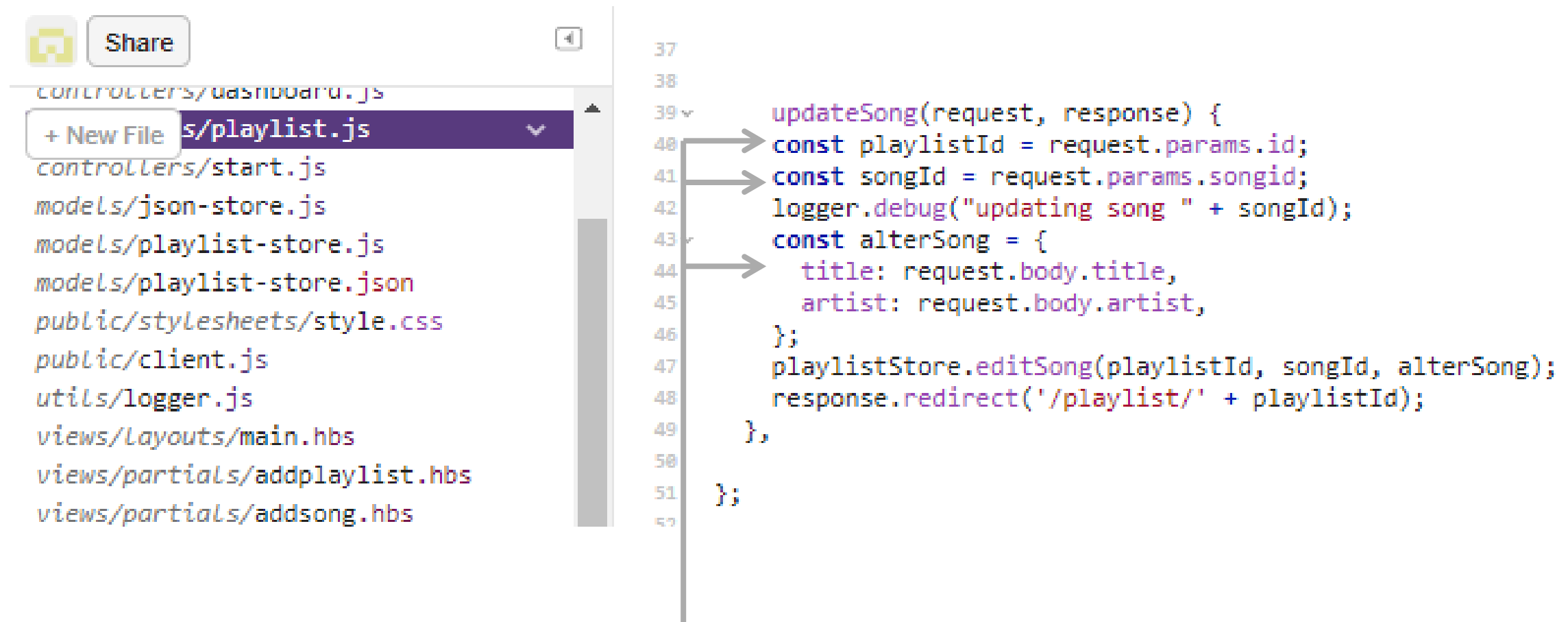


```
1  'use strict';
2
3  const express = require('express');
4  const router = express.Router();
5
6  const start = require('./controllers/start.js');
7  const dashboard = require('./controllers/dashboard.js');
8  const about = require('./controllers/about.js');
9  const playlist = require('./controllers/playlist.js');
10
11  router.get('/', start.index);
12  router.get('/dashboard', dashboard.index);
13  router.get('/about', about.index);
14  router.get('/playlist/:id', playlist.index);
15  router.get('/playlist/:id/deletesong/:songid', playlist.deleteSong);
16  router.get('/dashboard/deleteplaylist/:id', dashboard.deletePlaylist);
17
18  router.post('/playlist/:id/updatesong/:songid', playlist.updateSong);
19  router.post('/playlist/:id/addsong', playlist.addSong);
20  router.post('/dashboard/addplaylist', dashboard.addPlaylist);
21
22  module.exports = router;
```

`action="/playlist/{.{../playlist.id}}/updatesong/{id}"`

Form Action matches Router call to a controller method

Update a Song – *playlist.js*



```
37
38
39  updateSong(request, response) {
40      → const playlistId = request.params.id;
41      → const songId = request.params.songid;
42      logger.debug("updating song " + songId);
43      → const alterSong = {
44          → title: request.body.title,
45          → artist: request.body.artist,
46      };
47      playlistStore.editSong(playlistId, songId, alterSong);
48      response.redirect('/playlist/' + playlistId);
49  },
50
51  };
52
```

Retrieve the `playlistId` and `songId` of the song that needs to be updated and the field values (title and artist) that need to be changed.

Update a Song – *playlist-store.js*



```
47
48  editSong(id, songId, songDetails) {
49      const playlist = this.getPlaylist(id);
50      const songs = playlist.songs;
51      const thepos = songs.findIndex(field=> field.id === songId);
52      songs[thepos].title=songDetails.title;
53      songs[thepos].artist=songDetails.artist;
54  },
55
56  };
57
58  module.exports = playlistStore;
59
```

Retrieve the position of the song to be updated in the songs array and update the song object with the field values passed in.