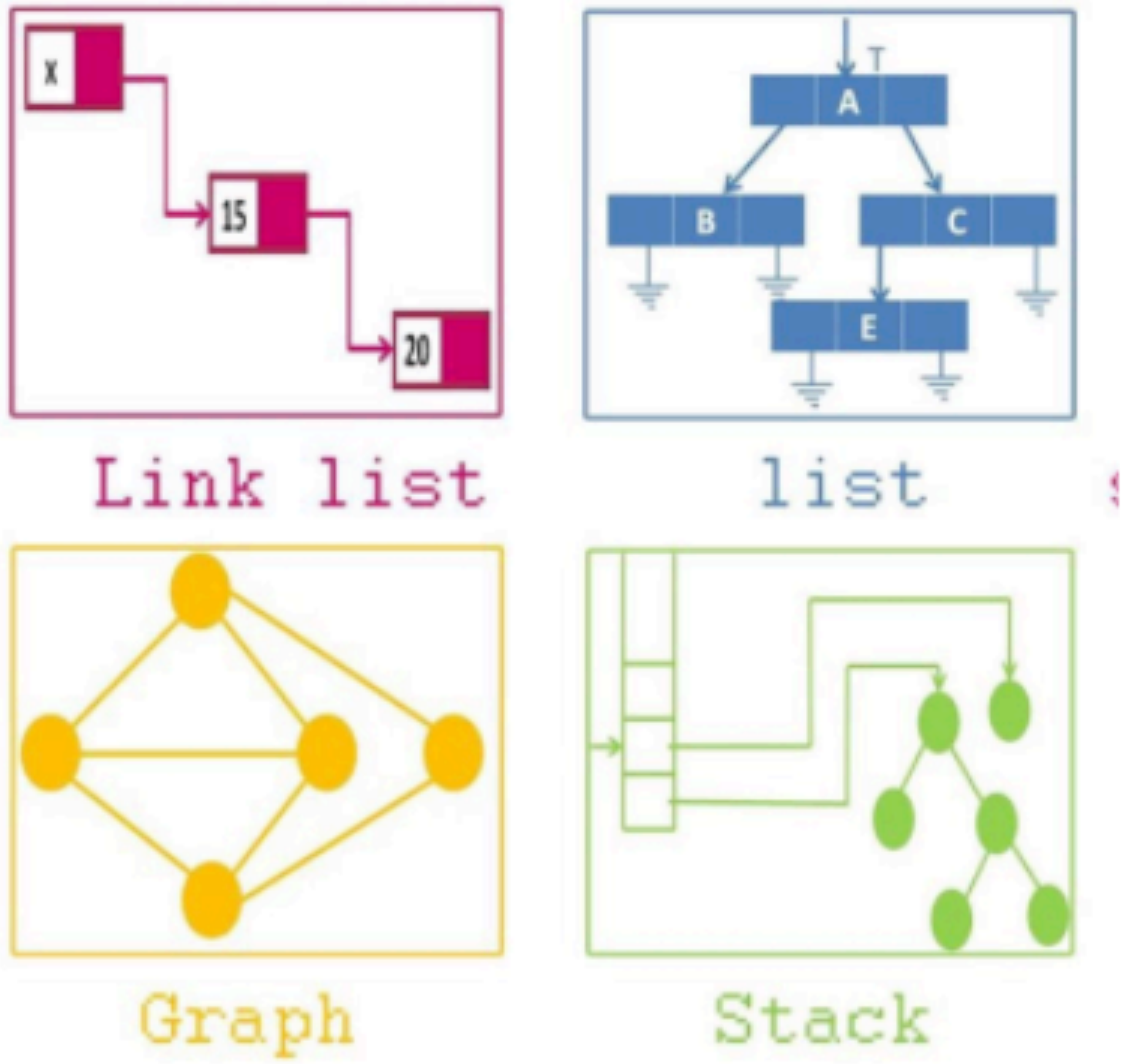
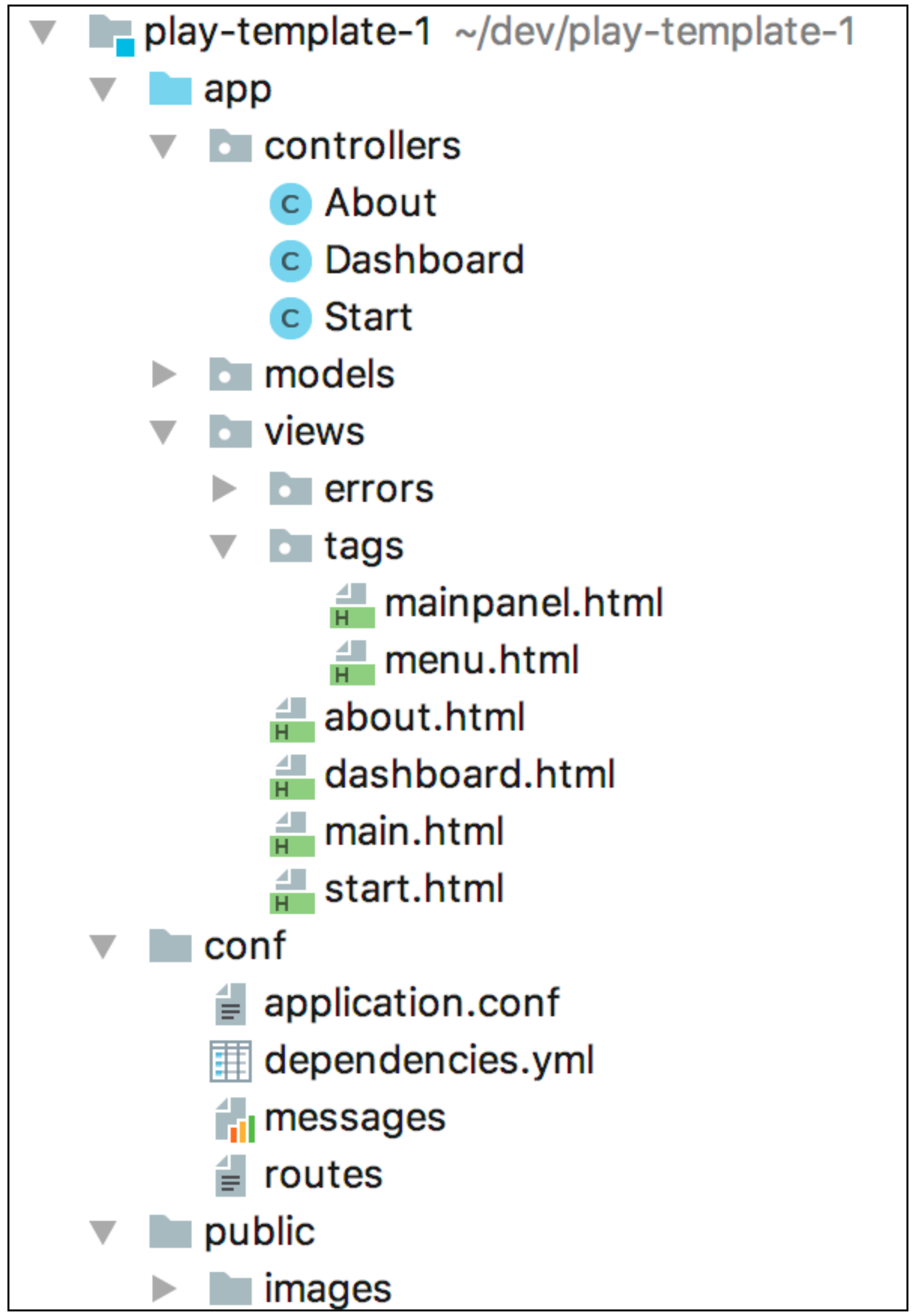
Models



model · arraylist · playlist ·
render · template

Application Structure

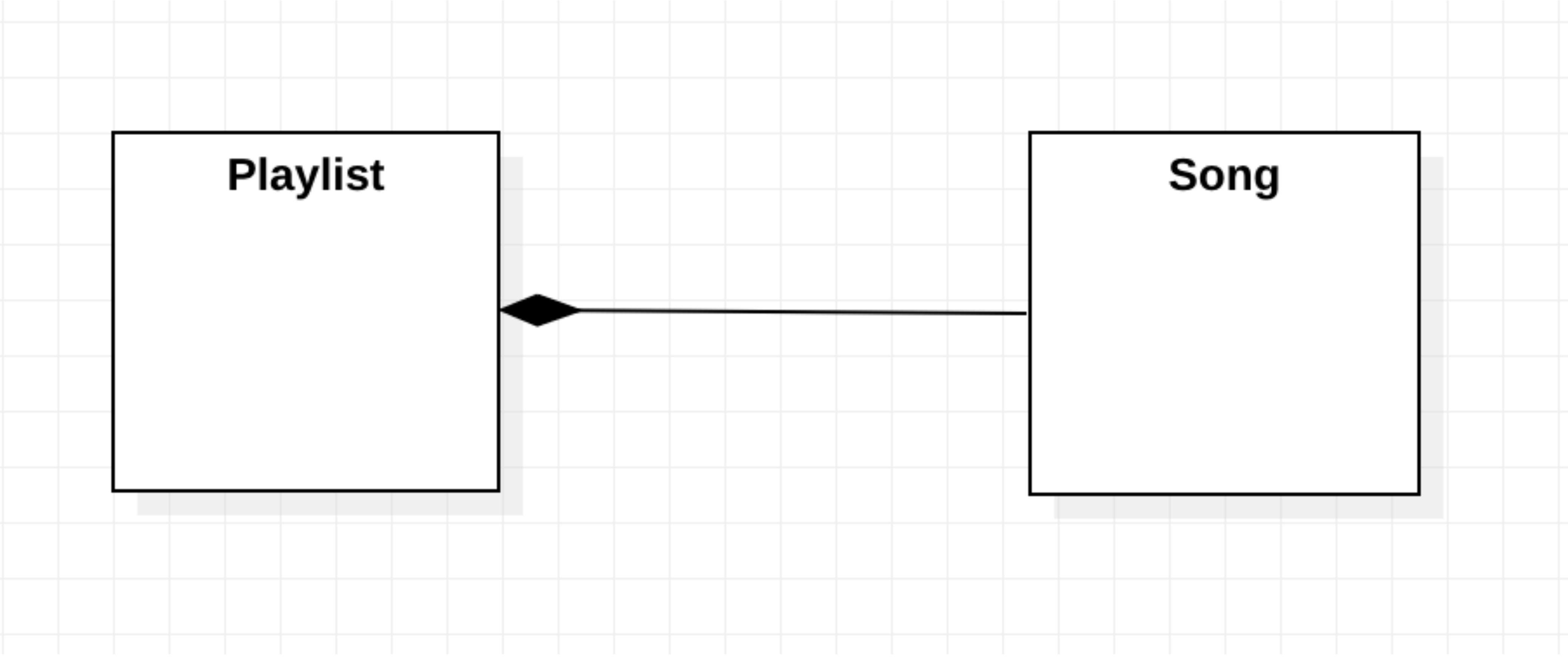


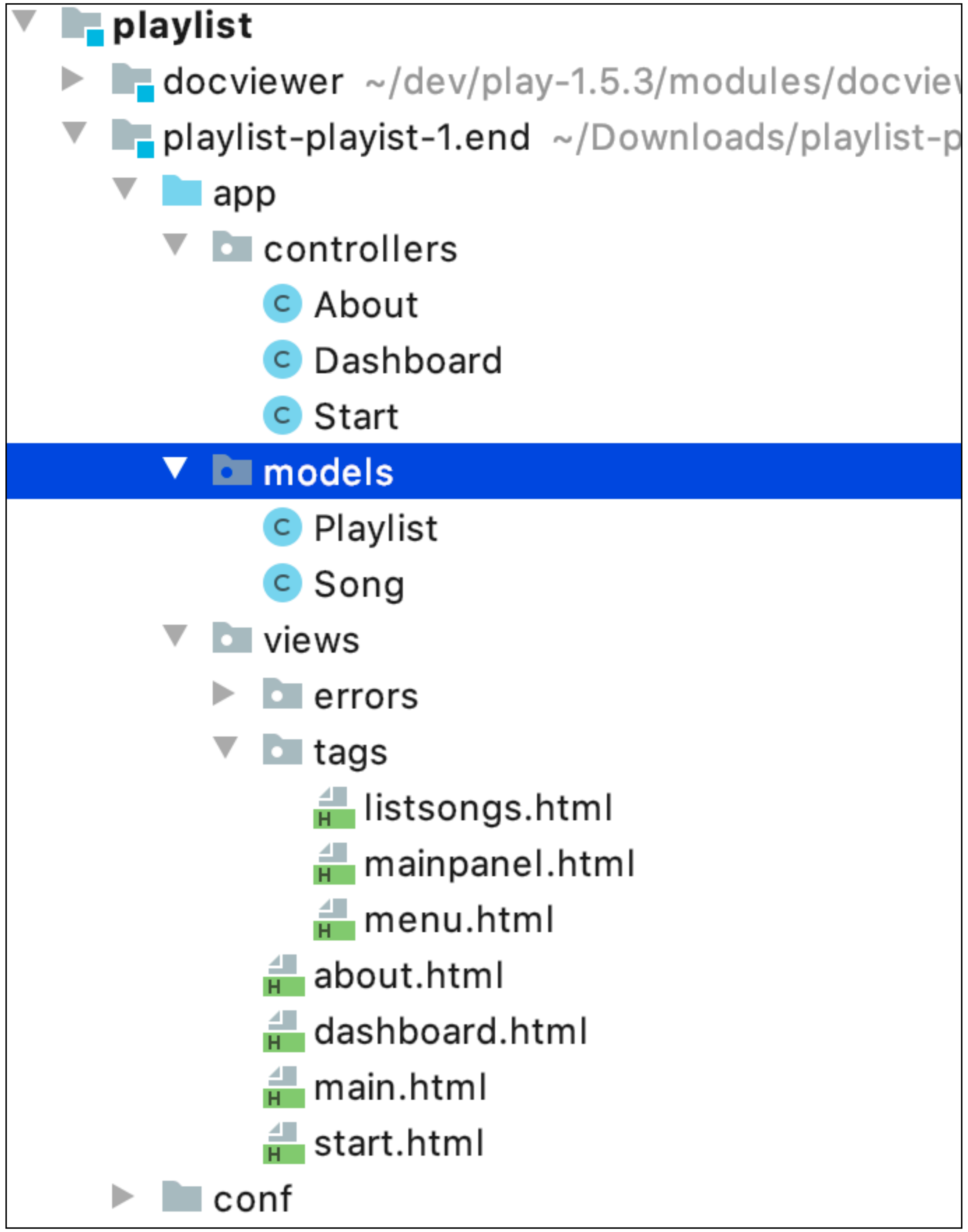
```
▼ play-template-1  ~/dev/play-template-1
  ▼ app
    ▼ controllers
        About
        Dashboard
        Start
    ▶ models
    ▼ views
      ▶ errors
      ▼ tags
          mainpanel.html
          menu.html
        about.html
        dashboard.html
        main.html
        start.html
  ▼ conf
      application.conf
      dependencies.yml
      messages
      routes
  ▼ public
    ▶ images
```

- App implements Routes + Model/View/Controller Architecture
- Back-end + Front-end collaborate to support structured, predictable application workflow

Models

- Models are the heart of a Web Application
- They represent the information system (the model) the application is concerned with.
- In Java - we model these as Classes



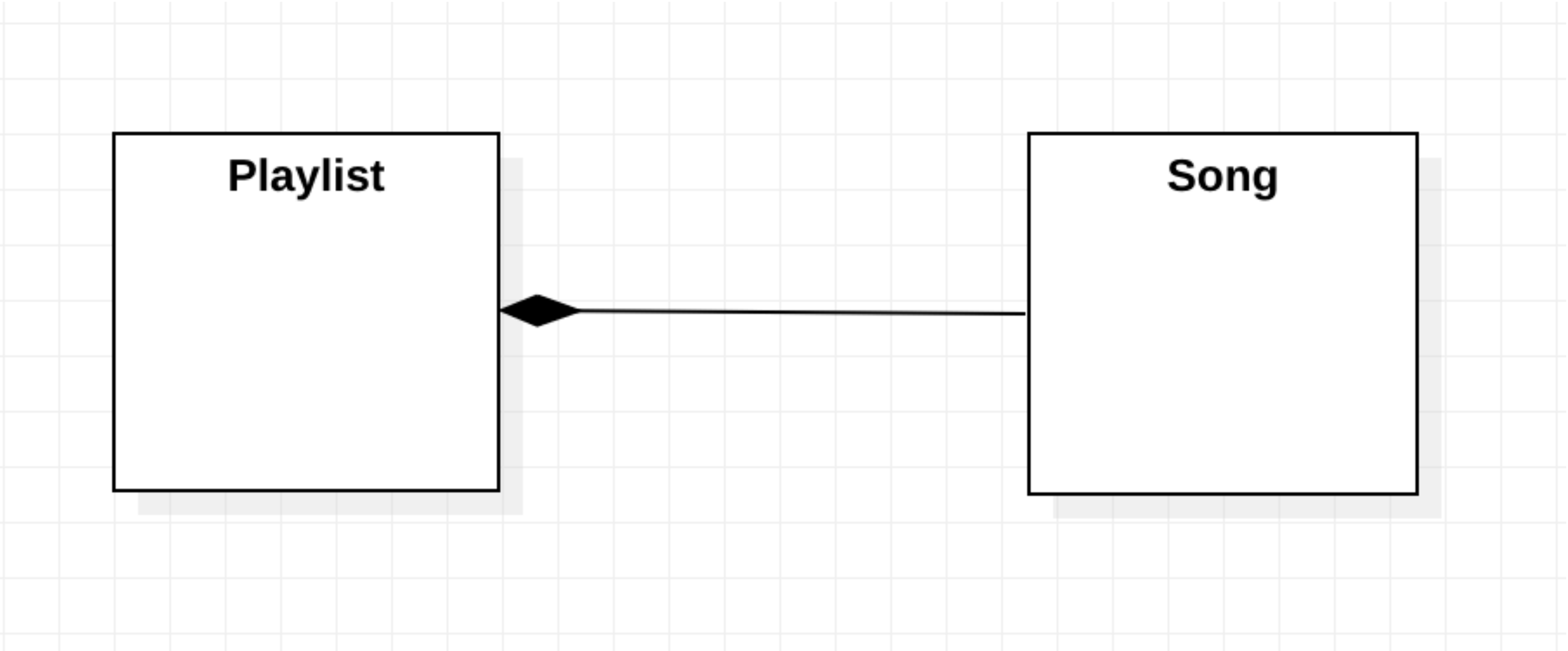


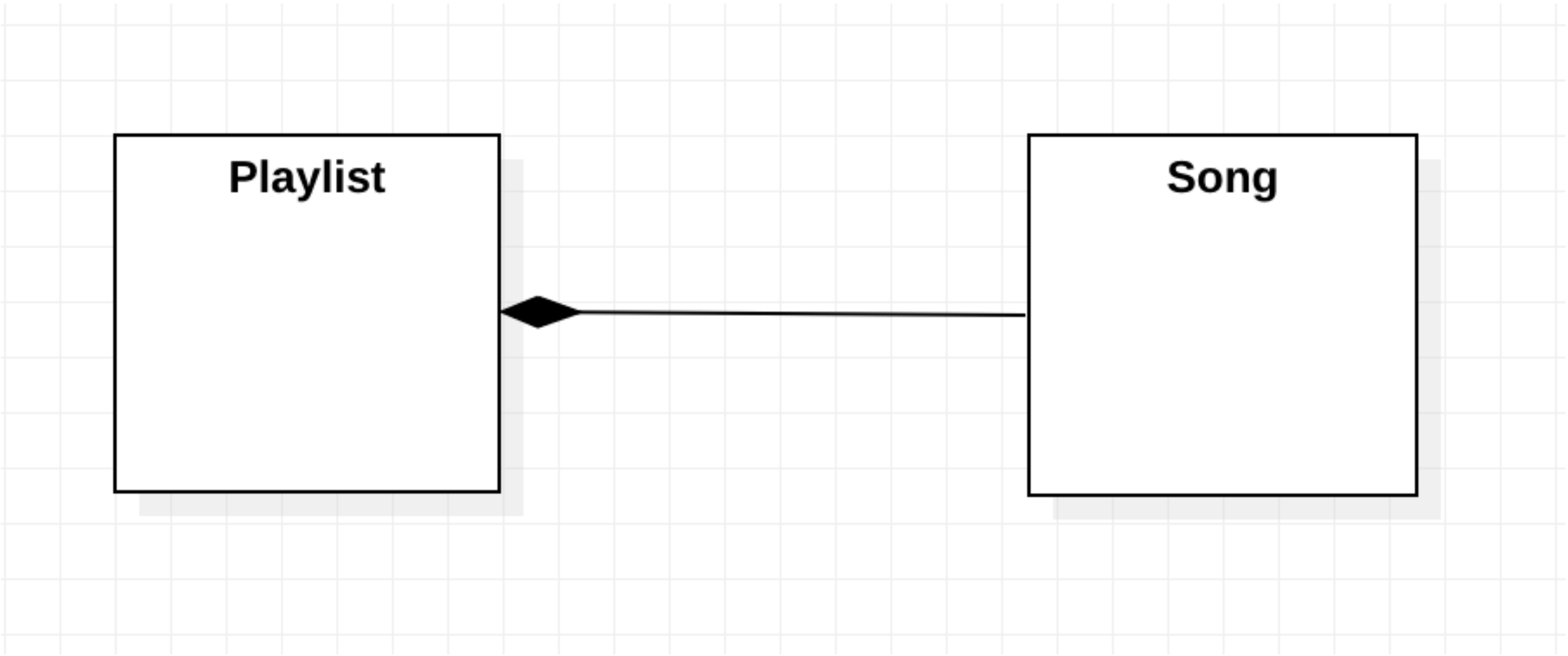
```
playlist
  docviewer  ~/dev/play-1.5.3/modules/docvie
  playlist-playist-1.end  ~/Downloads/playlist-p
    app
      controllers
        About
        Dashboard
        Start
      models
        Playlist
        Song
      views
        errors
        tags
          listsongs.html
          mainpanel.html
          menu.html
        about.html
        dashboard.html
        main.html
        start.html
    conf
```

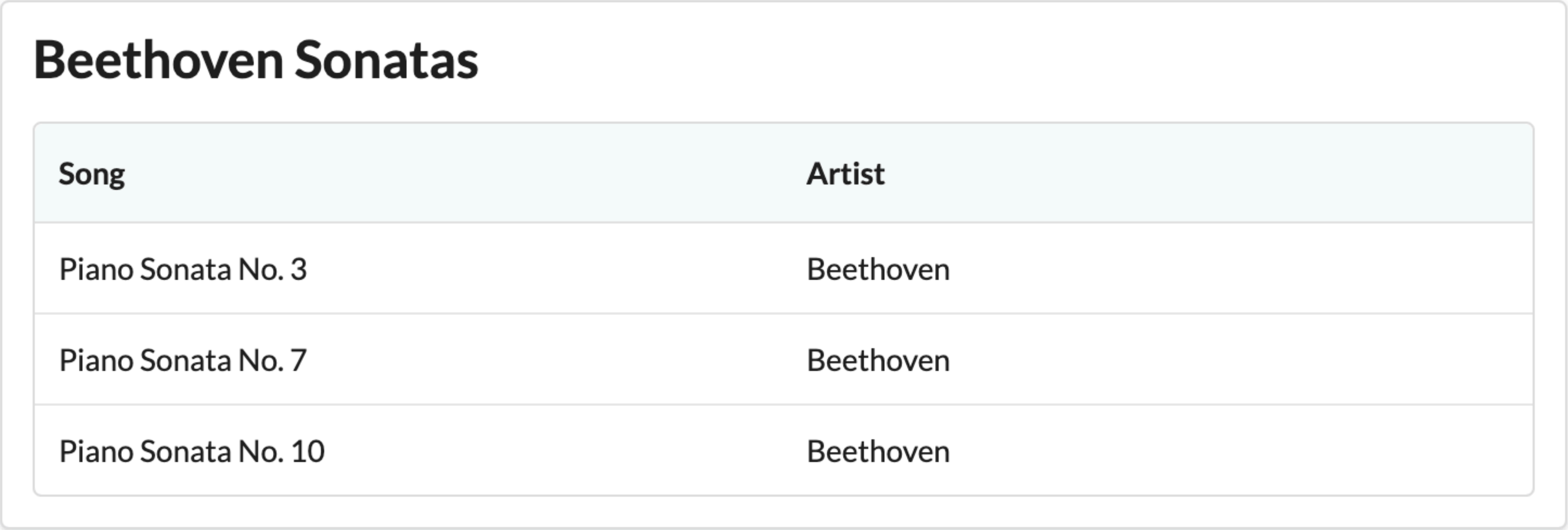
Playlist Model

```java
public class Playlist
{
  public String title;
  public List<Song> songs = new ArrayList<Song>();

  public Playlist(String title)
  {
    this.title = title;
  }
}
```

```java
public class Song
{
  public String title;
  public String artist;

  public Song(String title, String artist)
  {
    this.title = title;
    this.artist = artist;
  }
}
```

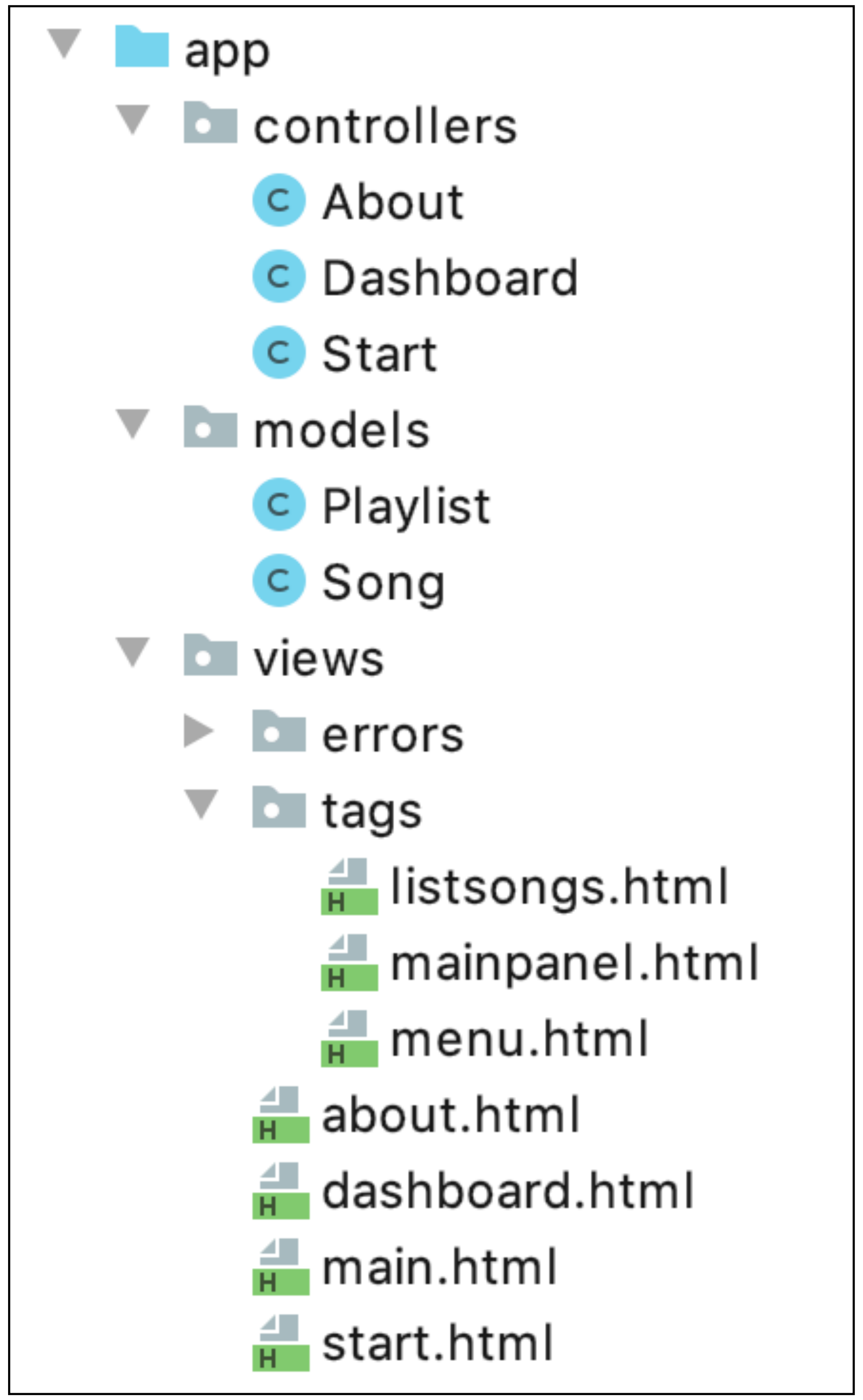






Beethoven Sonatas

Song	Artist
Piano Sonata No. 3	Beethoven
Piano Sonata No. 7	Beethoven
Piano Sonata No. 10	Beethoven



```
▼ app
    ▼ controllers
        About
        Dashboard
        Start
    ▼ models
        Playlist
        Song
    ▼ views
        ► errors
        ▼ tags
            listsongs.html
            mainpanel.html
            menu.html
        about.html
        dashboard.html
        main.html
        start.html
```

Song + Playlist Models

Creating Songs

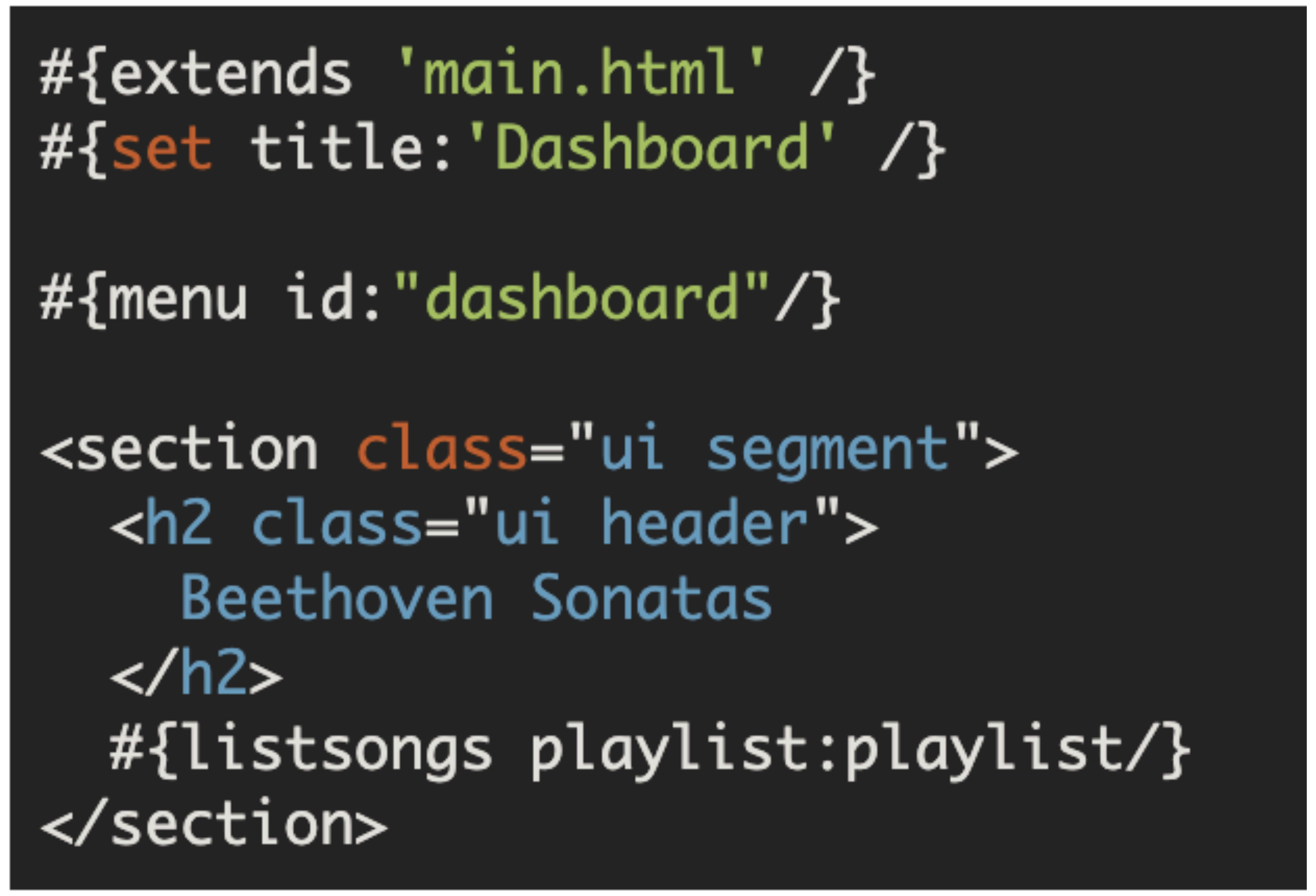
Creating an (empty) playlist

Adding the songs to the playlist

Sending the list to the view

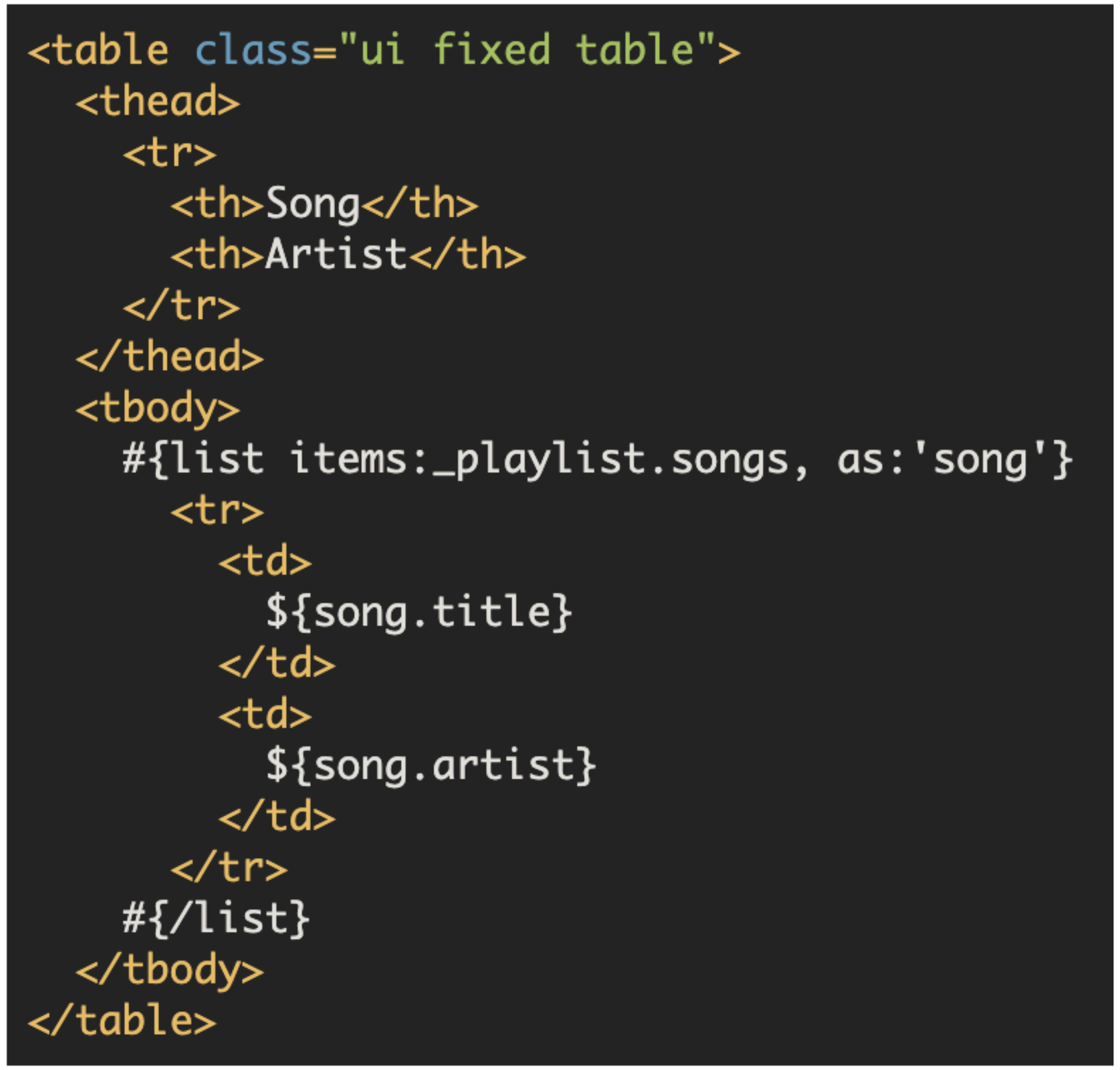
```
public class Dashboard extends Controller
{
  public static void index() {

    Song s1 = new Song("Piano Sonata No. 3", "Beethoven");
    Song s2 = new Song("Piano Sonata No. 7", "Beethoven");
    Song s3 = new Song("Piano Sonata No. 10", “Beethoven");

    Playlist playlist = new Playlist("Beethoven Sonatas");


    playlist.songs.add (s1);
    playlist.songs.add (s2);
    playlist.songs.add (s3);


    render ("dashboard.html", playlist);
  }
}
```

app/views/dashboard.html

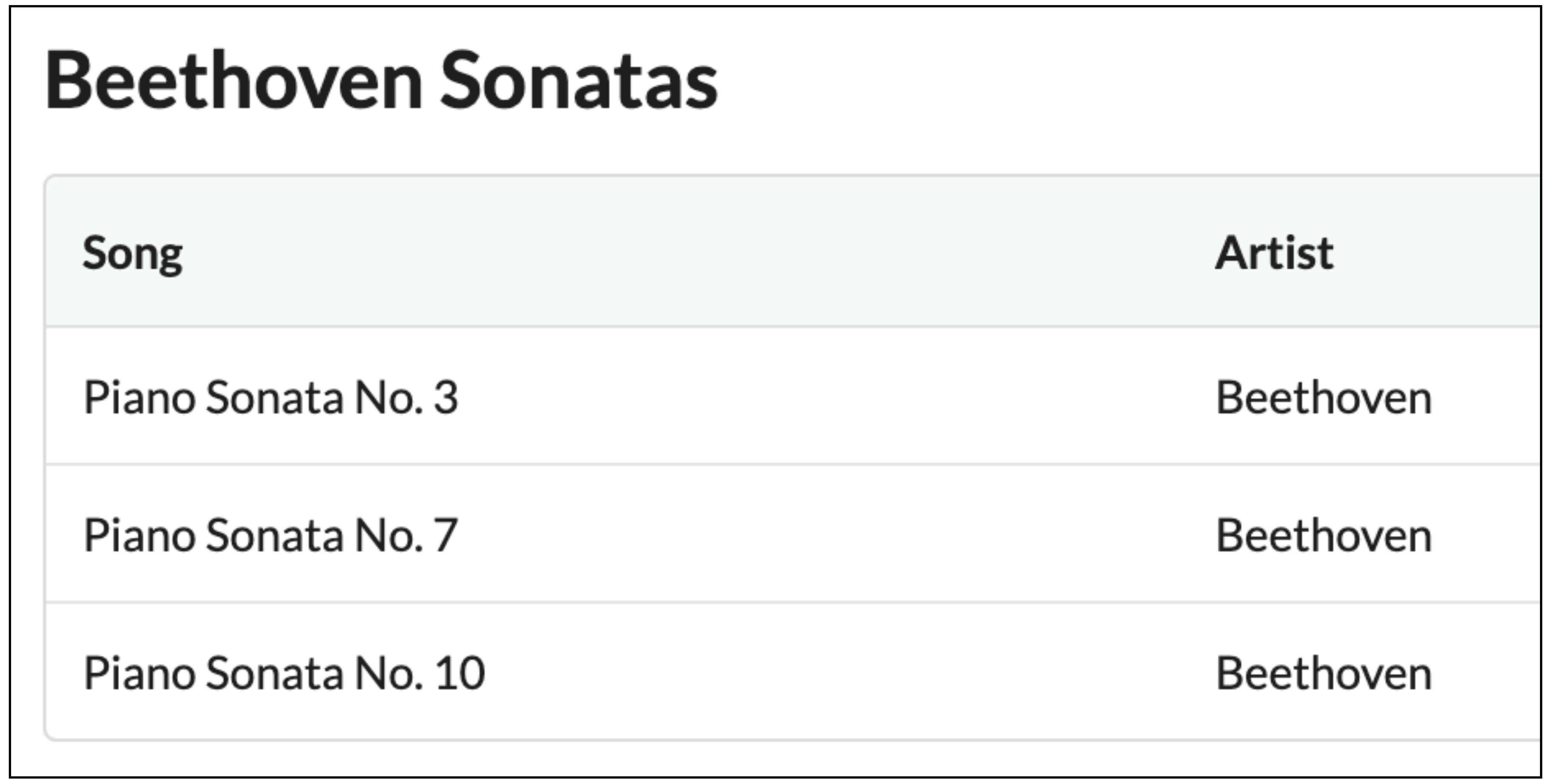


```
#{extends 'main.html' /}
#{set title:'Dashboard' /}

#{menu id:"dashboard"/}

<section class="ui segment">
  <h2 class="ui header">
    Beethoven Sonatas
  </h2>
  #{listsongs playlist:playlist/}
</section>
```

views/tags/listsongs.html



```
<table class="ui fixed table">
  <thead>
    <tr>
      <th>Song</th>
      <th>Artist</th>
    </tr>
  </thead>
  <tbody>
    #{list items:_playlist.songs, as:'song'}
      <tr>
        <td>
          ${song.title}
        </td>
        <td>
          ${song.artist}
        </td>
      </tr>
    #{/list}
  </tbody>
</table>
```

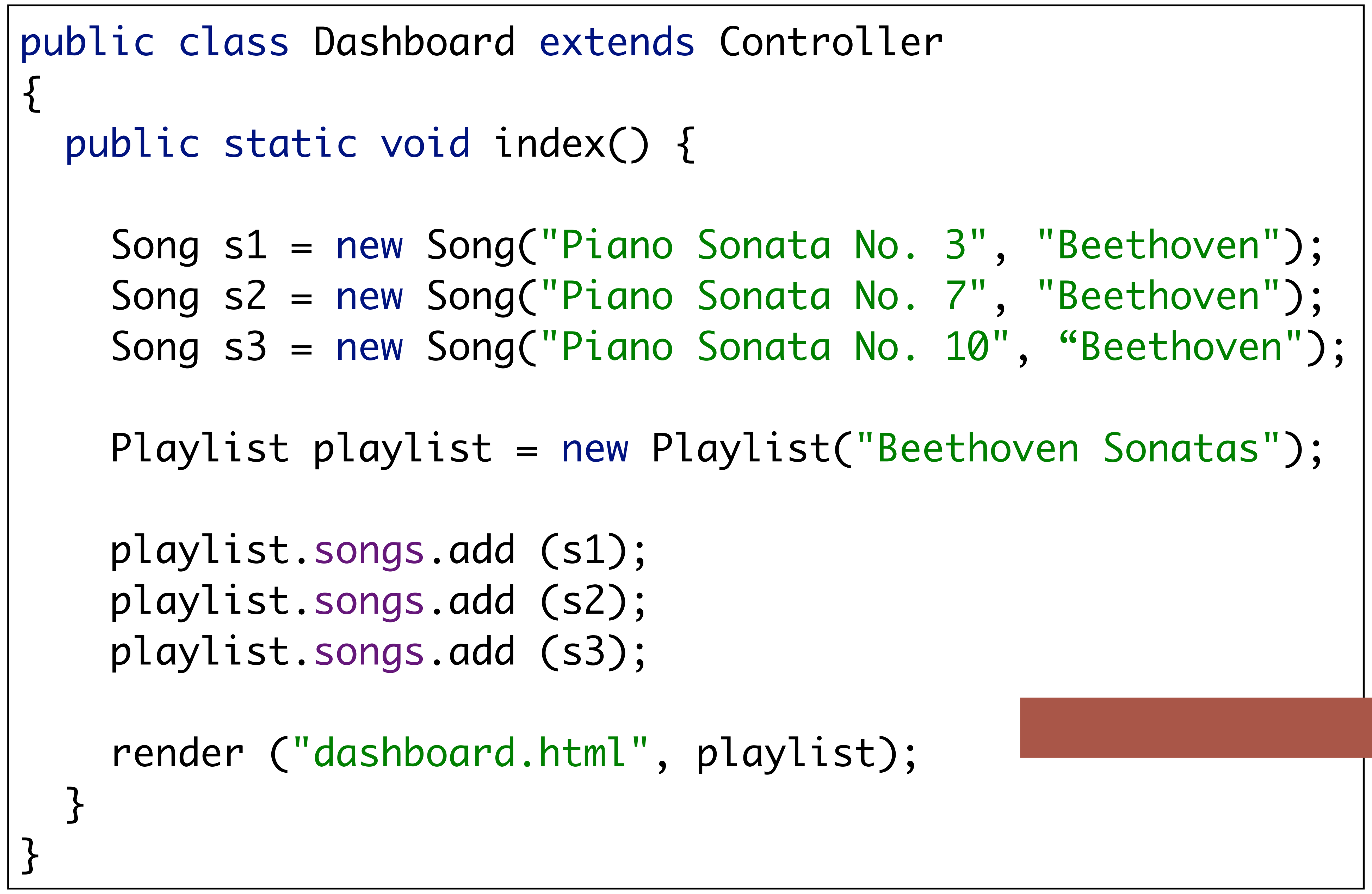


Beethoven Sonatas

Song	Artist
Piano Sonata No. 3	Beethoven
Piano Sonata No. 7	Beethoven
Piano Sonata No. 10	Beethoven

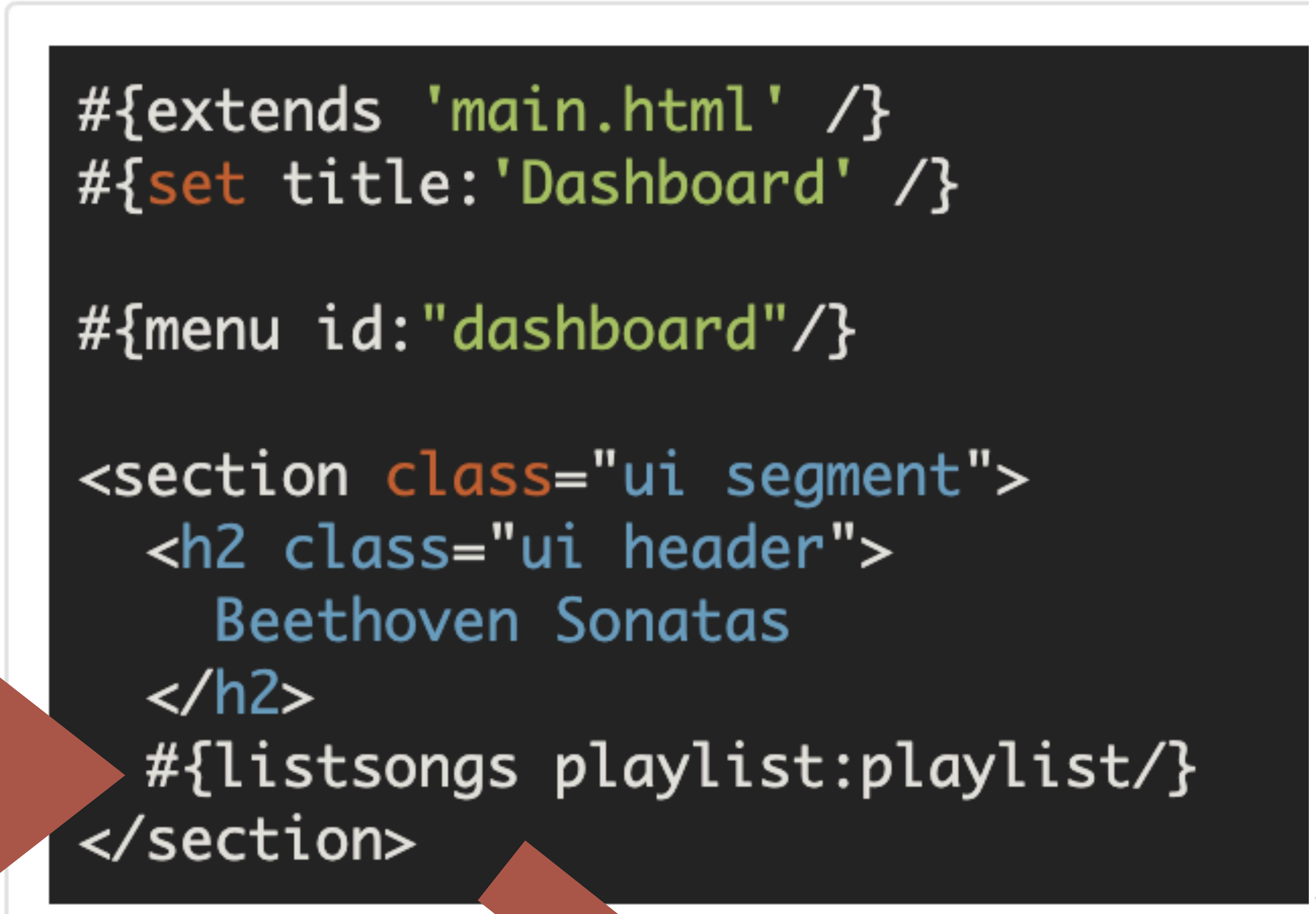
This is a new template language - part of the Play framework

Mixture of html + special constructs to include information from application ‘Model’



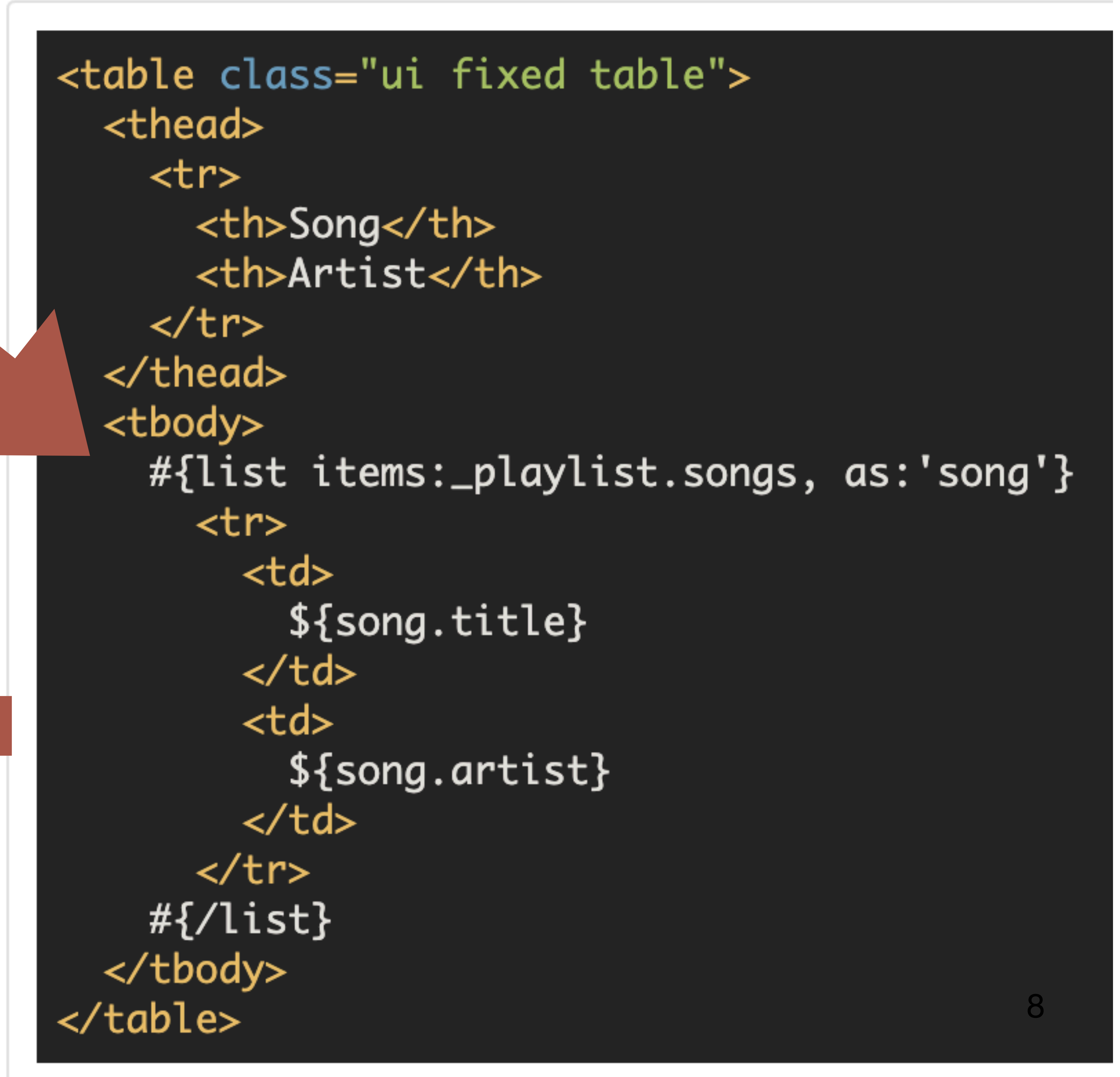
```
public class Dashboard extends Controller
{
  public static void index() {

    Song s1 = new Song("Piano Sonata No. 3", "Beethoven");
    Song s2 = new Song("Piano Sonata No. 7", "Beethoven");
    Song s3 = new Song("Piano Sonata No. 10", “Beethoven");

    Playlist playlist = new Playlist("Beethoven Sonatas");

    playlist.songs.add (s1);
    playlist.songs.add (s2);
    playlist.songs.add (s3);

    render ("dashboard.html", playlist);
  }
}
```

app/views/dashboard.html

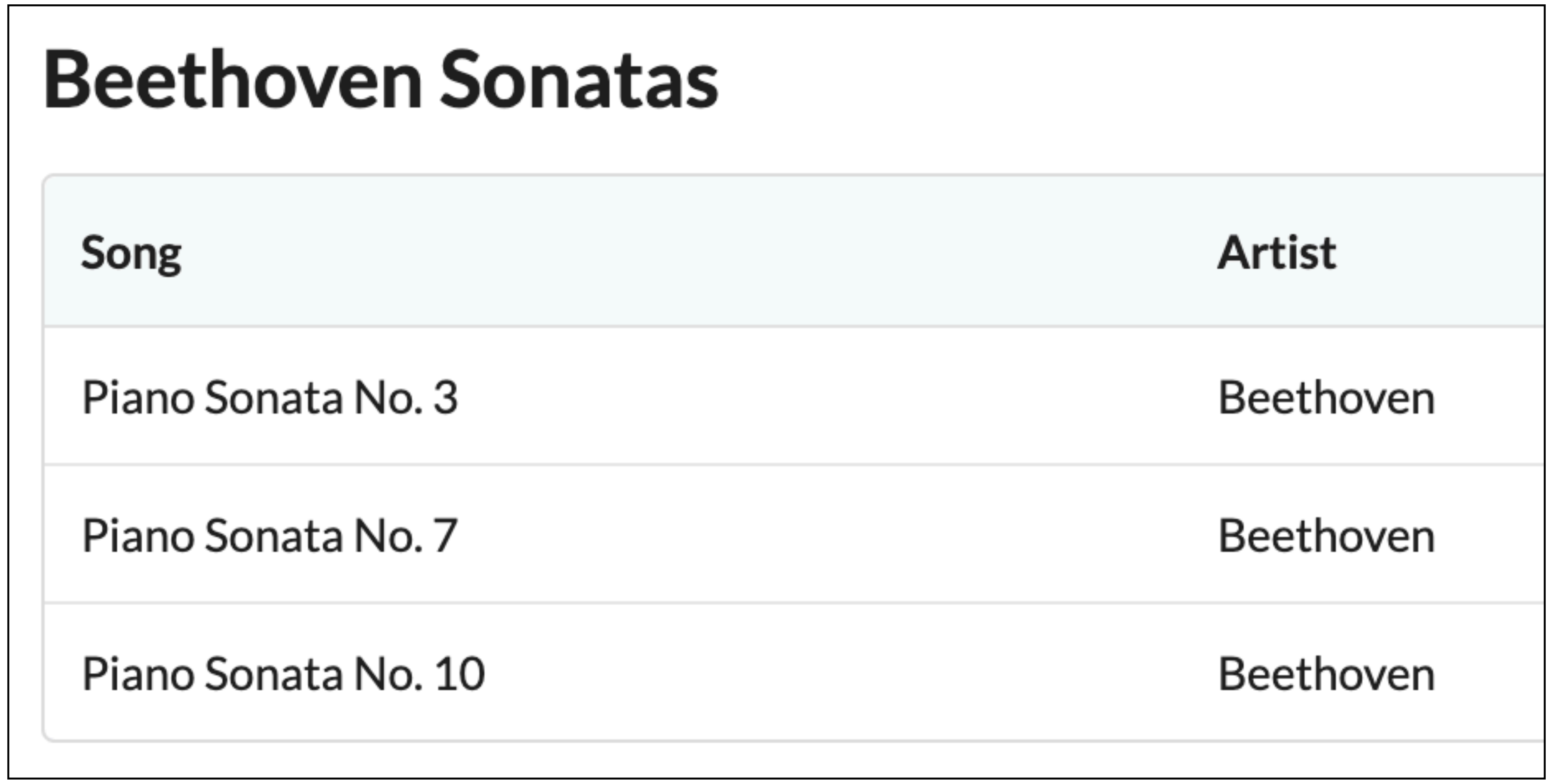


```
#{extends 'main.html' /}
#{set title:'Dashboard' /}

#{menu id:"dashboard"/}

<section class="ui segment">
  <h2 class="ui header">
    Beethoven Sonatas
  </h2>
  #{listsongs playlist:playlist/}
</section>
```

views/tags/listsongs.html

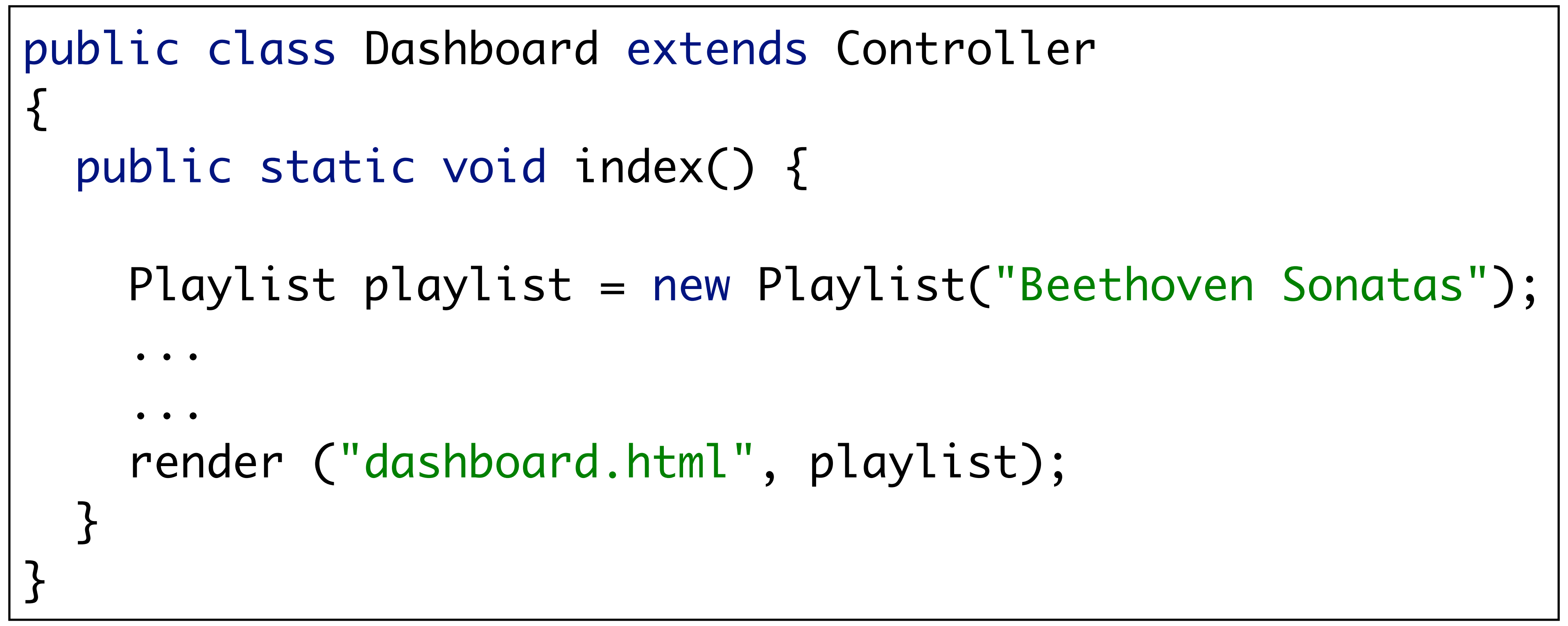


```
<table class="ui fixed table">
  <thead>
    <tr>
      <th>Song</th>
      <th>Artist</th>
    </tr>
  </thead>
  <tbody>
    #{list items:_playlist.songs, as:'song'}
      <tr>
        <td>
          ${song.title}
        </td>
        <td>
          ${song.artist}
        </td>
      </tr>
    #{/list}
  </tbody>
</table>
```



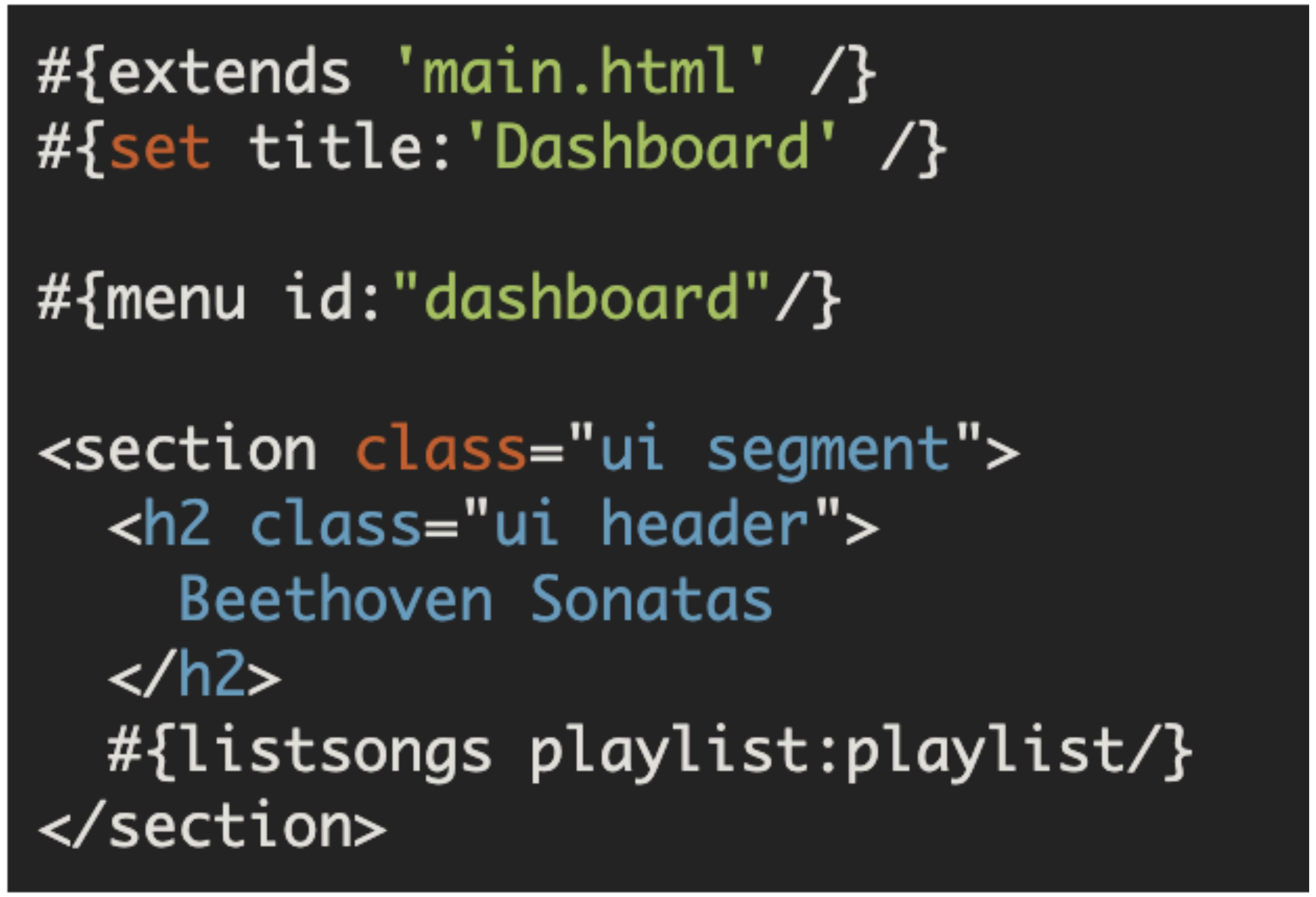
Beethoven Sonatas

Song	Artist
Piano Sonata No. 3	Beethoven
Piano Sonata No. 7	Beethoven
Piano Sonata No. 10	Beethoven



```
public class Dashboard extends Controller
{
  public static void index() {

    Playlist playlist = new Playlist("Beethoven Sonatas");
    ...
    ...
    render ("dashboard.html", playlist);
  }
}
```

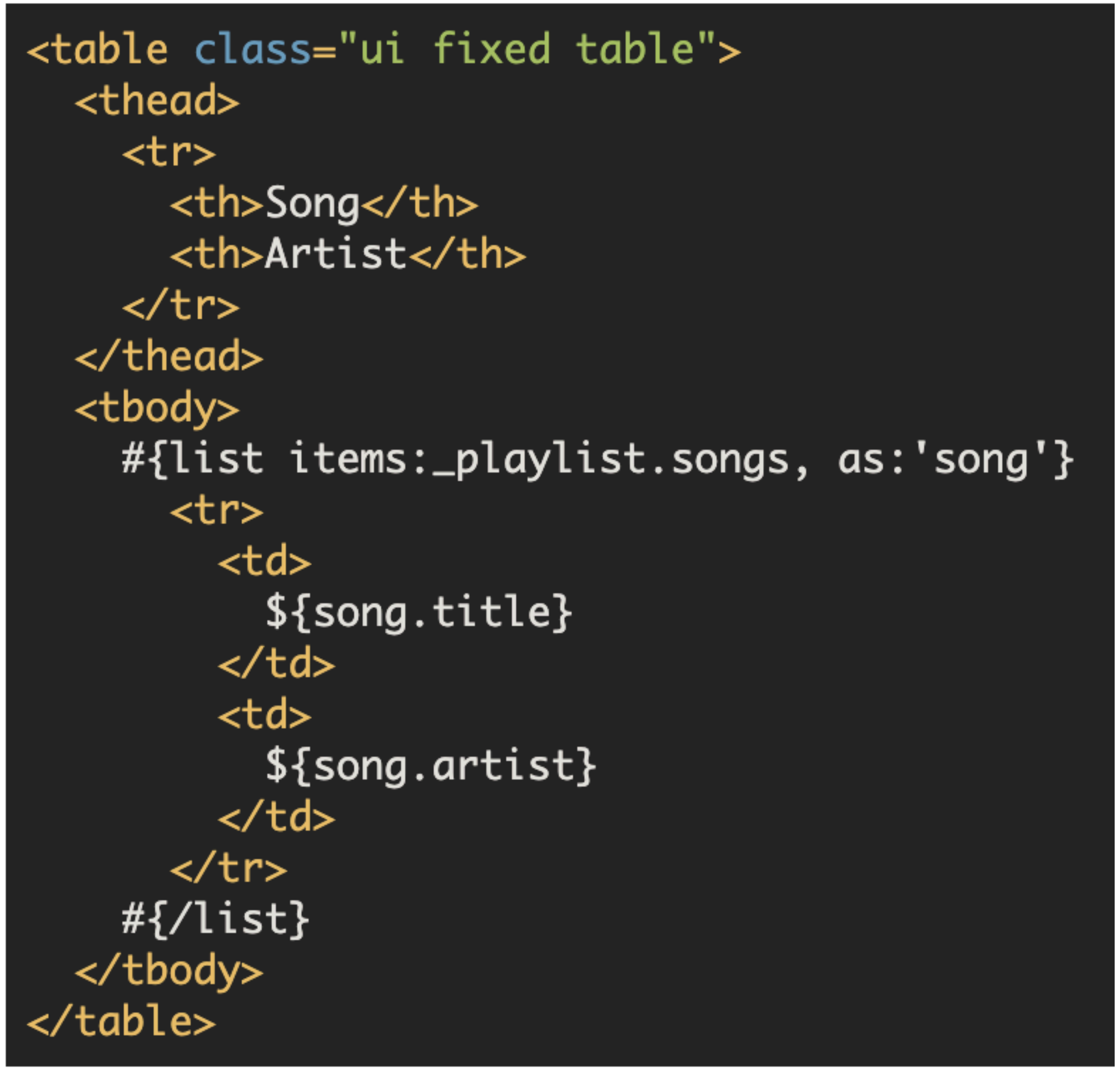
dashboard.hbs



```
#{extends 'main.html' /}
#{set title:'Dashboard' /}

#{menu id:"dashboard"/}

<section class="ui segment">
  <h2 class="ui header">
    Beethoven Sonatas
  </h2>
  #{listsongs playlist:playlist/}
</section>
```

Include a partial (tag) called listsongs
Pass the playlist object to this partial

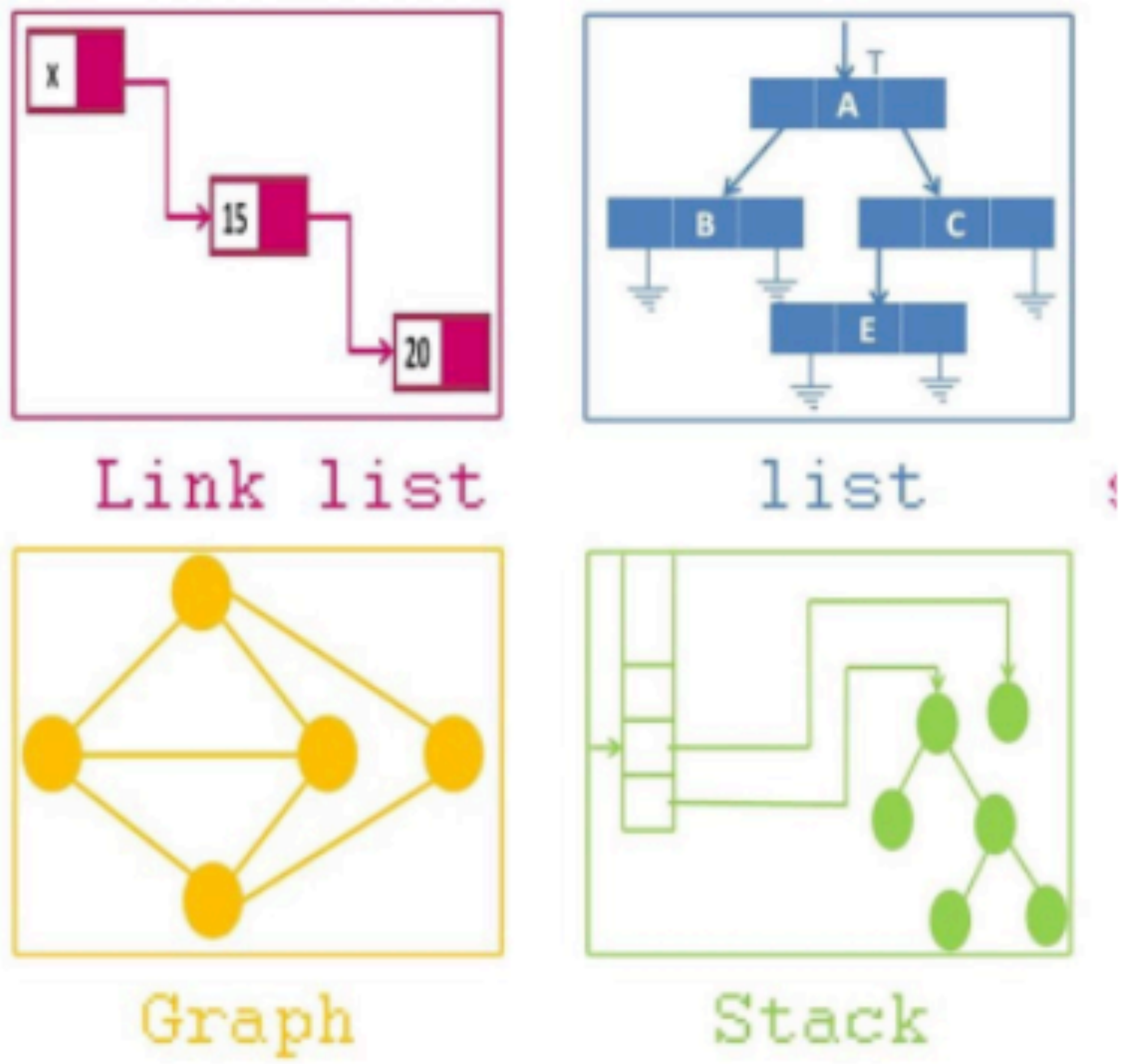
listsongs.hbs



```
<table class="ui fixed table">
  <thead>
    <tr>
      <th>Song</th>
      <th>Artist</th>
    </tr>
  </thead>
  <tbody>
    #{list items:_playlist.songs, as:'song'}
      <tr>
        <td>
          ${song.title}
        </td>
        <td>
          ${song.artist}
        </td>
      </tr>
    #{/list}
  </tbody>
</table>
```

Loop through the playlist.songs ArrayList and populate a table of all songs

Models



model · arraylist · playlist · render · template