# Exercise: Bootstrap

# IRunes

In the previous exercise you should have implemented the **IRunes** **application** – a simple **music store application**. Due to the fact that **exceptional design** was **not required**, the pages, which the application supported, were implemented – using only HTML. Thus, they looked like something implemented 50 years ago – ugly and simple.

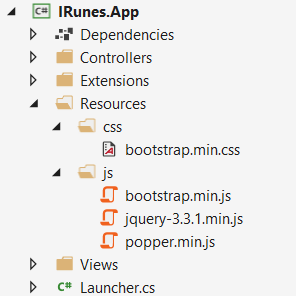
But this time, the case is different... We now have a powerful friend to our side – **Bootstrap**. Let’s use it to make the pages more beautiful.

# Initial Configuration

But is our Server capable of handling resource requests? Because requesting bootstrap.css is a resource request, which must be handled differently. Well, let’s configure it to do so.

## Resources Folder

The Resources Folder will be in the Application Project. Each application has its own resources.

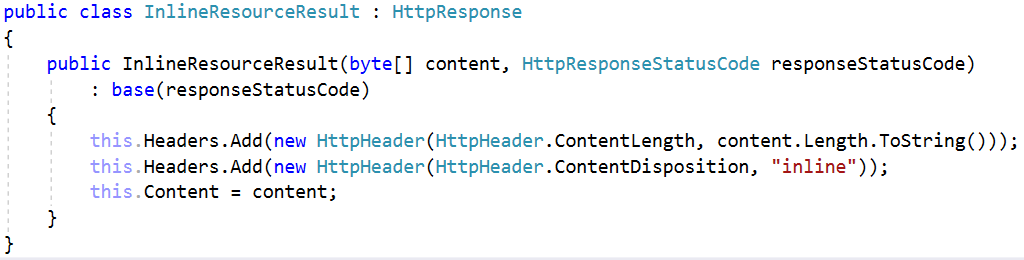


**Create** the **folder** and **divide** the **different resources**. (**download** the files shown above, locally, from their respective sites). **NOTE**: There is **no particular meaning** to the resource files being separated into folders named as the resources’ extensions, it’s just well-structured resource folder.

## Inline Resource Result

First, we need to create a Response, which will be returned with the Resource File Contents.

Create a class named InlineResourceResult, which we will use for that purpose. It should look like this:



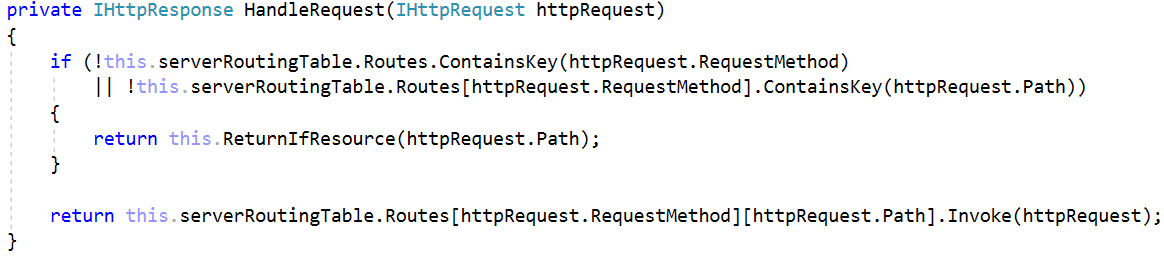
As you can see the content given in the constructor is byte[]. That’s because the resource might be a media file. For example: favicon.ico.

We also set the Content-Length and Content-Disposition Headers. The Content-Disposition is inline, as this particular Result class is used to **return** only **inline resource**, which are used in the web pages.

**NOTE**: It is not quite good to depend on the **Client’s** **Browser** to set the Content-Type header for you, but for now we will. (Will be fixed in the future).

## ConnectionHandler

The other class we need to change is the ConnectionHandler. Particularly, the way Requests are **Handled**.   
Modify the HandlerRequest() method to look like this:



The ReturnIfResource() method should check for a **resource** with the **given name** (in the path of the Request, for example: /css/bootstrap.min.css), in the Resources folder of the Application.

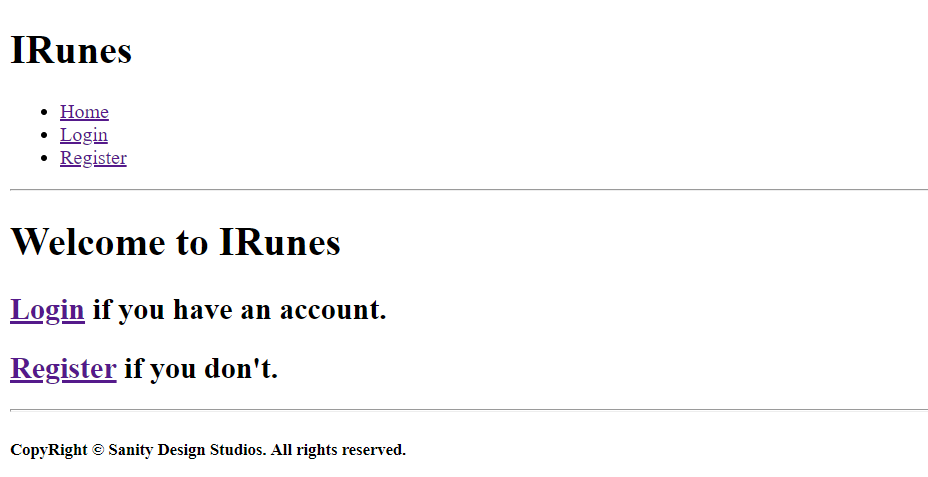
* If there is, an InlineResourceResponse should be returned, with the Resource File’s contents.
* If there isn’t, a Not Found HttpResponse should be returned.

However, the method’s **implementation** is up to **you**. 😊

# Templates

You will be shown how the **template looked before**, and **how it should look now**. Style it **using Bootstrap**, to match the given screenshot. Be as precise as you can.

## Index (guest, logged-out) (route=”/Home/Index”, route=”/”)



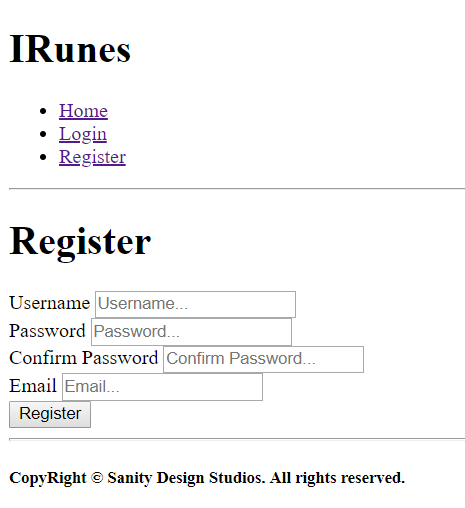


## Login (guest, logged-out) (route=”/Users/Login”)



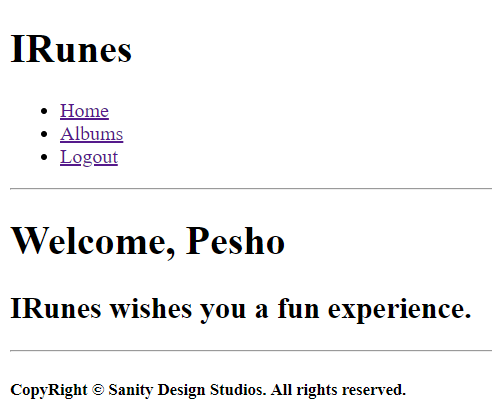


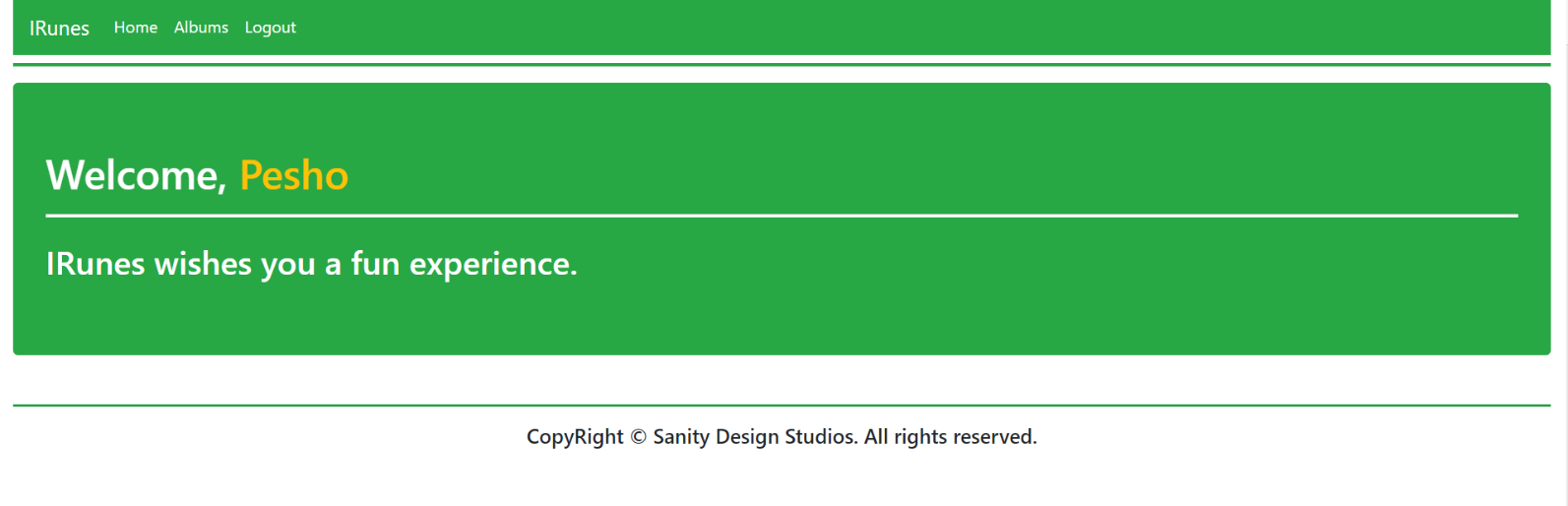
## Register (guest, logged-out) (route=”/Users/Register”)



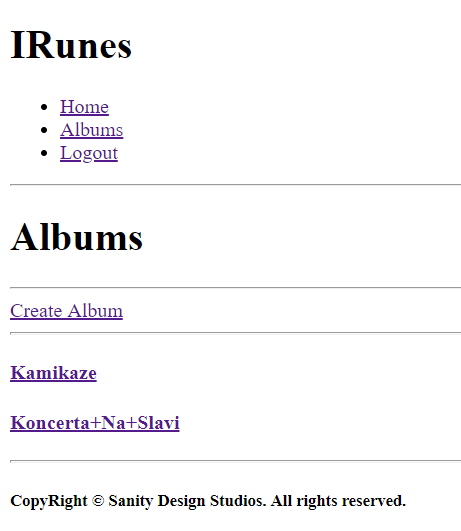


## Index (user, logged-in) (route=”/Home/Index”, route=”/”)



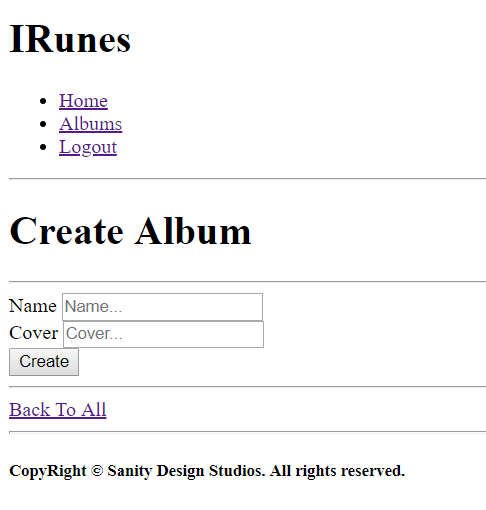


## All Albums (user, logged-in) (route=”/Albums/All”)



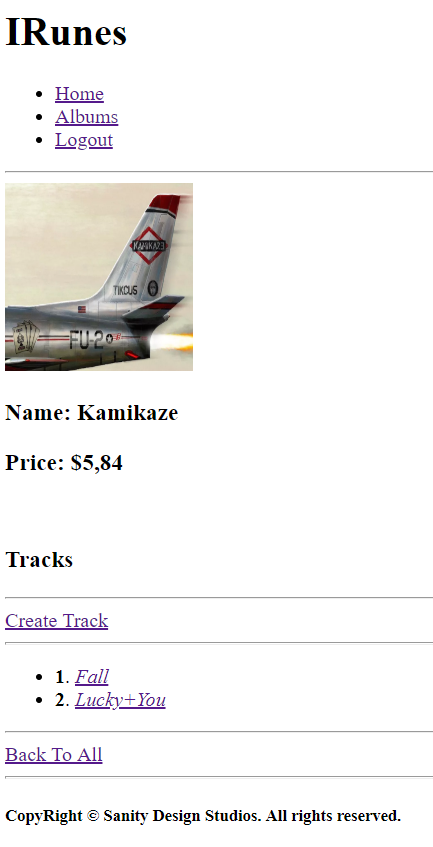
****

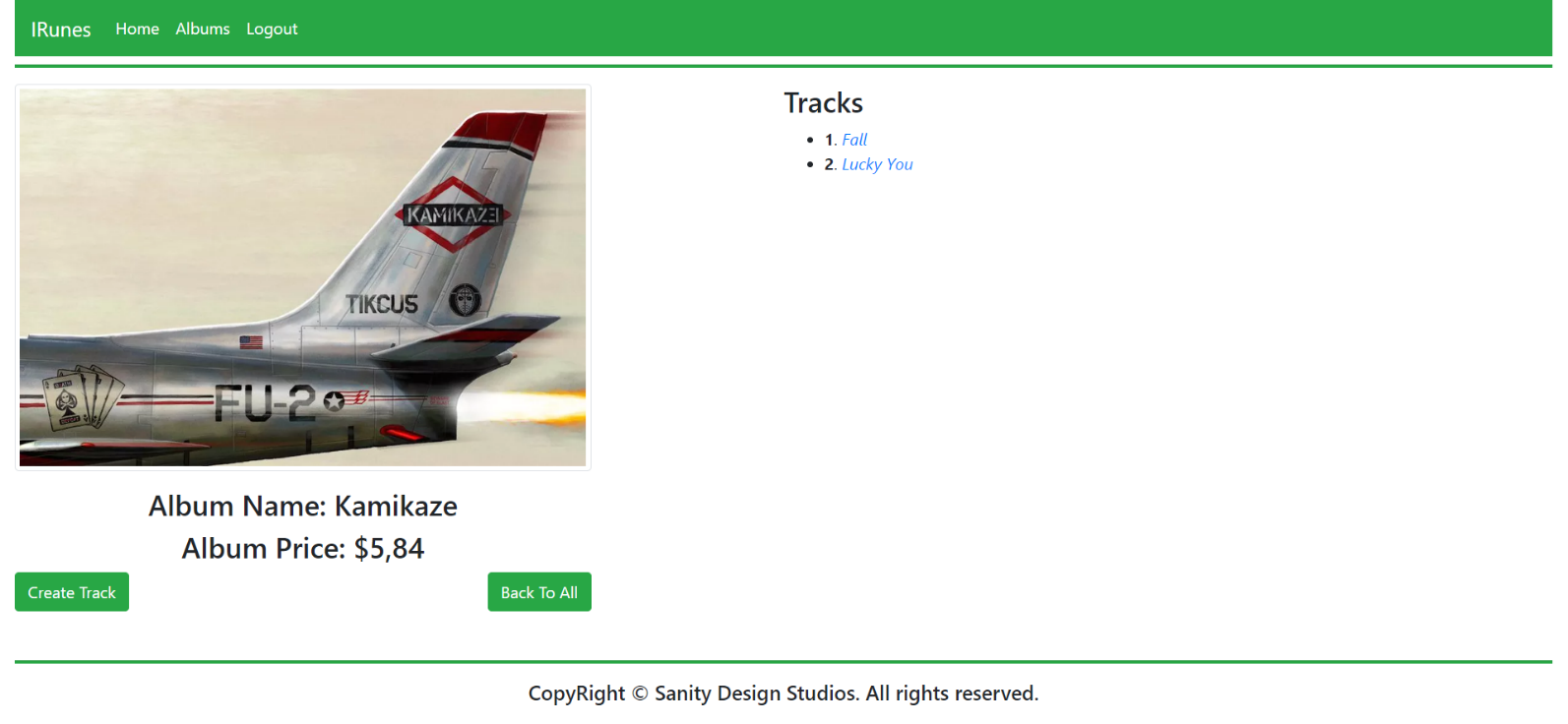
## Album Create (user, logged-in) (route=”/Albums/Create”)



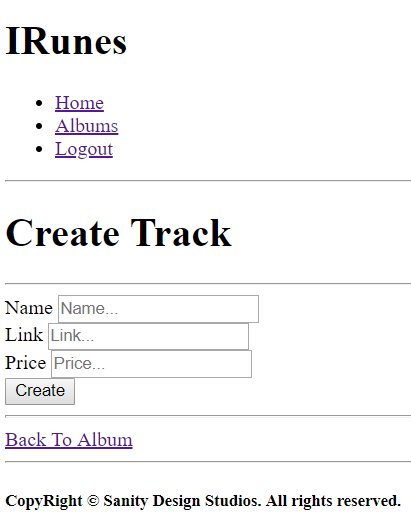


## Album Details (user, logged-in) (route=”/Albums/Details?id={albumId}”)



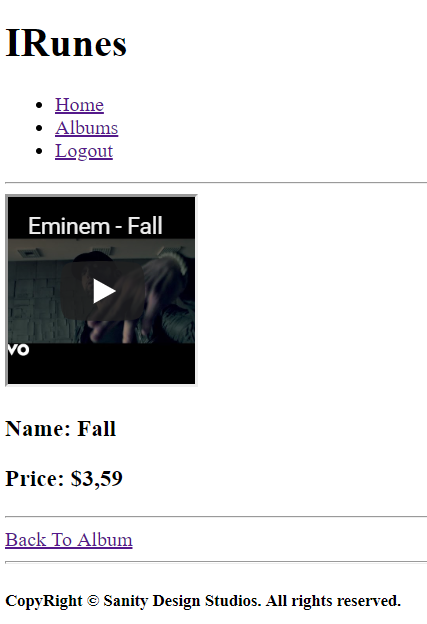


## Track Create (user, logged-in) (route=”/Tracks/Create?albumId={albumId}”)





## Track Details (user, logged-in) (route=”/Tracks/Details?albumId={albumId}&trackId={trackId}”)





## Hints

If you don’t know how to do something, just search for it on the internet. The **Bootstrap Documentation** and **Community** are quite helpful.

Bootstrap does not provide style for Horizontal lines (<hr />). To implement the lines shown in the screenshots above with the appropriate height. Just use the following way (<hr style=”height: 2px”)