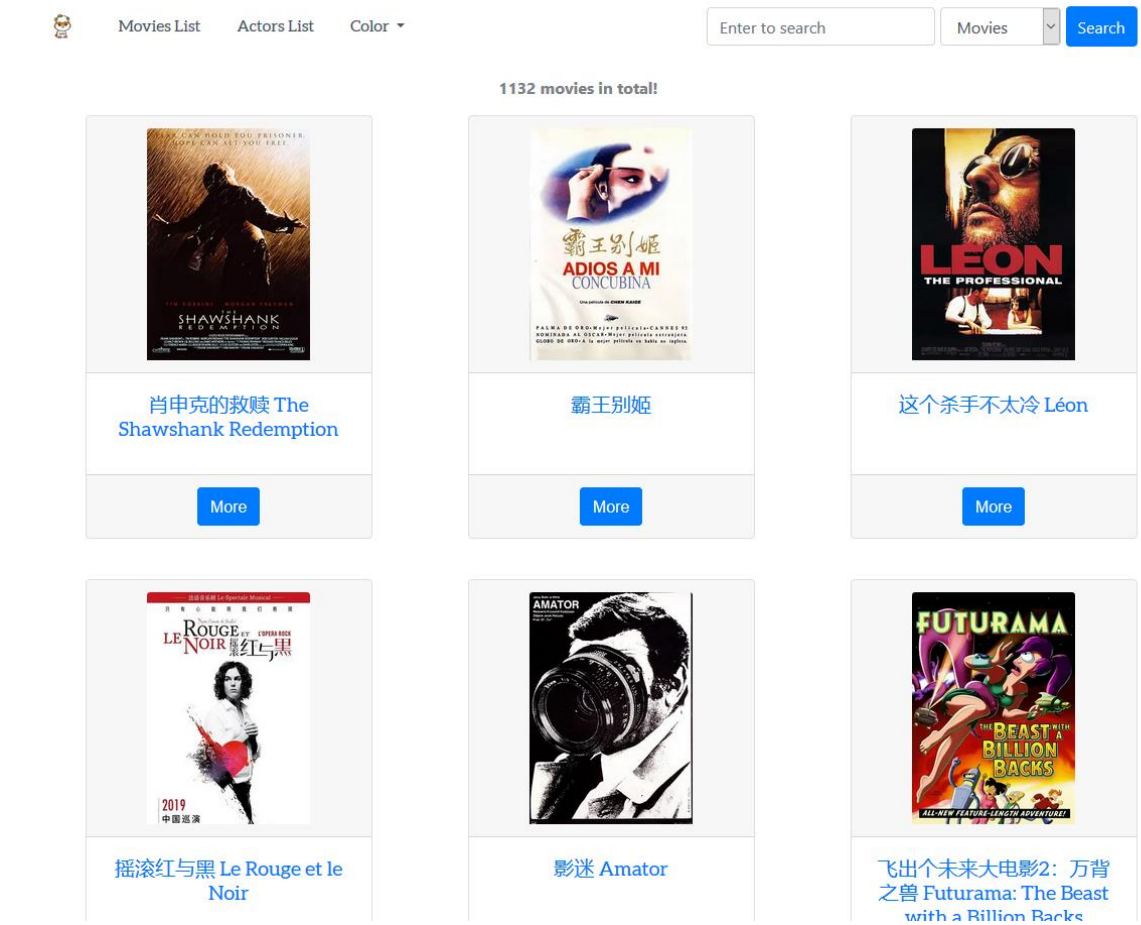
爬虫+搜索网站设计文档

1. 总体结构介绍

1.1 网站简介

本网站是搭载部分豆瓣电影及其相关演员信息的展示网站。我首先利用网络爬虫从豆瓣电影上爬取了 1132 部电影及与电影相关的 6848 位演员的信息，之后我将爬下来的内容保存到数据库中，并从制作了本网站用于展示我爬取到的信息。用户可以进入网站查看爬取到的电影以及演员列表，可以进入详情页查看每部电影或每个演员的简介、照片和其他相关内容。特别地，用户还可以输入关键字进行搜索，网站会向用户展示匹配的电影或演员信息。在搜索结果展示页中，网站还会为用户标识出匹配的关键词位置。本网站利用 bootsrap 和 css 样式进行了界面的化，整体界面风格满足了简约而美观的要求（simple&perfect），这也是我的一个设计理念。用户还可以通过顶部的"Color"下拉框更换网站的主题颜色。下面是网站的主界面：

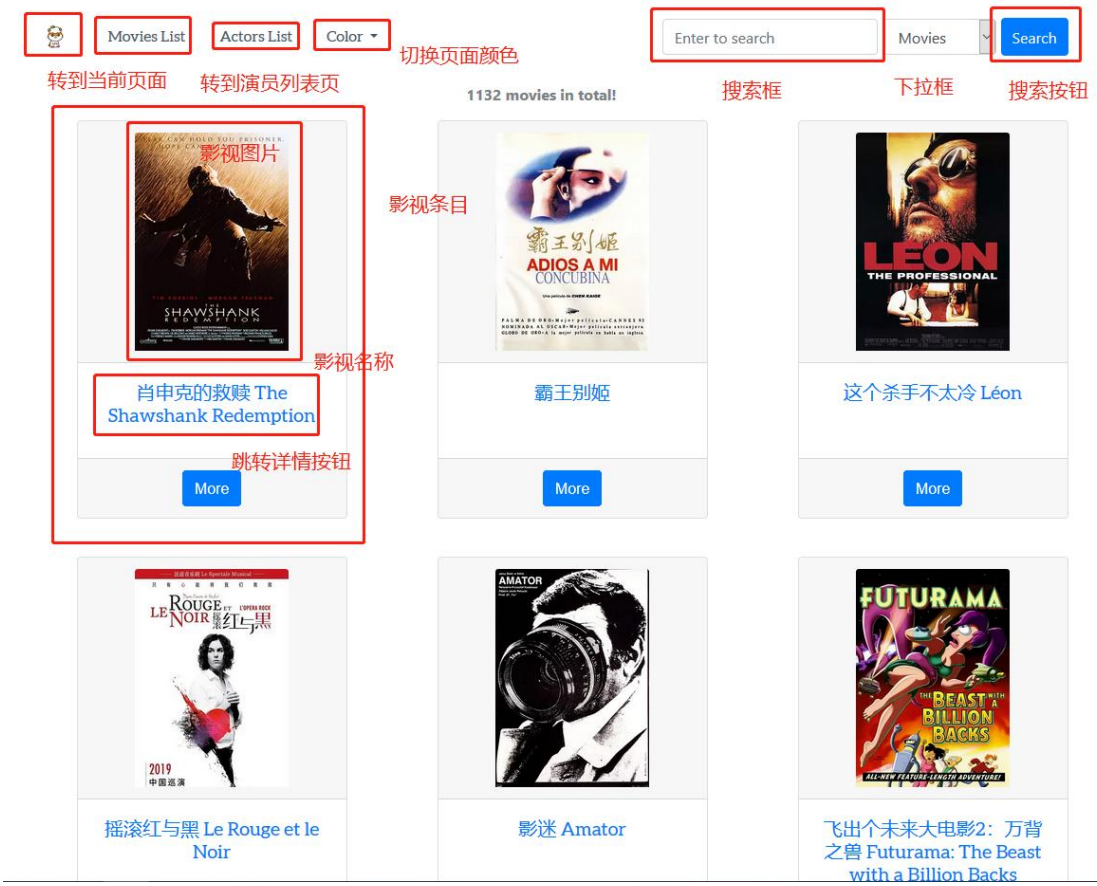


1.2 网站功能展示

网站分为五个页面：影视列表页面，演员列表页面，影视详情页面，演员详情页面，搜索结果页。下面逐一介绍这些页面。

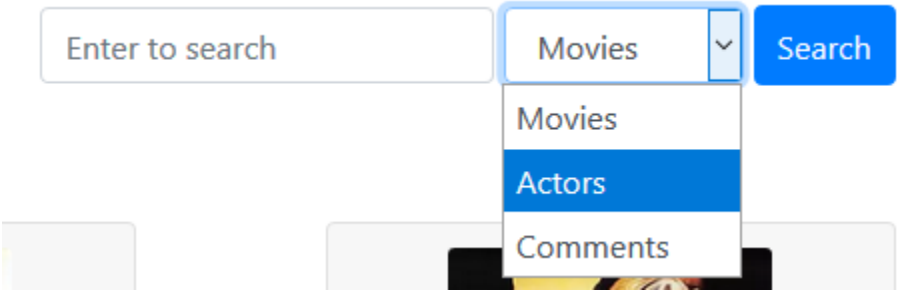
1.2.1 影视列表页面

影视列表页面是网站的主页，访问网站则自动进入影视列表页。影视列表页列举了系统中所有的影视条目。影视列表页的界面如下：



页面上方是导航栏。本网站的所有页面都具有同样的导航栏。下面简单介绍一下导航栏的内容以及功能。

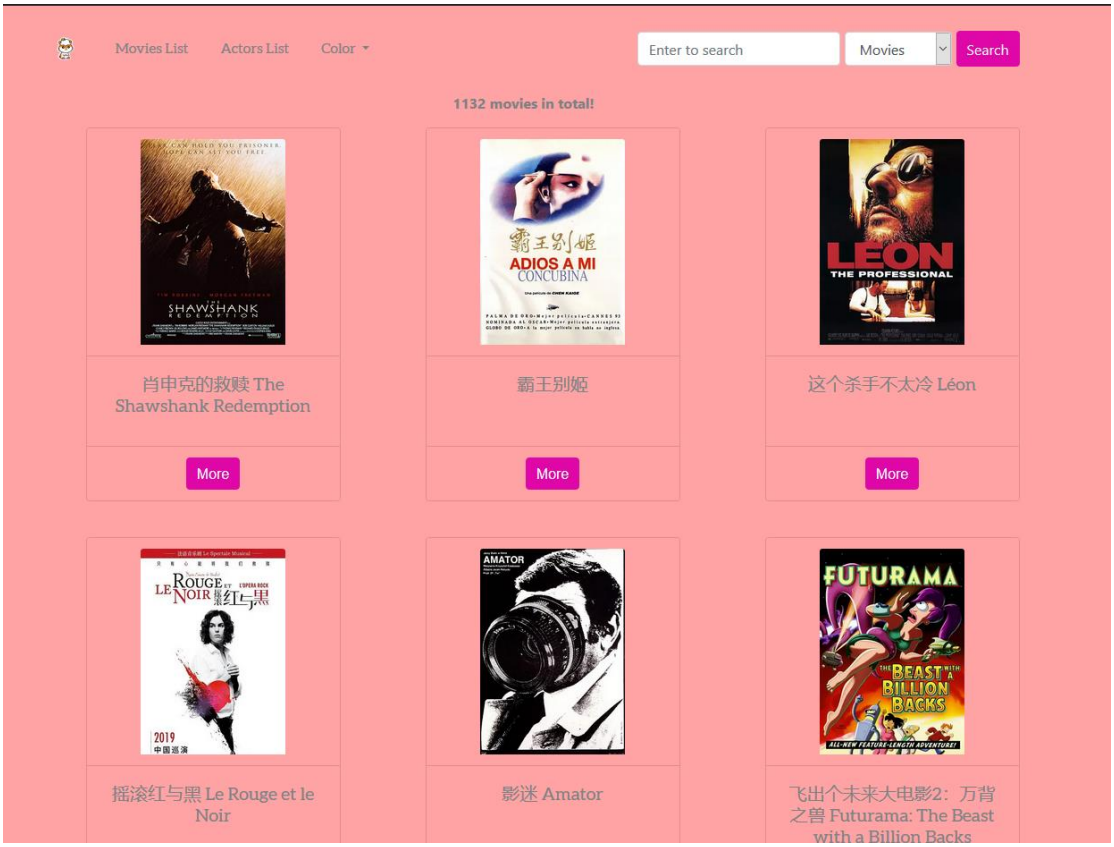
导航栏最左边是一个可爱的小猫图片，这个图片除了起到美化的效果外还起到了跳转主页的效果。在任何一个页面点击该图片，都会跳转至主页（影视列表页，也就是本页面）。左边第二个"Movies List"也是跳转主页的链接，点击该链接可以跳转至主页（影视列表页，也就是本页面）。左边第三个"Actors List"是跳转演员列表页的链接，点击该链接可以跳转至演员列表页。左边第四个"Color"下拉框可以切换当前页面的背景色。可以切换为红、白、蓝、黄等颜色。白色的效果就是上图的效果。左边第五个是搜索控件，搜索框可以输入长度小于等于 20 的文本，下拉框可以选择搜索的种类（可以选择在影视、演员或影评中检索），点击搜索按钮后可以执行搜索（输入完后回车也可以），进而跳转至搜索结果页面（搜索结果页面下面介绍）。搜索控件如下图。



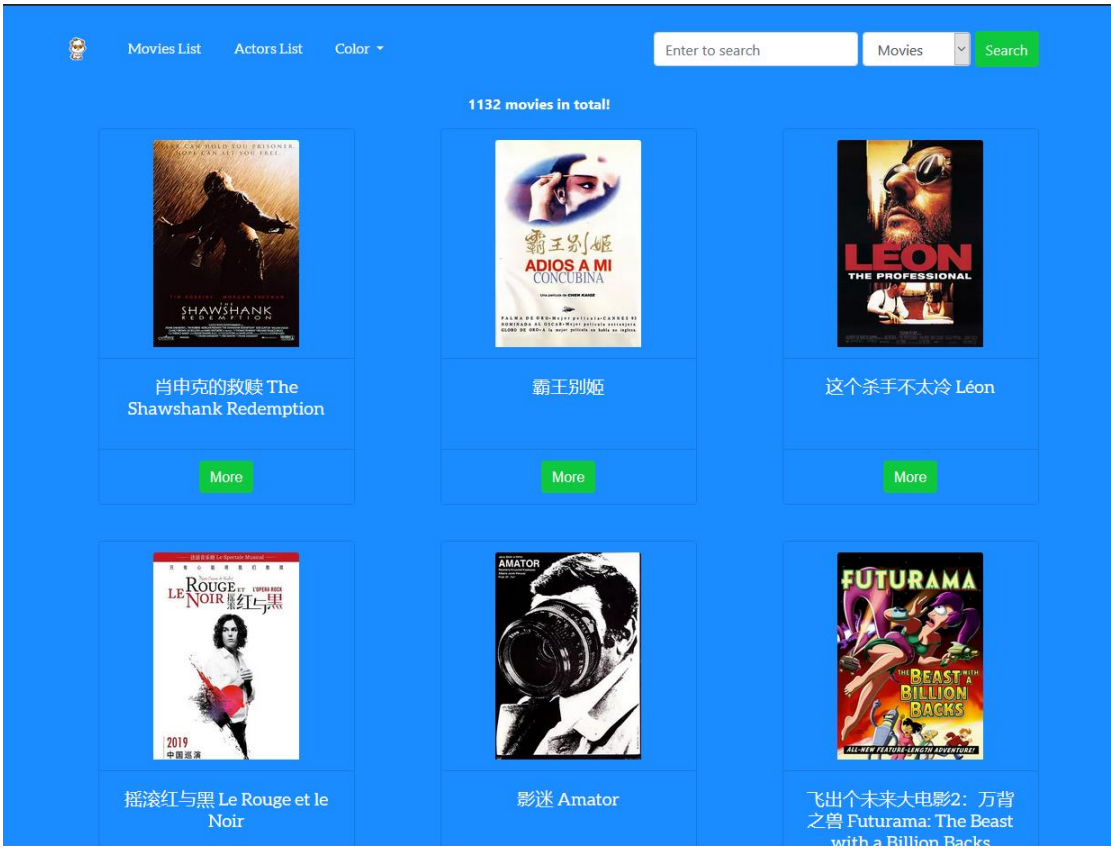
当搜索框为空时，若下拉框的选项为 Movies（默认状态）或 Comments，发起搜索，则会跳转至影视列表页面，若下拉框的选项为 Actors，发起搜索，则会跳转至演员列表页面。

下面展示切换其他背景色的效果。

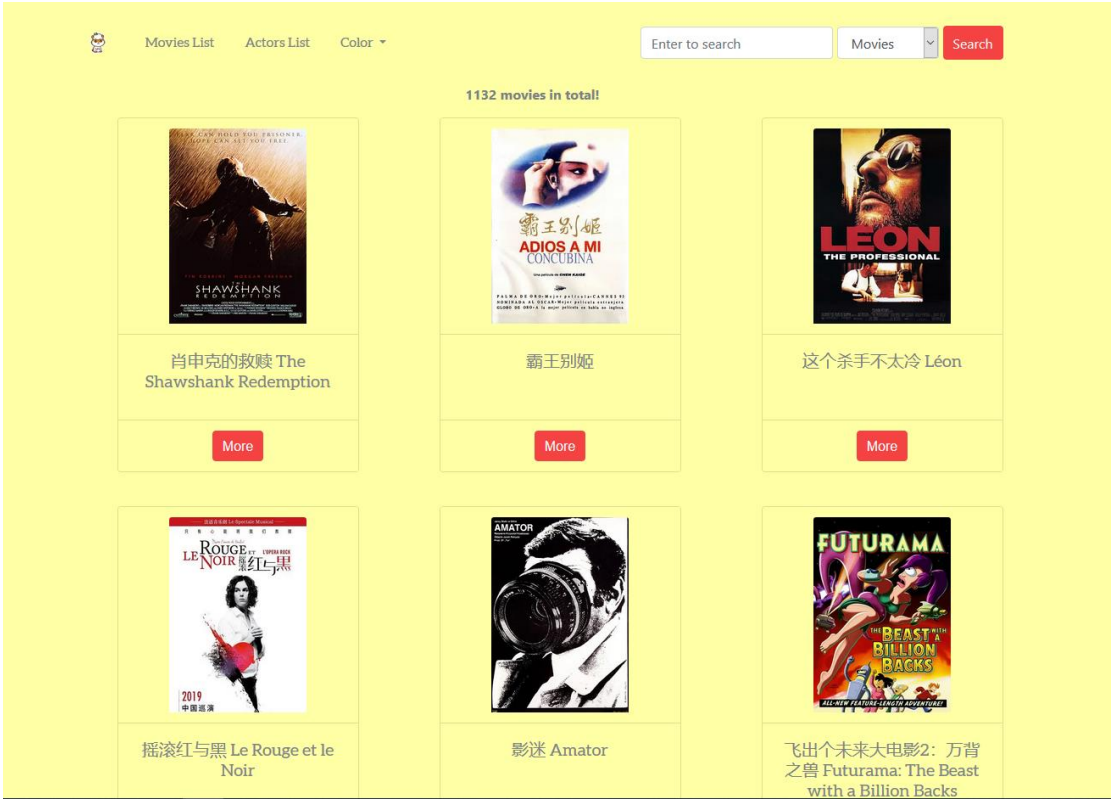
红色：



蓝色：

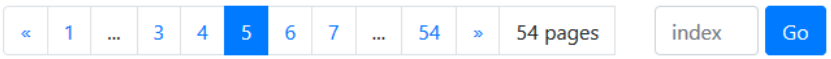

Movies List
Actors List
Color
Enter to search
Movies
Search
1132 movies in total!
肖申克的救赎 The Shawshank Redemption
More
霸王别姬
More
这个杀手不太冷 Léon
More
摇滚红与黑 Le Rouge et le Noir
影迷 Amator
飞出个未来大电影2：万背之兽 Futurama: The Beast with a Billion Backs

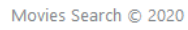
黄色：


Movies List
Actors List
Color
Enter to search
Movies
Search
1132 movies in total!
肖申克的救赎 The Shawshank Redemption
More
霸王别姬
More
这个杀手不太冷 Léon
More
摇滚红与黑 Le Rouge et le Noir
影迷 Amator
飞出个未来大电影2：万背之兽 Futurama: The Beast with a Billion Backs

导航栏中央下方显示一共有 1132 部电影。页面主体是电影条目，以卡片的形式展现。一页最多有 21 个电影卡片。每个卡片中展示了电影的名称，图片，和一个“More”按钮，点击三者的任何一个都可以跳转至对应影视详情页面。

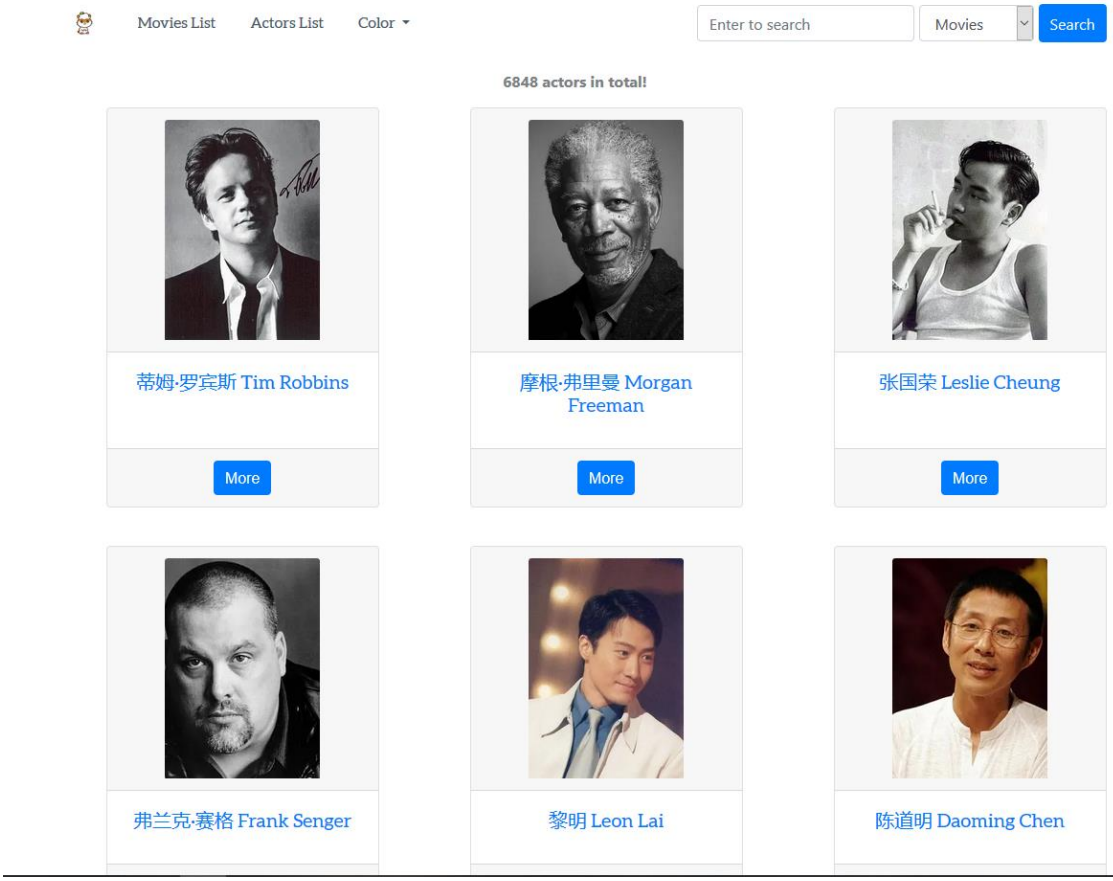
页面下方（如下图）为分页控件。分页控件包含<<和>>以及页码按钮。点击<<可以跳转至上一页（如果有的话），点击>>可以跳转至下一页（如果有的话）。页码展示了当前页前后共五页和首页、尾页的页码，点击页码可以跳转至相应页，其余页码均以省略号代替。右侧展示了共有几页，以及跳转页面的输入框及按钮。在输入框中输入[1,页面数]的整数，并按按钮或者回车便可以跳转至相应页面，如果输入不合法（不是整数或者页码不在合理范围内）前台会出现提示对话框提示用户。页面底部是一个 footer 标签。





1.2.2 演员列表页面

演员列表页面列举了系统中所有的演员条目。演员列表页的界面如下：



演员列表页上方的导航栏和下方的分页控件与主页的完全一致。此外，影视详情页面，演员详情页面，搜索结果页都具有同样的导航栏，搜索结果页也具有同样的分页控件。不再赘述。

主体的演员卡牌与主页的影视卡片也十分类似。一页最多有 21 个演员卡片。每个卡片中展示了演员的名称，图片，和一个“More”按钮，点击三者的任何一个都可以跳转至对应演员详情页面。

1.2.3 影视详情页面

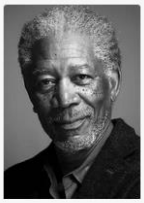
影视详情页面列举了对应影视的详细信息。影视详情页面的界面如下：


Movies List
Actors List
Color
Enter to search
Movies
Search
肖申克的救赎 The Shawshank Redemption
FEAR CAN HOLD YOU PRISONER. HOPE CAN SET YOU FREE.
THE SHAWSHANK REDEMPTION
state:美国 language:美国 length:142分钟
Introduction:
20世纪40年代末，小有成就的青年银行家安迪（蒂姆·罗宾斯 Tim Robbins 饰）因涉嫌杀害妻子及她的情人而锒铛入狱。在这座名为鳖堡的监狱内，希望似乎虚无缥缈，终身监禁的惩罚无疑注定了安迪接下来灰暗绝望的人生。未过多久，安迪尝试接近囚犯中颇有声望的瑞德（摩根·弗里曼 Morgan Freeman 饰），请求对方帮自己搞来小锤子。以此为契机，二人逐渐熟稔，安迪也仿佛在鱼龙混杂、罪恶横生、黑白混淆的牢狱中找到属于自己的求生之道。他利用自身的专业知识，帮助监狱管理层逃税、洗黑钱，同时凭借与瑞德的交往在犯人中间也渐渐受到礼遇。表面看来，他已如瑞德那样对那堵高墙从憎恨转变为处之泰然，但是对自由的渴望仍促使他朝着心中的希望和目标前进。而关于其罪行的真相，似乎更使这一切朝前推进了一步..... 本片根据著名作家斯蒂芬·金（Stephen Edwin King）的原著改编。
该页面展示了影视的名称，照片，国家，语言，长度，简介，以及五条短评（见下图）。
Comments:
Fear Can Hold You Prisoner, Hope Can Set You Free
没有人会不喜欢吧！书和电影都好。
点在哪？浅白的励志片诶
Hope is a good thing, and maybe the best thing of all.
一部没有女主，没有爱情，没有特技的好片子，也是很多大学英语老师喜欢放给学生看的片子，这部片子讲诉的就是一个很纯粹的故事，一次的救赎，换回了自由，也是一种信念驱使他这么做，这部片子是感人的，它也让很多人成长。
短评下方展示了该影视的演员名单(如下图)，每个演员条目也是以卡片的形式展现的。
每个卡片中展示了演员的名称，图片，和一个“More”按钮，点击三者的任何一个都可以跳转至对应演员详情页面。

Actors:

蒂姆·罗宾斯 Tim Robbins

More



摩根·弗里曼 Morgan Freeman

More

鲍勃·冈顿 Bob Gunton

More

威廉姆·赛德勒 William Sadler

More

克兰西·布朗 Clancy Brown

More

吉尔·贝罗斯 Gil Bellows

More

马克·罗斯顿 Mark Rolston

More

詹姆斯·惠特摩 James Whitmore

More

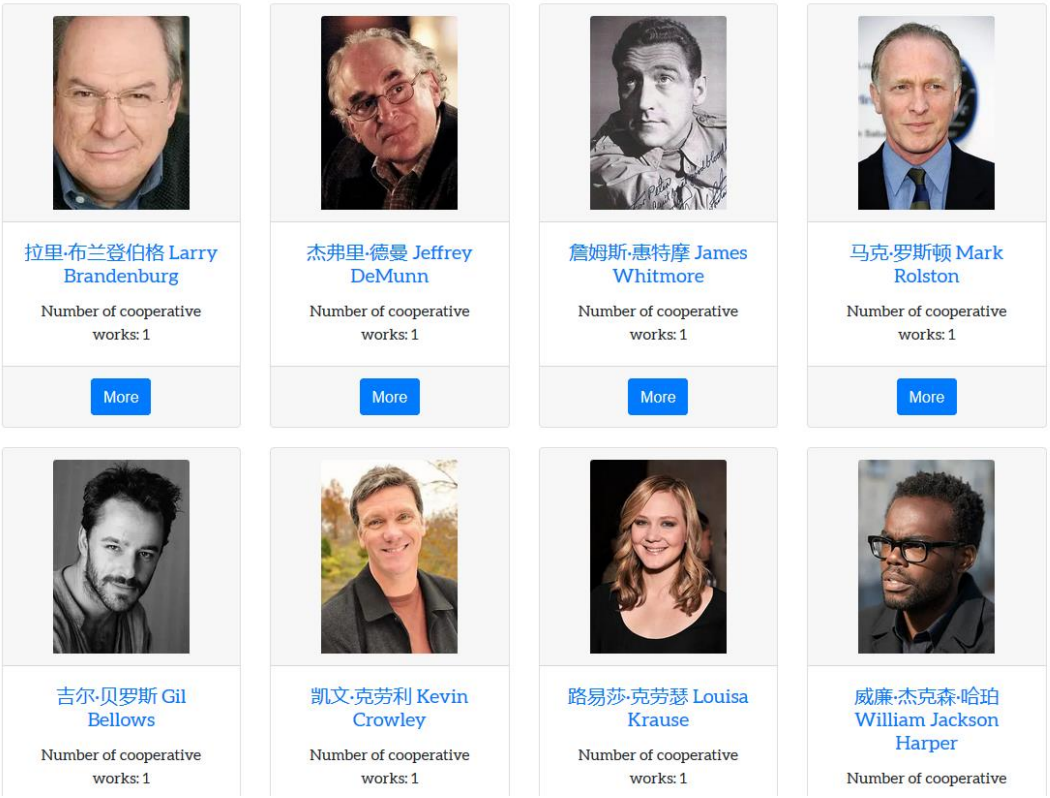
1.2.4 演员详情页面

演员详情页面列举了对应演员的详细信息。演员详情页面的界面如下：

该页面展示了演员的名称，照片，性别，星座，生日，出生地，简介。

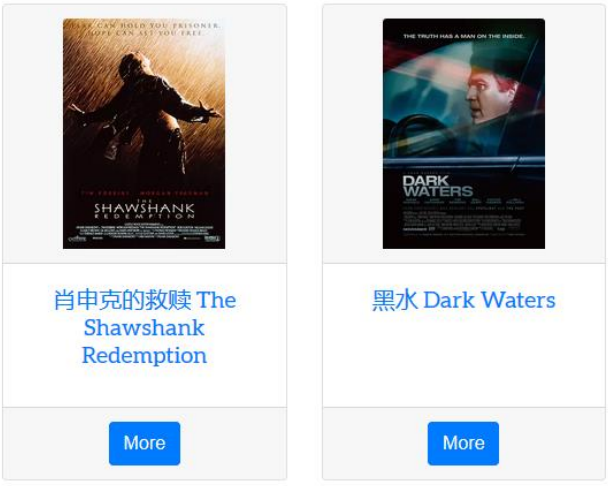
简介下方是合作演员。合作演员展示了该演员合作最多的 10 位演员的名单 (如下图)，每个演员条目也是以卡片的形式展现的。每个卡片中展示了演员的名称，图片，和一个“More”按钮，点击三者的任何一个都可以跳转至对应演员详情页面。此外，每个卡片还展示了卡片对应的演员与该演员合作的作品的数量。

Co-Actors:



合作演员下方是电影作品。电影作品展示了该演员的参与主演的所有电影（如下图），每个电影条目也是以卡片的形式展现的。每个卡片中展示了电影的名称，图片，和一个“More”按钮，点击三者的任何一个都可以跳转至对应电影详情页面。

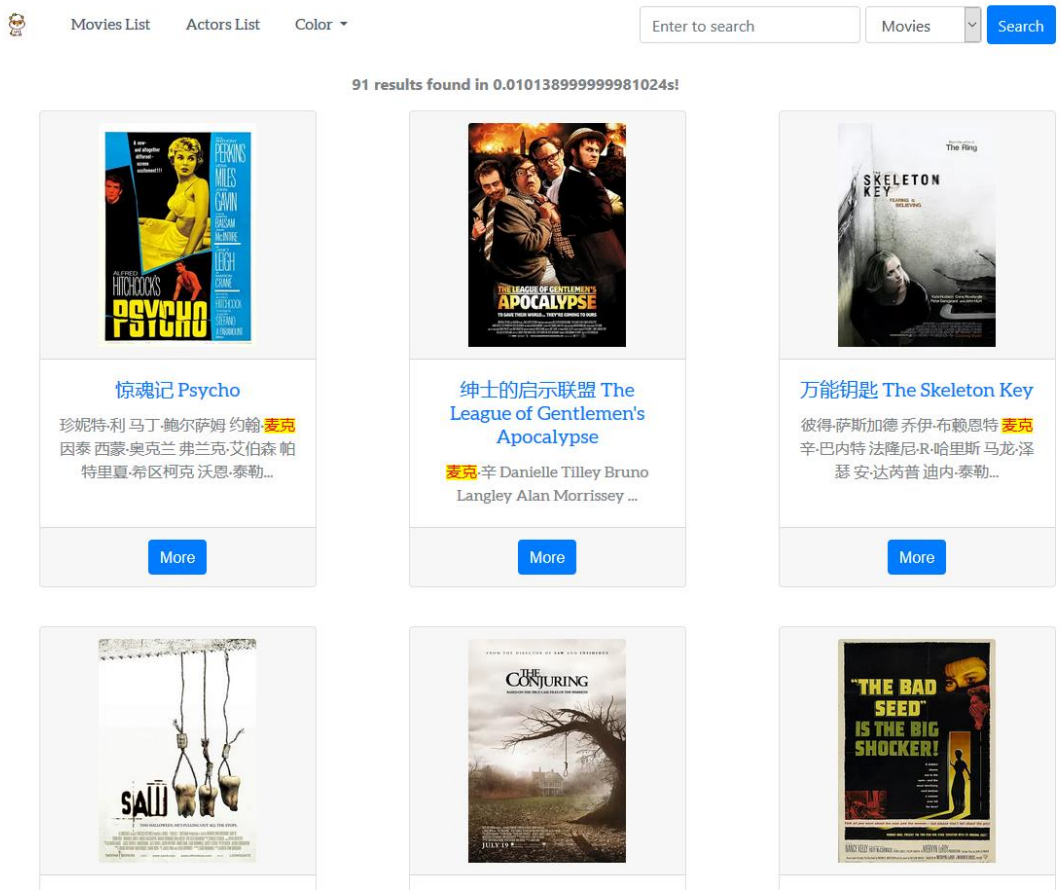
Film Works:



1.2.5 搜索结果页面

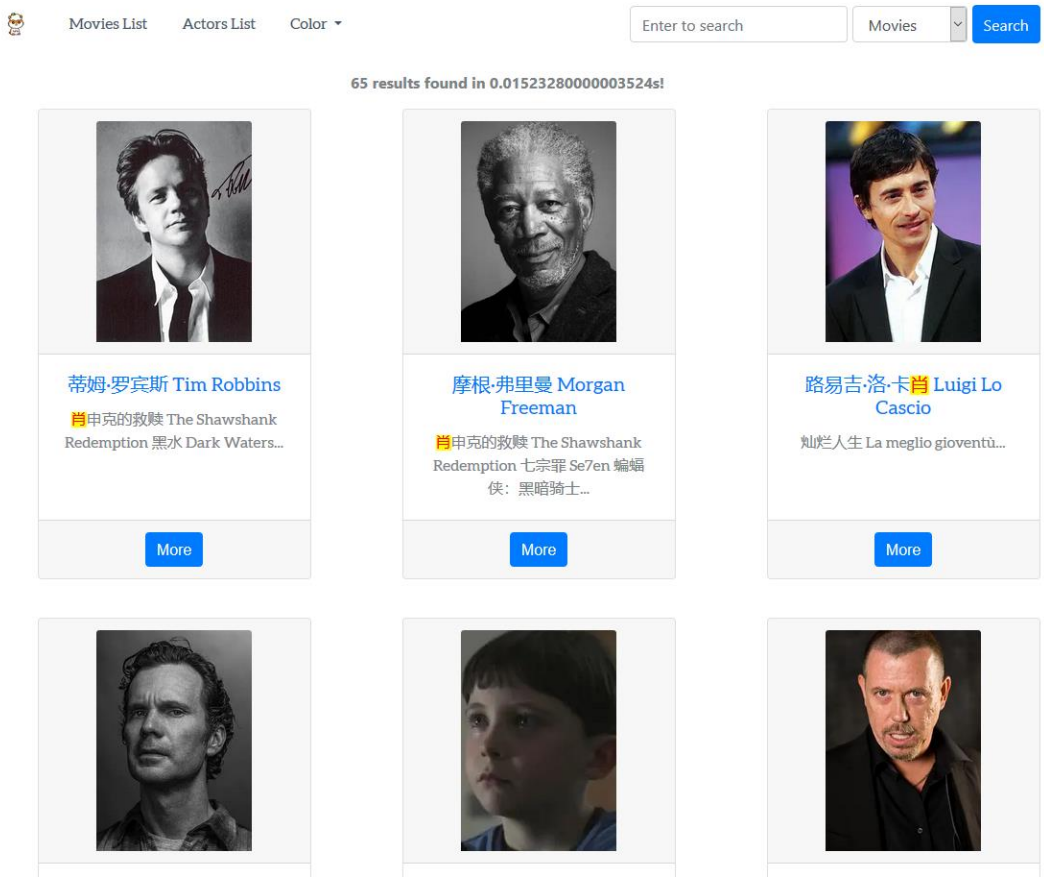
当在搜索框中填写了关键词文本，在下拉框中选择一个范围并且发起搜索后，页面跳转至搜索结果页。特别地，搜索结果页会以黄底红字的方式高亮搜索结果中的关键词。

若下拉框选择了 Movies，则系统会搜索所有名称或参演人员名单中包含该关键词的电影，同时以卡片列表的形式展示给用户。每个卡片中展示了电影的名称，图片，和一个“More”按钮。卡片中还展示了电影的参演名单，如果参演名单过长，则会截取包含关键词的一段名单（如果名单不包含关键词且只有名称包含关键词，则截取开头一段）。点击四者的任何一个都可以跳转至对应电影详情页面。搜索 Movies 的效果如下图（关键词为“麦克”）。

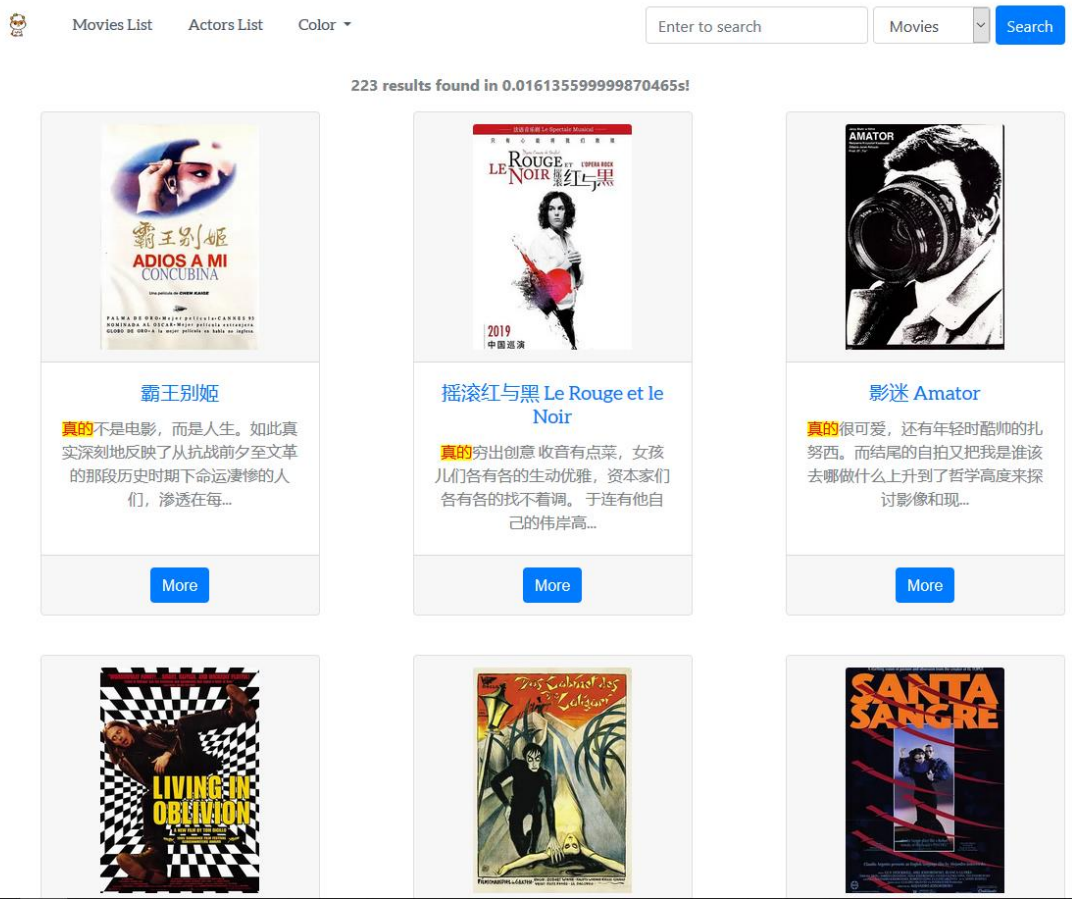


若下拉框选择了 Actors，则系统会搜索所有名称或影视作品名单中包含该关键词的演员，同时以卡片列表的形式展示给用户。每个卡片中展示了演员的名称，图片，和一个“More”按钮。卡片中还展示了演员的影视作品名单，如果影视作品名单过长，则会截取包含关键词

的一段名单（如果名单不包含关键词只有名称包含关键词，则截取开头一段）。点击四者的任何一个都可以跳转至对应电影详情页面。搜索 Actors 的效果如下图（关键词为“肖”）。



若下拉框选择了 Comments，则系统会搜索所有影评文本中包含该关键词的电影，同时以卡片列表的形式展示给用户。每个卡片中展示了电影的名称，图片，和一个“More”按钮。卡片中还展示了电影的影评文本，如果影评文本过长，则会截取包含关键词的一段文本。点击四者的任何一个都可以跳转至对应电影详情页面。搜索 Comments 的效果如下图（关键词为“真的”）。



2. 有关数据和性能介绍

我通过 python 爬虫共爬取了 1132 部豆瓣电影的信息，6848 位演员的信息。其中 1132 部电影的信息是全部存在的，而演员中只有 6130 位演员的信息是从有效链接中获取的，其他的演员都来自是缺失内容或无效的连接。因此对于剩下的 718 位演员，在数据库中只建立了一个条目，包含名字和合作演员，其他信息暂时为空（豆瓣中本身就没有该演员的详细信息）。

网站的搜索性能还是不错的，通过优化后，在 Movies 中搜索的平均用时大致为 9ms 左右，在 Actors 中搜索的平均用时大致为 12ms 左右，在 Comments 中搜索的平均用时大致为 14ms 左右。Movies 用时最短，因为影视的数量比较少，而且搜索涉及的文本比较少；Actors 用时较长，可能是因为演员的数量更多；Comments 用时最长，可能是因为 Comments

的文本长度比较长，搜索涉及的文本长度最长。

搜索范围\关键词	麦克	觉得	不错	幸运	也许	都市	平均结果
Movies	12ms	8ms	9ms	8ms	9ms	8ms	9ms
Actors	16ms	8ms	8ms	8ms	15ms	15ms	12ms
Comments	13ms	19ms	15ms	13ms	13ms	12ms	14ms

运行环境：

Windows 10 专业版(x64)

处理器:Intel(R) Core™ i7-4720HQ CPU @ 2.60GHz

RAM:16.0GB

以上搜索性能统计结果均是利用长度相同的关键文本进行平行查询 6 次后统计的平均数据。

3. 实现方法介绍

3.1 爬虫及数据库准备

首先进入豆瓣电影官网，查看电影分类排行榜（剧情片，喜剧片，恐怖片），并将网页 html 文件保存至本地。接着在 get_movie_list.py 中利用正则表达式解析保存到本地的 html 文件，从中读取电影的名称及其链接，并保存为 excel 文件。

接着运行 get_movie_contents.py。在该文件中，利用了 selenium 库模拟浏览器向豆瓣电影发起请求。为什么一定要 selenium 模拟浏览器行为呢，有两个原因，第一：豆瓣反爬机制比较厉害，需要模拟浏览器行为提供更加完善的头部信息；第二：豆瓣电影中很多内容都是 js 异步加载的，直接用 urllib 请求到的是静态网页源代码而不是完整的源代码，比较方便的方法就是直接模拟浏览器去触发 js 事件，获取完整的源代码。这里 selenium 利用的 Chrome 浏览器，也是一个通用性比较强的浏览器。再从之前保存的 excel 文件中获取各个电影的链接，接着利用 selenium 访问该链接，并获取电影主页的源代码。再利用正则表达

式从源代码中获取我们需要的电影信息，其中就包括后面需要用到的参演演员的姓名和链接。最后将有关演员的名称和链接存为 csv 文件，将电影的相关信息（名称、简介、图片链接、短评、演员列表等）存为 json 文件。值得注意是，即使用了 selenium 库，豆瓣仍然能识别出爬虫，因此这里还采用了动态 ip 代理。我个人购买了网上的高速动态 ip 代理服务，并且利用软件的接口动态更换 ip，每 20 分钟左右更换一次 ip 代理。采取这样的策略后我就轻轻松松地完成了电影和演员的爬取工作。

在爬取完电影信息和演员的链接后，运行 get_actor_contents.py。在该文件中，读取了之前存在 csv 中的演员名称和链接，采用与爬取电影一样的方法根据这些链接爬取演员。将爬取到的演员的信息存为 json 文件，同时会忽略无效链接（有些演员的链接指向的是错误的页面，或者豆瓣根本就没有该演员的信息）。至此，我们已经以 json 文件的形式存取了电影和演员的基本信息（不含那些需要进一步计算的数据，如与该演员合作次数最多的 10 名演员）。另外，从网上爬取的演员和电影存在重复的情况，因此这里在爬取完后运行了 check_number.py 对爬取到的电影和演员 json 文件进行去重操作（详情页链接一致才认为电影或演员重复，这是为了防止同名电影或者同名演员被误删的情况）。

最后是数据库的录入。首先运行 web 项目目录下的 load_actor.py。将演员 json 文件中的内容录入数据库，具体是录入了每个演员的名字、简介、图片链接、性别、星座、生日、出生地、该演员的详情页链接（这里的链接指的是豆瓣该演员详情页的链接，而非本网站的链接，在录入数据时作为该演员的唯一 id 标识，下同）。接着运行 load_movie.py 将电影信息录入数据库中。具体而言，首先录入该电影的名字、简介、图片链接、国家、语言和时长。接着将该电影的所有短评组合成一个字符串储存入数据库（以"\n"连接），将该电影的所有演员名单组合成一个字符串储存入数据库（以" "连接，为了方便后续的检索，字段名为 actors_name_str）。最后利用 django 的 ManyToMany 键建立电影和演员关系。遍历数据库

的演员，如果某个演员的链接与电影 json 文件存储的演员链接一致（这么做是为了防止同名演员被误录入的情况），则认为该演员参演了这部电影，于是设置外键。如果遍历之后没有找到该演员（说明这个演员的链接是无效的，爬虫没能爬下来这个演员的信息），那么就创建一个新的演员条目，将其名字和链接分别赋值为电影 json 文件中相应的演员名和演员链接（相当于建立了一个只有名字的演员条目），接着设置电影与新演员的外键。至此，就基本完成了电影信息的存储。最后运行 complete_actor.py。这个文件完成了两个工作，第一：遍历每个演员，通过外键找到其参演的电影，该演员的所有作品的名称组合成一个字符串储存入数据库（以" "连接，为了方便后续的检索，字段名为 movies_name_str）；第二：计算出与每个演员合作次数最多的前十员演员，并将其名字以及合作次数存入数据库。至此，数据库已经准备完毕。

3.2 前端

本网站共有五个页面，分别是影视列表页面，演员列表页面，影视详情页面，演员详情页面，搜索结果页。其中影视列表页面，演员列表页面和搜索结果页都属于列表页，只是展示的具体内容不一样而已，因此这三个页面采取同一个模板 result.html。其他两个页面都属于详情页，只是展示的具体内容不一样而已，因此它们也采取同一个模板 detail.html。两个模板都用到了 bootstrap 来封装界面，bootstrap 的页面美化效果非常的优秀。

下面介绍列表模板 result.html 的前端实现。

首先导航栏部分使用 bootstrap 的 navbar 模块，链接使用了 nav-link 包装，下拉框使用了 dropdown 进行包装。搜索框和按钮则使用了 bootstrap 的 form-control 和 btn-primary 进行封装。在 bootstrap 的美化效果之上我还加入了一些自己的 css 样式进行进一步的调整优化。

列表部分采用 django 的模板进行 for 渲染，如果后端提供的是影视列表，那么就渲染影视列表，如果后端提供的是演员列表，那么就渲染演员列表，如果后端提供的是搜索结果，那么就渲染搜索结果。具体到每个条目，这里是用 bootstrap 的 card 模块去做类似卡片的效果的，条目之间用 flex 进行排列，并且用是 bootstrap 的 col-sm-6 和 col-lg-4 进行布局管理。

分页控件部分采用了 bootstrap 的 Page navigation 进行包装。前端根据后端提供的页面信息进行相应的渲染。至于后端如何分页后端部分会详细介绍。

底部的页脚采用 bootstrap 的 footer-basic 进行包装。

至于页面的一些功能：变换背景色直接通过 js 脚本改变对应控件的颜色实现的；高亮显示关键词则是通过搜索页面中的关键词并修改相应位置的 html 源代码，加入使其高亮实现的；页面跳转则是在回车或者按下按钮后，通过 js 检查用户的输入是否合法，如果合法直接调用 js 的 document.location.href 进行页面跳转，如果不合法则调用 alert 对用户进行提示。

下面介绍列表模板 detail.html 的前端实现。

导航栏和页脚部分的实现与 result.html 一致。不同的在于主体页面。主体页面根据后端传来的是影视还是演员信息用模板进行相应的渲染。渲染方式就是直接利用<p>和<a>进行渲染，值得注意的是合作演员、参演演员和作品的渲染，这部分的渲染也是采用卡片的形式，与 result.html 很类似，不同的是这里用的是 col-sm-6 和 col-lg-3 而不是 col-sm-6 和 col-lg-4，目的是让卡片更加小、排列得更紧密，毕竟这只是页面主体的一部分。

至于页面的变色功能的实现与 result.html 一致。

3.3 后端

下面介绍后端各个 api 的实现。

1、 index:

该 api 是访问主页（也就是影视列表页）时调用的 api，该 api 直接将从数据库中取出所有的影视数据，分页后发送给前端。分页这里是用 django 的 Paginator 实现的，可以便捷地将多条数据分成多页，每页 21 条数据。所有列表页的分页都采用了 Paginator，下面不再赘述。

2、 actor

该 api 是访问演员列表页时调用的 api，该 api 直接将从数据库中取出所有的演员数据，分页后发送给前端。

3、 find_result

该 api 是在提交搜索框表单后调用的 api。该 api 会提取 request 中的关键词和搜索类型，并将这些数据传递给 show_result()。如果关键词为空，则跳转至相应的列表页。

4、 show_result

该 api 接受 find_result()传入的关键词文本和搜索范围。首先对关键词文本进行首尾去空格。判断搜索范围。

若搜索范围为 Movies，则调用 django 的 fileter 方法，利用 Q 表达式查找 name 或 actors_name_str 中含有关键词的影视，接着将该影视的 actors_name_str 中包含关键词的部分截断出来并将其有关信息加入结果中。

若搜索范围为 Comments，则调用 django 的 fileter 方法，利用 Q 表达式查找 comments 中含有关键词的影视，接着将该影视的 comments 中包含关键词的部分

截断出来并将其有关信息加入结果中。

若搜索范围为 Actors，则调用 django 的 fileter 方法，利用 Q 表达式查找 name 或 movies_name_str 中含有关键词的演员，接着将该演员的 movies_name_str 中包含关键词的部分截断出来并将其有关信息加入结果中。

在获得相应搜索结果后进行分页，将有关数据发送给前端。前端需要的查询计时采用 python 标准库的 time 模块进行计时。

5、 show_movie_detail

该 api 根据传入的 id 在数据库中寻找相应的影视。如果找不到，则直接返回 HttpResponse('找不到该影片的信息')。否则从数据库中提取对应 id 的电影信息，分页后发送给前端。

6、 show_actor_detail

该 api 根据传入的 id 在数据库中寻找相应的演员。如果找不到，则直接返回 HttpResponse('找不到该演员的信息')。否则从数据库中提取对应 id 的演员信息，分页后发送给前端。

4. 总结

此次作业充分利用所学知识和互联网上的资源，成功地爬取了足够的影视和演员信息，搭建了属于自己的小小网站。总的来说，网站界面简洁用户友好，代码编写有一定的结构性和可读性，各项功能实现得也比较完备。特别地，本程序在要求的基础上利用动态 ip 代理提高爬虫的鲁棒性，利用 bootstrap 和 css 对界面进行了较大力度的美化，利用 js 实现了关键词高亮和变更背景色的功能，对查询部分进行了优化，获得了比较好的查询性能。经过测试，本程序的功能基本符合预期，没有出现不符合预期的情况。