```
ient() {
        // ES服务器URL
        String serverUrl = "http://127.0.0.1:9200";
        // ES用户名和密码
        String userName = "xxx";
        String password = "xxx";

        BasicCredentialsProvider credsProv = new
BasicCredentialsProvider();
        credsProv.setCredentials(
                AuthScope.ANY, new
UsernamePasswordCredentials(userName, password)
        );

        RestClient restClient = RestClient
                .builder(HttpHost.create(serverUrl))
                .setHttpClientConfigCallback(hc ->
hc.setDefaultCredentialsProvider(credsProv))
                .build();

        ElasticsearchTransport transport = new
RestClientTransport(restClient, new JacksonJsonpMapper());
        return new ElasticsearchClient(transport);
    }

}
```

索引操作

代码中的 esClient 就是 ElasticsearchClient，请自行注入 bean

```
// 索引名字
String indexName = "student";

// 索引是否存在
BooleanResponse books = esClient.indices().exists(e ->
e.index(indexName));
System.out.println("索引是否存在：" + books.value());
```

```
// 创建索引
esClient.indices().create(c -> c
        .index(indexName)
        .mappings(mappings -> mappings  // 映射
                .properties("name", p -> p
                        .text(t -> t // text类型，index=false
                                .index(false)
                        )
                )
                .properties("age", p -> p
                        .long_(t -> t) // long类型
                )
        )
);

// 删除索引
esClient.indices().delete(d -> d.index(indexName));
```

文档操作 (CRUD)

下文以官方测试数据 [account. json](http://cxyroad.com/"https://github.com/elastic/elasticsearch/blob/7.5/docs/src/test/resources/accounts.json") 为例

实体类

首先定义实体类，用于 ES 中的字段

```
public class Account {
private String id;
// 解决ES中字段与实体类字段不一致的问题
@JsonProperty("account_number")
    private Long account_number;
    private String address;
    private Integer age;
    private Long balance;
    private String city;
    private String email;
```

```
    private String employer;
    private String firstname;
    private String lastname;
    private String gender;
    private String state;
... 省略get、set方法
}

```

新增

```
String indexName = ”account”;  // 索引名字
Account account = new Account();
account.setId(”1”);
account.setLastname(”guyu”);

// 新增
C vate String address;
    @Field(type = FieldType.Integer)
    private Integer age;
    @Field(type = FieldType.Long)
    private Long balance;
    @Field(type = FieldType.Text)
    private String city;
    @Field(type = FieldType.Text)
    private String email;
    @Field(type = FieldType.Text)
    private String employer;
    @Field(type = FieldType.Text)
    private String firstname;
    @Field(type = FieldType.Text)
    private String lastname;
    @Field(type = FieldType.Text)
    private String gender;
    @Field(type = FieldType.Text)
    private String state;
... 省略get、set 方法
}

```

```
IndexOperations idxOpt = template.indexOps(Account.class);
```

```
// 索引是否存在
boolean idxExist = idxOpt.exists();

// 创建索引
boolean createSuccess = idxOpt.createWithMapping();
System.out.println(createSuccess);

// 删除索引
boolean deleted = idxOpt.delete();
```

文档操作（CRUD）

```
Account account = new Account();
account.setId(”1”);
account.setLastname(”guyu”);

// 这是插入或覆盖，如果id存在了就是覆盖
template.save(account);

// 修改，用的是es的_update
template.update(account);

// 删除
template.delete(account)

// 批量新增(用的是es的_bulk)
List<Account> accountList = ...
template.save(accountList);

// 根据id查询
Account account = template.get(”1”, Account.class);
```

搜索 + 排序 + 分页

```
// 搜索 firstname = Amber AND age = 32
Criteria criteria = new Criteria();
criteria.and(new Criteria(”firstname”).is(”Amber”));
```

```
criteria.and(new Criteria("age").is(32));

// 分页
int pageNum = 1; // 页码
int pageSize = 20; // 每页数量
Query query = new CriteriaQueryBuilder(criteria)
        .withSort(Sort.by(new Order(Sort.Direction.ASC, "age"))) // 排序字段1
        .withSort(Sort.by(new Order(Sort.Direction.DESC, "balance"))) // 排序字段1
        .withPageable(PageRequest.of(pageNum - 1, pageSize)) // 浅分页
        // 不需要查询的字段
        .withSourceFilter(new FetchSourceFilterBuilder().withExcludes("email", "address").build())
        .build();

SearchHits<Account> searchHits = template.search(query, Account.class);
long totalValue = searchHits.getTotalHits(); // 匹配到的数量
for (SearchHit<Account> searchHit : searchHits.getSearchHits()) {
    Account account = searchHit.getContent(); // 这就是得到的实体类
}

```

总结
==

本文介绍了 SpringBoot 整合 ElasticSearch 的两种方案，但均只是简单提及，更详细的用法需要自行查看官方文档。