Internship Log/Notes

Internship period: August 9 - August 22, 2021

Internship location: Zhejiang FanhaiZhixingDianli Technology Co., Ltd

Internship area: Java Backend Development

Intern name: Haoquan Zhou

Day 1

JDK Build

The JDK is the basic compiler for Java, in a sense, it is equivalent to the g++/gcc tool for C++/C. We just download the tool from the website and install it. Using this tool, we can then use code editors like VS Code to write the code and then compile the code, using the Windows Shell to run the compiled code.

IDEA Build

IDEA is just an integrated building tool for Java, like the Eclipse. It is useful for big project development. The basic procedures for writing a code is like:

1. Use `new` to create a project and give it a name.
2. Use `new` to create a module in the `src` directory.
3. Use `new` to create a java class, then we can start to build a class

GitLab Access

Here we use a tool SmartGit to access the GitLab to download the resources in the Git server. The SmartGit is actually a multi-system capable command line editor. Here we use the bash (Linux-based) one, I really appreciate that point. We also have cmd (Windows) one.

IDEA Running

It is unfortunate that we try to load and run a pre programmed project in IDEA, but we failed to run it. We have tried three different versions of IDEA, but they all ended in the same error.

Day 2

First Java Program

I have developed my first Java program using IDEA. It is just a single-ended-singly-linked list. I start the project at Day 1 and finished in Day 2. Also, I tested the project and verify that all the member functions are working properly. I have uploaded the project to GitHub, the link is shown below

https://github.com/TaikiShuttle/single-ended-singly-linked-list-in-Java.

The Environment of IDEA

It turns out that the free version of the IDEA does not support the running tool "spring boot". So I re-downloaded the IDEA and set up the spring boot. The configuration status is fine, but the problem is still there.

Day 3

Fix the Problem of IDEA

I finally fixed the compiler problem of IDEA. The point is that the project I am using is built under the circumstance of JDK 1.8, while my JDK version is 16.0.2, which is the newest version. It turned out that the version will actually infect the compiler since some features have been deleted through years. As a matter of fact, there is a feature in this program that is disabled since JDK 13. Then I used the JDK 1.8.0_241 version provided by the programmer to run the program. Also, I have put some effect in changing and setting the environment variable both in the computer and for the IDEA.

Further Error about the Limitation

After the program ran properly, I was told by the compiler that the database can not be accessed, this is because I, as a internship student, do not have the right to access the database. Maybe I can look at the programmer's computer to see how the programs looks like in Day 4. Or I will just skip this procedure and begin to check the basic structure of this project.

The Basic Structure of MVC Project

During the debugging process, I learned that this project is actually a project based on MySQL, which is a database program. Further more, the programmer told me that there is a concept **MVC** in this programming industry. The **M** means model, which indicates the **model of the business**; the **V** here means View, which indicates the **users' output interface**; the C here means controller, which indicates the **users' input interface**. These three parts work as follows:

1. The **controller** takes the input of the user and send it to the model of business.
2. The **model of business**, which usually has a database connected to it, will then do some jobs like storing the data or fetch the data. It will also judge which business does the user want to do.
3. Based on the output logic of model of business, the **view** will then provide the user with corresponding result
4. The user may do next step after receiving the result provided by the view, which enters the next loop

This idea is first raised by Xerox and then adopted by Java EE. More details can refer to

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

Based on this design idea, the project designed using Java has following four layers:

1. **Controller/View Layer** (Call the corresponding controller based on the URL input, namely, the input of the user/ Show the corresponding result to the user)
2. **Service Layer** (Implementation of ADT, like abstract classes)
3. **Dao Layer** (In charge of communicating with the database)
4. **Mapper Layer** (Do some continuous manipulations to the database)

Note that the Mapper Layer is the bottom layer and the Controller/View Layer is the top layer, which means that you must build Mapper Layer first, then Dao Layer, then Service Layer, finally the Controller/View Layer.

Day 4

The usage of `@` in Java / API Document / Override

The character `@` has some special functions in Java. It is not a operator, but is used in some Dao techniques :

1. Use `@` to comment the author and some related information, for example

   ```
   /**
   * @author ZhangSan
   * @version 1.1.0
   * @since 2021.8.12
   * @see
   */
   ```

   This block is used to automatically generate an **API document**. The API document shows us how to use the classes and functions provided in the code. For example the Java API provided by Oracle, it shows the function of the pre-defined classes, interfaces... The link is shown below

   java.applet (Java Platform SE 6) (oracle.com)

   Some websites like GitHub also have its own API, see

   GitHub Documentation

   More often, a programmer will write an API document for his/her code, so that it is easier to communicate. And for a complete system with numerous methods, it is necessary for the provider to give an API document to the customer to help them know how to use the program. It also obey the rule of sealing the code and only providing the interfaces. There is a reference of an API of a complete report form system

   https://github.com/TaikiShuttle/An-API-for-report-form-.

2. `@` as Annotation

   The `@` can also be used in some algorithms to serve special usage, for exapmle

   ```
   @ + keyword/value
   ```

   It will send the keyword/value to the compiler so that the complier's action can be accurately controlled, for example

   ```
   @Override
   public void OnCreate(Build state) {
       ...
   }
   ```

   will make sure that the following function indeed inherit a function in its superclass. If there is on such function in the superclass, it will give an error. For example

```
@Override
public void oncreate(Build state) {
    ...
}
```

will result in the following error

```
The method oncreate(Build) of type HelloWorld must override or implement a
supertype method
```

If we do not have this `@Override` , the compiler will assume that we create a new method `oncreate(Build)` in the subclass.

So the whole inheritance block should be

```
public class father {
    public void foo();
}

public class child extends father {
    @Override
    public void foo();
}
```

Actually, this particular `@Override` sentence is the same as the keyword `override` in C++, like

```
class father {
    public:
    virtual void foo();
};

class child: public father {
    public:
    void foo() override;
};
```

Note that in Java, the member functions **are virtual by default**, use `final` to stop overriding. While in C++, the member functions **are real by default**, use `virtual` to enable overriding.

A good habit is to add `@Override / override` whenever possible.

Go back to the topic of `@` , the algorithm `@ + keyword/value` also has other usages.

Use of `static`

In Java, a class may contain several member functions. Some of them are declared as `static` while the others are not. In the `static` ones, we cannot call a non-static function. For example, the `main` function is always `static` , so it cannot call any function that is non-static.

The usage of `static` is that it can be used by several classes. We can call a static member in this class or in other classes using the format

```
[class name].[static member name]
```

`impliments` vs. `extends`

Java extracts the concept of interface and make it a new type called `interface`, it behaves just like `class`, but it does not contain any implementation, just like the interface in C++. To inherit the interface and give further implementation, we use `implements` instead of `extends`. Note that we can implement several interfaces in one class, like

```
public class A implements B,C {
    ...
}
```

Or

```
public class A extends B implements C,D {
    ...
}
```

An important fact is that the implemented class **cannot override** the members in the base interface, it just provides an implementation for the base class.

Also, we need to know that `extends` can only inherit **one** class, but `implements` can inherit **several** interfaces.

Abstract Class vs. Interface

This part is written based on the blog in

https://blog.csdn.net/b271737818/article/details/3950245

Same Point

1. The same point is that they are all abstract class type, and they cannot be used to declare an instance.
2. The implementation class of the interface and the subclass of the abstract class must implement all the methods declared interface/abstract class.

Different Point

1. The interface requires `implements` while abstract class requires `extends`.
2. A class can implements several interfaces while a class can only extends one abstract class.
3. The implementation class for a interface **must implement all the methods** declared in the interface. The subclass of the abstract class may not implement all the methods.

   For those **non-abstract methods** declared in the abstract class, the implementation **must provided in the base class**. And the subclass may directly inherit them, or it can override them (recall that all the members in class are **virtual** in Java).

   For those **abstract methods** declared in the abstract class, the subclass either gives an implementation, or continues to declare it as abstract member to its subclass (in this case the subclass should also be declared as `abstract` and cannot be used to declare an instance).

   This is the same as that in C++. A class can be used to declare an instance if and only if all its methods have been implemented.

4. Usually, the interface does not contain constants and variables, while the abstract class may contain constants and variables. Recall that **having no data members** is the core spirit of interface. So we find that we indeed needs some data members, we can use abstract class to implement this.

   As concluded by the original author: "The abstract class serves as the bridge between interface and class. One the one hand, it declares the methods that the subclass must implement. On the other hand, it also provides some implementations of methods such that the subclass can directly inherit or override. Furthermore, the abstract class can also have some data members for subclass to inherit to use, if necessary."

HashMap in Java

This part is written based on blog in

https://blog.csdn.net/woshimaxiao1/article/details/83661464

What is a HashMap

The exists data structure all have their pros and cons. Array has time complexity `O(1)` in finding elements using index and `O(n)` in finding the elements using value and `O(n)` in inserting value. The sorted array may have time complexity `O(logn)` in finding elements. Linked list has `O(1)` in inserting value while `O(n)` in finding the elements using value. A balanced binary tree has all the methods with time complexity `O(logn)`. But HashMap has all the methods with time complexity `O(1)`.

The HashMap is based on an array, but it use a **key** with a Hash mapping function **f**

$$f : key \rightarrow address$$

to implement the searching method.

Note that if several keys are mapped to the same address, the later ones are inserted using linked list.

How the HashMap is Implemented

The basic array for a HashMap in Java is an `Entry` array, namely

```
transient Entry<K,V>[] table = (Entry<K,V>[]) EMPTY_TABLE;
```

while for every `Entry` element, it is a static class which is actually a **linked list node** `first`

```
static class Entry<K,V> implements Map.Entry<K,V> {
    final K key;
    V value;
    Entry<K,V> next;//A single-ended-singly-linked-list
    int hash;//the hashcode for key after Hash mapping f
             //Use space in exchange for time
    /**
     * Constructor, creates new entry.
     */
    Entry(int h, K k, V v, Entry<K,V> n) {
        value = v;
        next = n;
        key = k;
        hash = h;
    }
```

```
}
```

Then, for the inserting or finding methods, we can first use Hash mapping to find the address, if there is just one element, then we are done; else we traverse the linked list to find the element or insert the new element.

Apart from the `Entry<K,V> []` array, in the `HashMap` class, we also have some members which are important

```
transient int size; // to store the current number of key-value pairs
int threshold; //The capacity of the list, initially it is 16

//By default is 0.75, if 0.75 of the HashMap have been filled,
//the map will expand. This is aimed to prevent hash collision
final float loadfactor;

//If the element in HashMap is changed due to other operations
//we need to throw an exception
transient int modCount;
```

There are also lots of methods provided in the `HashMap` class, include

```
//The universal methods for containers
public void clear();
public boolean isEmpty();
public int size();

//Some special methods for maps, including HashMap

//If the table is empty, create a new table, else use the
//current table. Then use hash mapping method to insert.
//If there is an old value, return it.
public V put(K key, V value);

//Object is a pre-set super class for all the class
//This will return the value for key, null if we cannot find
public V get(Object key);

public V remove(Object key); //Almost the same as get, just add a remove
function
public boolean containskey(Object key);
public boolean containsValue(Object value);
public Set keySet(); //return a Set of keys
public Collection values(); //return a Collection of values
```

Day 5

Java Servlet

This part is written based on the blog in

https://blog.csdn.net/qq_19782019/article/details/80292110

The **Servlet** is an interface used by Java to communicate with the Web Server and write down figures. The working steps of Servlet is:

1. The user sends message to the Web server.
2. The Web server call the Servlet, the Servlet responds the message and send the response back to the server.
3. The server sends the response to the user.

The Servlet API in Java contains four package (recall what is API in Day 4), including

1. javax.servlet
2. javax.servlet.http
3. javax.servlet.annoutation
4. javax.servlet.descripter

The main package among them is **javax.servlet** , it should be implemented by all the Servlet classes.

It contains several methods to use

```
public interface Servlet {
    //initialization, only called once
    void init(ServletConfig var1) throws ServletException;

    ServletConfig getServletConfig();

    void service(ServletRequest var1, ServletResponse var2) throws
ServletException,        IOException;

    String getServletInfo();

    void destroy();
}
```

Among them, the `service()` method takes a `ServletRequest` , which is a class that seals a http request sent by the user. Similarly the `ServletRequest` is a class that seals a http response to be sent to user. The programmer can directly use the methods provided in these classes in API, and does not need to know the implementation.

The package **javax.servlet.http** totally extends the **javax.servlet** (actually the GenericServlet, which is an updated servlet) package and is currently widely used in the development industry. The extension is

```
public abstract class HttpServlet extends GenericServlet implements Serializable
{
    ...
}
```

Correspondingly, we have the `ServletRequest` and `ServletResponse` extended as well

```
public interface HttpServletRequest extends ServletRequest {
    ... //Details omitted. But it extends and add some new methods
}
public interface HttpServletResponse extends ServletResponse {
    ... //Details omitted. But it extends and add some new methods
}
```

By overriding the `service` method in GenericServlet, the HttpServlet has its own `service` method

```
public void service(ServletRequest req, ServletResponse res) throws
ServletException, IOException {
    HttpServletRequest request;
    HttpServletResponse response;
    try {
        request = (HttpServletRequest)req;   //since the type is determined
        response = (HttpServletResponse)res; //forced type conversion is
possible here
    } catch (ClassCastException var6) {
        throw new ServletException("non-HTTP request or response");
    }
    this.service(request, response);
}
```

Note that the `this.service(request, response)` calls a new `service` method in `HttpServlet` , it analyzes the message in `request` and call one of the http methods like `doGet` or `doPost` . These http methods will write on the `response` and result in a proper output.

Filter

In the practical development, the user may make some request like the program should be able to defend itself from hacking or some bad attack, or simply filter out some sensitive words. This is realized by adding a filter. In the servlet API we has a filter interface, we can implement it to realize the filter function. Usually the filter class has three methods to be implemented, `init`, `doFilter` and `destroy`. The interface are shown below

```
package javax.servlet;

import java.io.IOException;

public interface Filter {
    void init(FilterConfig var1) throws ServletException;

    void doFilter(ServletRequest var1, ServletResponse var2, FilterChain var3)
throws IOException, ServletException;

    void destroy();
}
```

Among them, the `doFilter` methods implements how the filter gets the information filtered. In reality, we may have several filters connected together, we may regard them as a chain. Indeed, the file `web.xml` requests programmer to define the order of filters. When the first filter is called, the Web will send a `FilterChain` object to it. Then we can use the filter like

```
public void doFilter(ServletRequest request, ServletResponce responce,
FilterChain chain){
    ...
    chain.doFilter(ServletRequest request, ServletResponce responce, FilterChain
chain);
    //This sentence calls the next filter, if there is no next filter, return
    ...
}
```

It acts like a recurrence relation, we operate one filter, and go to next filter (it will end automatically). So we just need one line of `chain.doFilter`, all the filters will be called in the pre-defined order.

Collection/Set & List

The collection is the most common interface for a group of elements, the set and list are two different inheritance of collection. The different points are: List contains no duplicate elements, like vector or linked list; Set may contain duplicate elements.

Extra Day

First SpringBoot Program

I built my first SpringBoot program in home according to the guideline in

https://blog.csdn.net/baidu_39298625/article/details/98102453

The program can be visited in

https://github.com/TaikiShuttle/A-Basic-SpringBoot-Program

I will keep it updated and add more features.

How to write a simple SpringBoot Program

The SpringBoot is the most popular tool for MVC programming, and it also has a strong function in web development. The following steps are the procedure to show a sentence in a website:

1. First choose to build a new project using the **Spring Initializr** function, the detailed settings should be set to match the version in your computer.
2. Then we choose the modules we need in the interface, we need **Spring Web** in Web, **Thymeleaf** in Template Engines, **JDBC API**, **MySQL Driver** and **MyBatis Framework** in SQL.
3. Next we choose the path and then click ok.
4. Then we go to the **Maven Settings** in the right hand side, change the settings to the local Maven directory.
5. We find the folder **Resources->templates** and add a new html file called **index.html**, then we enter the file and add the following code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>hello</title> //this line is the title of the website
</head>
<body>
    Hello world. My name is Haoquan, a beginner in Springboot!
    //This line is the context of the website
</body>
</html>
```

6. Then we go to the main package under **java** directory, create a new package called **controller** (must be this name). Under the package we create a new class, say **hellocontroller**, and the following code

```
package com.example.test.controller; //The directory of this file may be
different


import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class hellocontroller {

    @RequestMapping("/index")
    public String sayHello() {
        return "index";
    }
}
```

7. Under the **resources** directory, we can find the **application** file, we can either edit it in **properties** format or in **yml** format. The default format is properties. To change it to yml, we can right click it and refactor->rename it. Then we add following code

```
spring:
  datasource:
    name: test  #database name
    url: jdbc:mysql://localhost:3306/test #url
    username: root  #username
    password: 123456  #password
    driver-class-name: com.mysql.jdbc.Driver  #database driver
```

8. Now we go to the Application class under the main directory, and we find it has already been written for us. We can directly run it.
9. Open the website and go to `localhost:8080`, we can see the sentences printed on the website.
10. The following jobs are creating a database with **MySQL** and do some manipulations on it. The basic programming structure follows the Controller, Service, Dao layers.

Day 6

Multi-Thread

What is thread

A thread is a certain part of a progress, it occupies a certain period of the progress. A sequence of threads construct a progress.

How to create a thread in Java

In Java, the essence interface for thread is `java.lang.Runnable`, also, a class `java.lang.Thread` implements the `java.lang.Runnable`, so we have two different ways to generate new threads.

1. We can create new instances in `Thread` type to represent different thread. There are two different constructor of `Thread` class

   ```
   public Thread(String threadName) {
       ...
   }
   public Thread() {
       ...
   }
   ```

   Usually we do not construct the thread instance directly, since the main function of a thread is written in the method `Thread.run()`, to create a self-defined thread, we need to inherit `Thread` and override the `run()` method. For example

   ```
   public class ThreadTest extends Thread {
       ...
       public void run() {
           ...
       }
   }
   ```

   Then in the main function, we can start the thread like

   ```
   public static void main(String [] args) {
       new ThreadTest().start(); //start() will call run()
   }
   ```

   Note that here the `start()` method will call the `run()` method, before the call of `start()`, this thread is only a instance which has not been started. Only after the `start()` method, the thread is launched and begins to work.

2. Note that in 1, we must write the program in the `run()` method which requires us to inherit the `Thread()` class. But if we already has a class inherited from other class, then it is impossible for us to both inherit its superclass and `Thread()`, since Java does not support multi-inheritance. So we need to implement the `Runnable` interface directly.

```
public class ThreadState implements Runnable {
    ...
    public void run() {
        ...
    }
}
```

In this case, we need to first implement the `Runnable` interface, as what is written above, then we need to create a new instance using this implementation class. Furthermore, we need to create a new `Thread` instance using this `Runnable` instance. Last, we can then use the `start()` method to create a thread. For example

```
public class test {
    public static void main(String [] args) {
        //A class implements Runnable interface
        ThreadState state = new ThreadState();

        //Use Runnable instance to create thread instance
        Thread thread = new Thread(state);

        //ready to start
        thread.start();
        ...

    }
}
```

Day 7&8

MyBatis

MyBatis is kind of like a plugin that serves as the bridge between database like **MySQL** and the **Java programs**. We can just create a **.xml** file and a **Mapper** interface, by configuring the .xml file, MyBatis will use the database to create an implementation class for the Mapper interface automatically. For example, we create a UserMapper interface for the Mapper layer of a database

```
package com.example.test.mapper;

import com.example.test.bean.UserBean;

public interface UserMapper {
    //Note that if we do not add prefix like "public", Java treats it as
"default"
    //Which means that the member is visible for all member in the same package
    UserBean getInfo(String name, String password);
}
```

Then we can config the UserMapper.xml as

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.example.test.mapper.UserMapper">

    <select id="getInfo" parameterType="String"
resultType="com.example.test.bean.UserBean">
        SELECT * FROM people WHERE name = #{name} AND password = #{password}
    </select>

</mapper>
```

it will select all the elements that satisfy the condition `name = #{name} AND password = #{password}`, where `name` and `password` are the parameter taken by `getInfo(String name, String password)`. If there is no such element, it will return `null`.

A SpringBoot Program Implementing a Simple Course Searching Website

Based on the work done in the extra day, I further developed the program and add more features to it.

The code can be seen at

https://github.com/TaikiShuttle/A-SpringBoot-Program-Implementing-a-Simple-Course-Searching-Website

1. Create a database using **MySQL** and connect it to IDEA. In this database, I created two tables named `people` and `course`, in order to store the user information and course information.
2. The program is built based on the **Controller->Service->DAO** structure. The basic idea is that one can first login the database using his/her account, then he/she can search for some pre-recorded courses.
3. **The DAO Layer**: This layer contains **two mapper interfaces** and two **.xml files**. Just like the UserMapper discussed before, the CourseMapper is similar to it. The only difference is that the CourseMapper only takes one argument `course_id` since we only allow searching use course code.
4. **The Service Layer**: This layer contains two interfaces and two implementation classes. Since the work we need to implement is just searching for courses, there is no extra manipulation for the Service Layer to do. Just take the element from the DAO Layer and send it to the Controller Layer. For example, the UserService interface and implementation class are

```
package com.example.test.service;

import com.example.test.bean.UserBean;

public interface UserService {
    UserBean loginIn(String name, String password);
}
```

```
import com.example.test.mapper.UserMapper;
import com.example.test.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

//Note that this @Service is necessary
```

```
@Service
public class UserServiceImpl implements UserService {

    //@Autowired is a field injection method, it will make the compiler
    //searching for the implementation automatically.
    //So we can use the interface to declare instances
    @Autowired
    private UserMapper userMapper;

    @Override
    public UserBean loginIn(String name, String password) {
        return userMapper.getInfo(name, password);
    }
}
```

5. **The Controller Layer**: This layer contains several classes which controls the behavior of the website, for example, jump from one website to another one if certain condition is satisfied. It also contains several .html files which contains the context of the website. For example, the `LoginController` controls the state of the login page. If the user enter a correct username and password, it will go to `success.html`, if the username or password are wrong, it will go to `error.html`, if the user does nothing, it will stay in current website. The code are as follows

```
package com.example.test.controller;

import com.example.test.bean.UserBean;
import com.example.test.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

//Note this @Controller is necessary
@Controller
public class LoginController {

    //inject service layer to controller layer
    @Autowired
    UserService userService;

    @RequestMapping("/login")
    public String show(){
        return "login";
    }

    @RequestMapping(value = "/loginIn",method = RequestMethod.POST)
    public String login(String name,String password){
        UserBean userBean = userService.loginIn(name,password);
        if(userBean!=null){
            return "success";
        }else {
            return "error";
        }
    }
}
```

While the .html files are quite frontend-sensed, I will omit them.

Day 9

Finish the Course-searching Website

I mentioned in the program description in GitHub that the Day 8 version does not support showing the detailed information when searching. This is because I was not able to transmit the variable in Java to the html file. However, I have learned today that the IDEA use one html editor called **thymeleaf**, by default. Note that in the extra day, when I create this program, I have included the **thymeleaf** package. It can transmit the Java variable in the controller layer to the html file using a **Model**. The detailed method is as follows:

1. In the controller layer we write like this:

```
package com.example.test.controller;

import com.example.test.bean.CourseBean;
import com.example.test.service.CourseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.server.DelegatingServerHttpResponse;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import java.util.Objects;

@Controller
public class searchController {

    @Autowired
    CourseService courseService;

    @RequestMapping("/course_search")
    public String show() {
        return "course_search";
    }

    @RequestMapping(value = "/Search_Result", method = RequestMethod.POST)
    public String go_result(String course_id,Model model) {
        CourseBean courseBean = courseService.resultLT(course_id);
        if(courseBean!=null) {
            return courseShow(model,courseBean);
        }
        return "badresult";
    }

    //This courseShow function take a model argument, which contains
    //the information we'd like to send to html file
    @RequestMapping(value = "/Search_Result", method = RequestMethod.GET)
    public String courseShow(Model model,CourseBean courseBean) {
        //This line send the information courseBean to html file
        //as a variable called course
        model.addAttribute("course",courseBean);
        return "result";
```

```
    }
}
```

2. Then we can use the created variable `course` in the html file as

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Result</title>
</head>
<body>
<p th:text="'Course ID: '+${course.getCourse_id()}"></p>
<p th:text="'Course Name: '+${course.getCourse_name()}"></p>
<p th:text="'Instructor: '+${course.getInstructor()}"></p>
</body>
</html>
```

   Here, `th:` for thymeleaf language, and the `+` is a connector, which does not show on the screen.

This blog gives the complete version, which includes adding/searching/deleting/updating a course.

https://blog.csdn.net/xuxin132133/article/details/83342525

But it may be harder to consider the frontend side.