고급 웹 프로그래밍 Term Project

빈 강의실 검색 시스템

TEAM. K	
김은지	201116817
이상윤	201017093
유주형	201125246
담당교수	김현태
제출일	2013.12.20

KU 건국대학교 KONKUK UNIV.

목차

1. 프로젝트 소개

1.1 개요

이 프로젝트는 학교에서 제공해주는 학사 정보들을 더 효율적으로 활용하기 위해서 건국대학교 글로컬 캠퍼스의 학생들을 타겟으로 어떠한 서비스를 제공할 수 있을까 고민하다, 도서관 자리가 한정되어 있기에 모든 학생들이 시험기간 같은 특정 기간에 한 장소에 몰려서 자리 다툼을 하는 것을 분산시키기 위해 고안한 시스템이다.

실질적으로 학교에 데이터베이스가 구축되고 학교 내에서 이용되고 있지만 이를 활용한 어플리케이션의 수가 많은 것은 아니다. 또한, 학생들이 직접적으로 이 정보를 활용해 편의성이 있는 서비스를 제공받기도 어려운 것이 현실이다.

빈 강의실을 비어두기 보다는 학생들을 위해서 공개하고, 더 효율적으로 활용할 수 있도록 빈 강의실을 찾아 학생들이 공부함에 있어 편리한 정보를 제공해 주는 시스템이다.

학생들은 늘어나고 있지만 부족한 자습실로 인해 많은 불편을 호소하고 있는 상황에서, 학교에서 직접적으로 제공해주는 데이터베이스를 이용하여 이를 이용해 빈 강의실을 검색하고 이 강의실을 학생들이 자율적으로 자습실로 이용할 수 있도록 이를 활용하는 어플리케이션을 구현하였다.

이 시스템은 학교에서 시간표 관련된 데이터베이스를 제공받아 온다는 전제하에 설계되었다. 실제로 시스템을 설계할 때 학사 정보를 확인하여 데이터베이스를 근접하게 설계할 수 있도록 노력하였다.

이러한 정보는 제공받는 학생 입장에서는 PC 보다는 모바일 환경에서 이용하는 경우가 더 많기에 모바일 환경에 맞춰서 프로젝트를 디자인하였다.

1.2 목표

빈 강의실 찾기 시스템의 최종적인 목표는 특정 기간에 공부할 수 있는 장소, 도서관의 부족한 자리를 분산시키고 학생들이 원하는 시간에 원하는 장소에서 편하게 공부를 진행 할 수 있는 검색과 정보를 제공해주는 것이다.

빈 강의실을 모두 열어두는 것은 또 아니지만, 이러한 정보가 학교에서 제공해주는 정보와 결합된다면 정확성뿐만 아니라 학생들의 편의를 더 증진시킬 수 있다고 판단했다.

1.3 프로젝트의 구성

이 프로젝트는 아래와 같은 구조로 구성되어 있다.

Spring 3.2.2.RELEASE 버전과, MyBatis 3.2.4-SNAPSHOT 버전과 Hibernate Validator 4.2.0.Final를 이용해 프로제트 진행에 있어 편의성과 효율성을 증가시킬 수 있도록 하였으며, RESTful한 서비스를 제공해주기 위해서 URL을 이용해서 편한 접근을 할 수 있도록 하였다.

- DAO (Interface)

    - StudentDao.java
    - TimetableDao.java
- Mapper (MyBatis)
    - studentMap.xml
    - timetableMap.xml
- DAO (Implementation)
    - StudentDaoImpl
    - TimetableDaoImple
- VO
    - Student.java
    - Timetable.java
- Controller
    - HomeController.java
    - JoinController.java
    - LoginController.java
    - TimetableController.java

Spring MVC를 이용하여 MVC 패턴에 맞춰서 Model, View 그리고 Controller를 분리하여 유지보수가 쉬운 코드를 작성할 수 있도록 노력하였다.
데이터베이스는 MySQL을 이용하였으며, Apache Tomcat 7 환경에서 구동이 정상적으로 이뤄지는 것을 확인하였다.

1.4 구성원 및 기여도

학번	이름	역할
201116817	김은지	조장
201017093	이상윤	조원
201125246	유주형	조원

항목	구성원	기여도(%)
모듈 설계 및 개발	이상윤	70
	김은지	20
	유주형	10
테스트 및 모듈 통합	이상윤	70
	김은지	20
	유주형	10

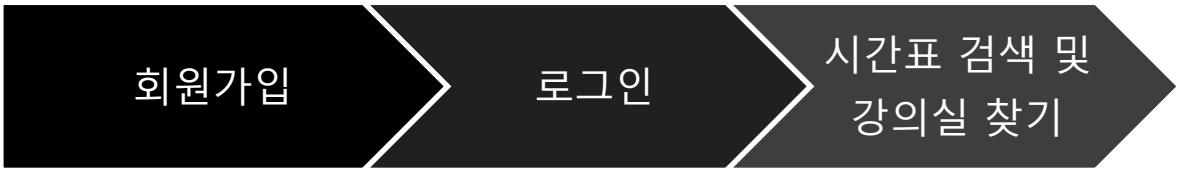
보고서 작성	이상윤	70
	김은지	10
	유주형	20

(참고) 모듈 설계 및 개발 안에 데이터베이스를 구축하는 과정까지 포함하였음.

2. 개발 어플리케이션

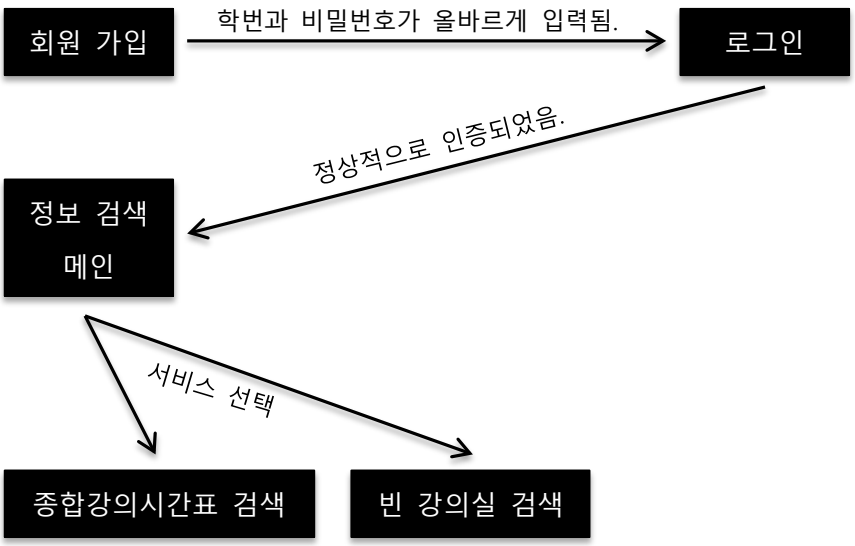
2.1 어플리케이션 구성

빈 강의실 찾기 시스템의 흐름은 아래와 같다.



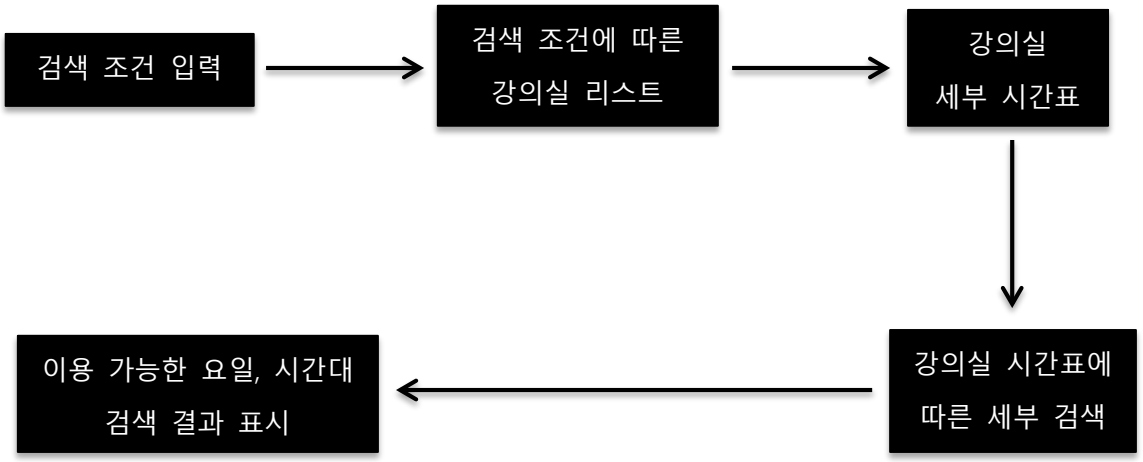
학생들은 학사 정보(학생 포탈에 SID와 Password)를 이용해 로그인할 수 있다.
이 과정은 프로젝트에서 회원가입과 로그인 부분으로 나누었다. 실제로 학생 데이터를 가져와 활용할 수 없기 때문에 추가적으로 가입과 로그인 부분을 넣어줬다.
실제로 학교에서 제공받을 수 있거나 이용할 수 있다면 이 부분은 생략 가능한 부분이다.

로그인 후 인증이 정상적으로 이루어지면 종합강의 시간표를 검색할 수 있고, 학생이 원하는 시간에 원하는 장소에 원하는 층에 관련하여 빈 강의실을 찾을 수 있다.
해당 서비스를 처음 이용하는 사람은 아래와 같은 흐름으로 서비스를 이용하게 된다.
기존 서비스를 이용해본 사람은 로그인 후 원하는 서비스를 선택하면 된다.



회원 가입과 로그인에서는 각 검증 과정이 포함되어 있다. SID(학번)과 비밀번호가 틀리면 다음 단계로 진행 할 수 없다. 또한, 인증 없이 아래의 정보를 활용할 수 없도록 세션을 이용해서 세션이 없다면 로그인을 요구하게 된다.

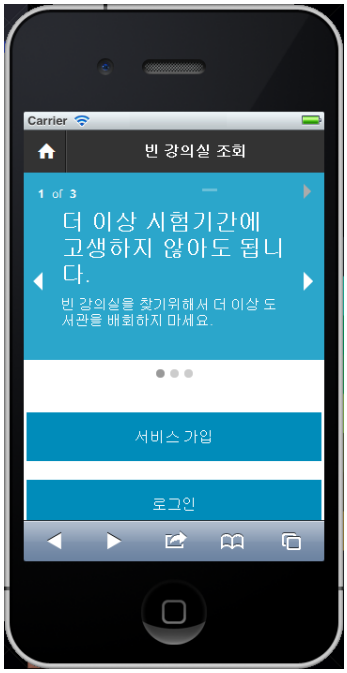
빈 강의실 검색은 아래와 같은 흐름으로 작동된다.



일차적으로는 장소, 층수, 시간대에 따라서 검색 조건을 입력해 전체적으로 이용 가능한 강의실의 리스트를 뽑아준다.
원하는 장소를 선택하면 세부적인 시간표를 확인할 수 있도록 하였다.
마지막으로, 해당 강의실 시간표를 통해서 요일과 시간대(교시)를 선택해 검색을 하면 원하는 시간에 이용가능한지 불가능한지 여부를 시각적으로 표시해주는 구조이다.

다음 장에서는 해당 어플리케이션에 기능을 소개하고자 한다.

2.2 기능 소개



<그림1> 모바일 웹에서 접근한 메인 페이지

메인 페이지에서는 서비스에 가입하거나 로그인 할 수 있는 서비스만 제공된다.

모바일 환경에 맞춰 디자인하였기에 쓸데 없는 이미지는 이용하지 않았고 Icon Font와 CSS만 이용하였다.

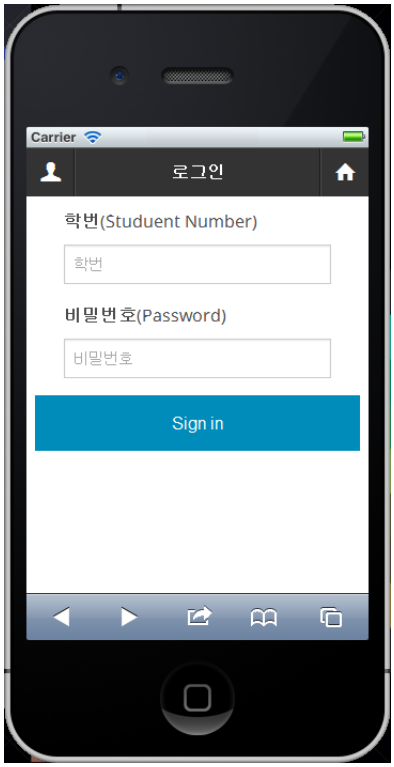
<그림2> 서비스 가입 페이지

서비스 가입 페이지에서는 최소 정보만 요구할 수 있도록 했다.

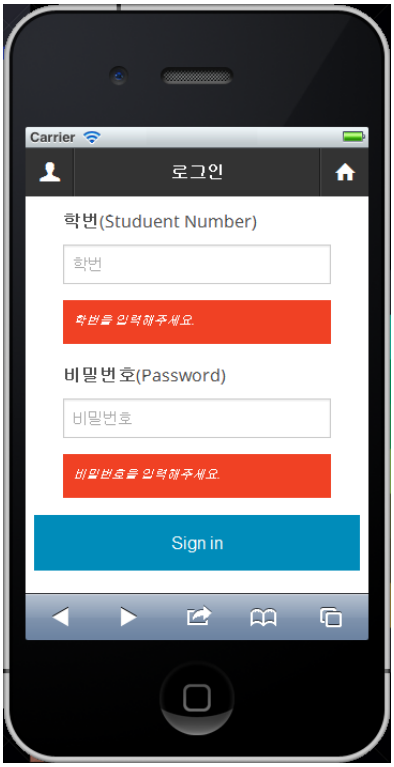
<그림3> 학번, 암호 입력오류

<그림4> 암호 입력오류

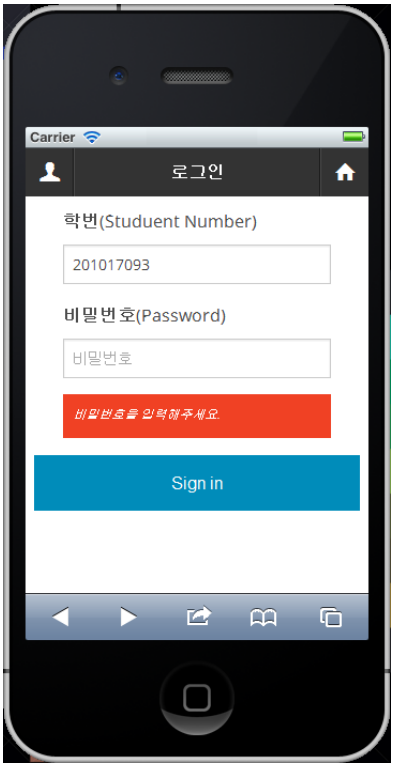
기본적으로 로그인에 필요한 학번과 암호는 필수적으로 입력해야 한다. 한 필드라도 입력이 되지 않았다면 오류 메시지를 출력해주는 구조이다.



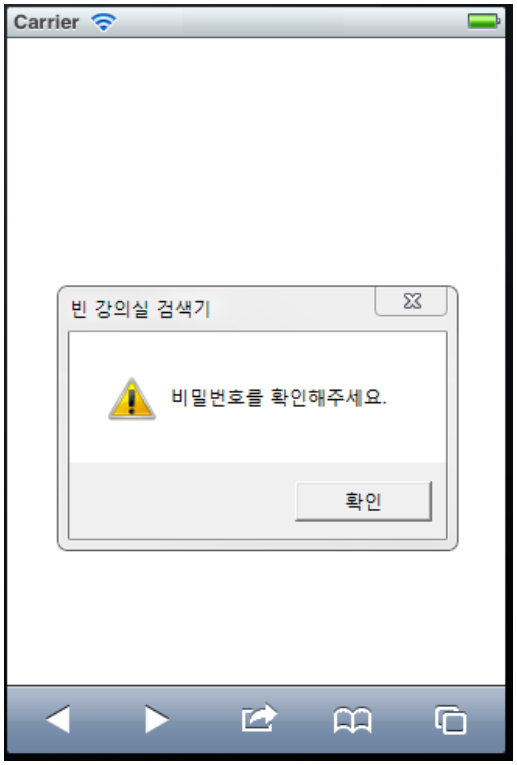
<그림5> 로그인 페이지

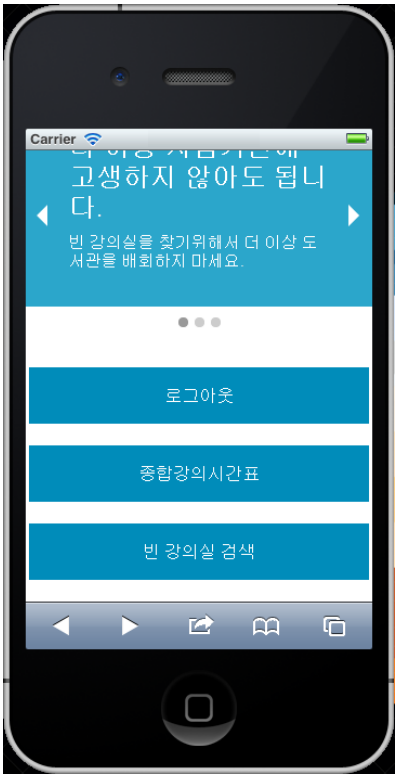


<그림6> 학번, 비밀번호 입력오류



<그림7> 비밀번호 입력 오류





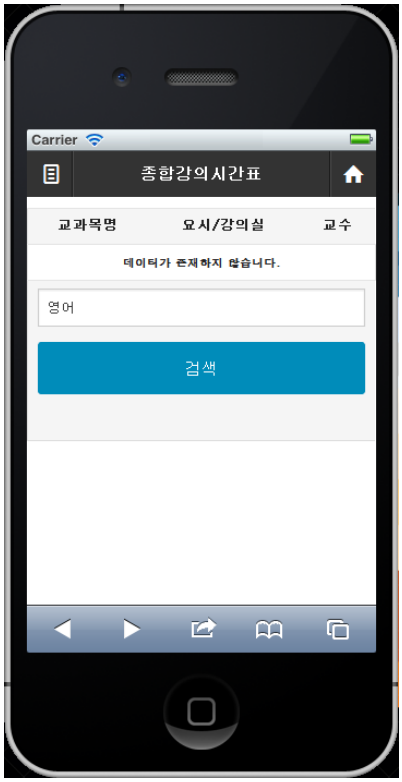
<그림8> 비밀번호 검증 오류 **<그림9> 로그인 후 서비스 메인페이지**

로그인을 하는 과정에서도 학번과, 비밀번호는 필수 입력 정보이므로 한 필드라도 공백이라면 오류 메시지를 출력해주게 된다. 로그인은 학번과 비밀번호 모두 검증을 한다.(그림8)
로그인이 완료되어 정상적으로 인증되었다면 그림9와 같은 서비스 제공 메인 페이지가 보여지게 된다. 이는 세션을 통해서 처리되는 부분이다.

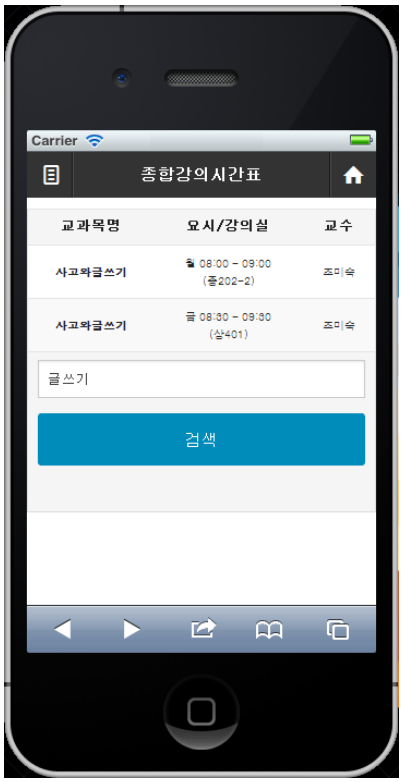
종합강의 시간표는 학교에서 제공하는 강의 시간표를 검색, 확인할 수 있는 서비스이다.
빈 강의실 검색 서비스는 본 어플리케이션의 주 목적이 되며 빈 강의실을 검색해 시각적으로 표시해주는 서비스 이다.

(다음 장에 계속)

<그림10> 종합강의시간표



<그림11> 검색결과 없을 때



<그림12> 검색결과 있을 때

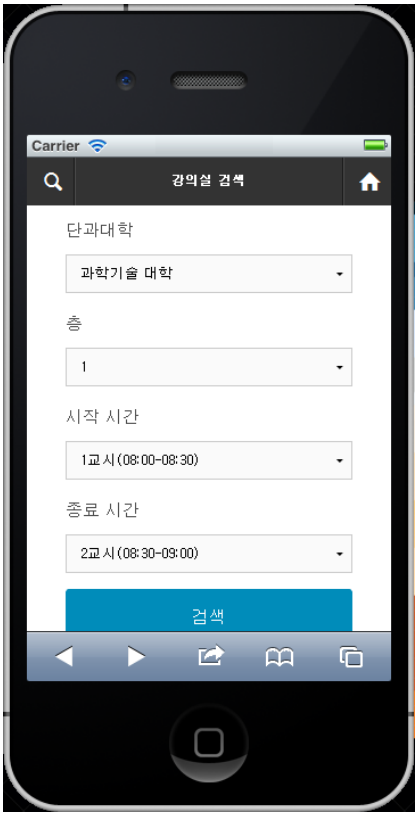
빈 강의실은 아래와 같은 조건으로 검색할 수 있다.

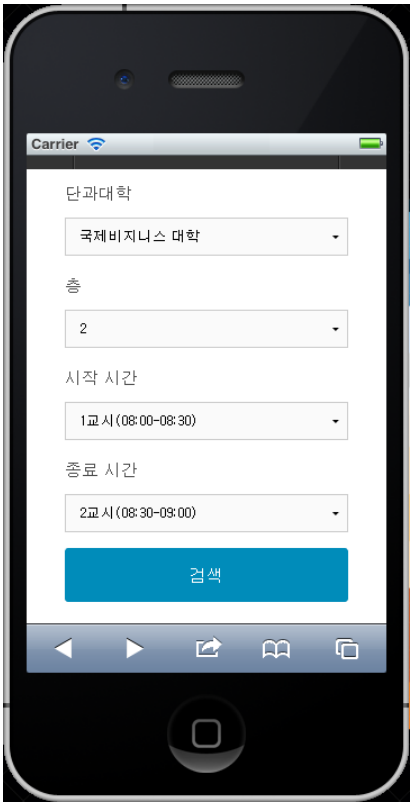
- 어느 단과 대학에서 공부하고 싶은가?
- 몇 층에 강의실을 이용하고 싶은가?
- 원하는 시간대가 언제인가?

위의 3가지 조건을 통해 최적의 장소를 찾아주게 된다.

또, 해당 강의실에서 어느 요일에 어느 시간대에 이용가능한지는 세부적인 페이지에서 확인할 수 있다.

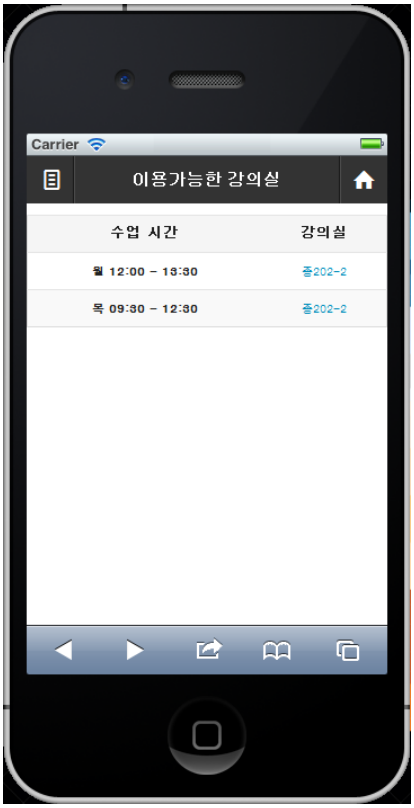
(다음 장에 계속)





<그림13> 강의실 검색페이지 **<그림14> 원하는 조건을 입력해봤음**

강의실 검색 조건은 앞 장에서 소개한 그대로다. 그림 14에서는 원하는 시간대에 원하는 조건을 입력해보았다.



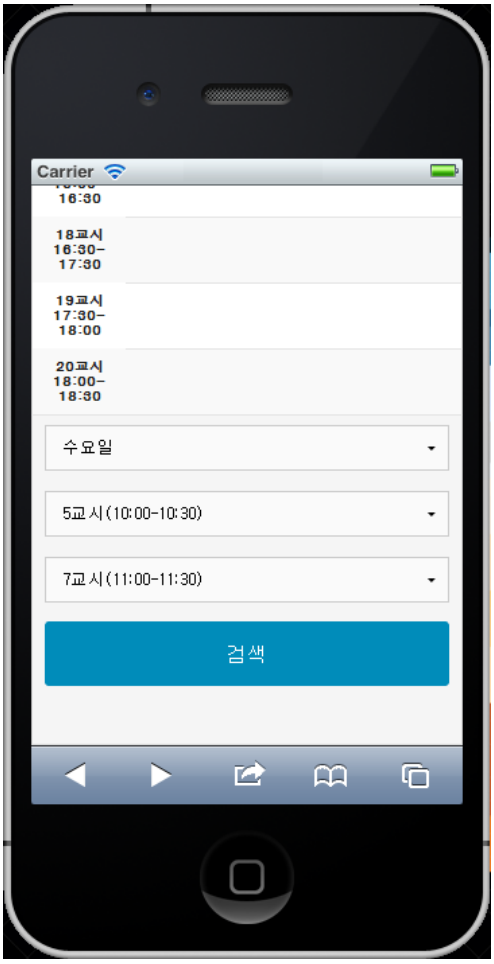
<그림15> 검색 결과

이용가능한 강의실 리스트를 출력시켜주고 수업이 진행되는 시간을 표시해준다.
강의실 명을 누르면 해당 강의실의 세부 시간표를 확인할 수 있다.

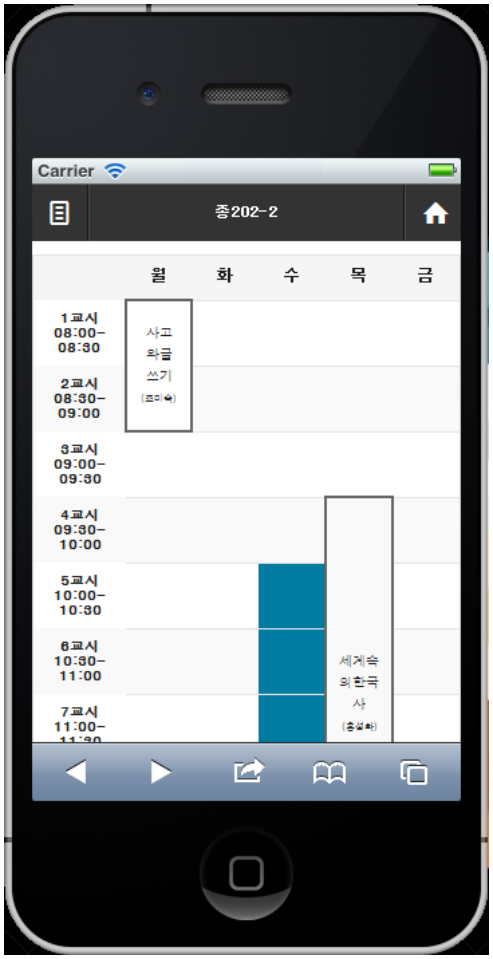
<그림16> 종합강의동 202-2의 시간표

해당 강의실에서는 요일과 시간대 조건을 이용해서 더 세부적으로 검색을 진행할 수 있다.

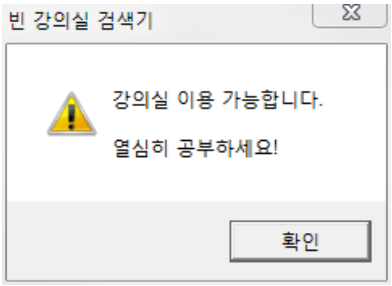
(다음 장에 계속)



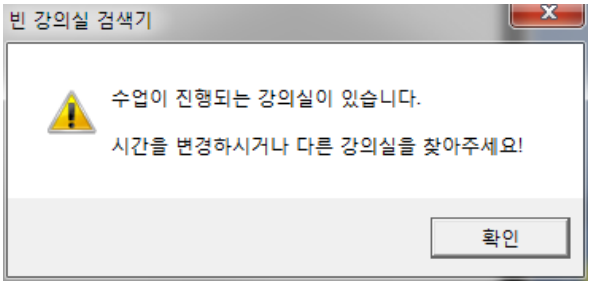
<그림17> 검색 조건 입력



<그림18> 결과를 시각적으로 보여줌.



<그림19> 이용 가능



<그림20> 이용 불가능

강의실이 이용가능한지 불가능한지는 경고창(Alert)을 통해서 알려주게 된다.
이용 가능하면 그림 19처럼 시각적인 결과를 제공해준다.

2.3 주요 코드 소개

Student VO는 아래와 같은 구조로 이루어져 있다.

이외에는 Constructor와 Getter, Setter Method가 구현되어 있다.

```
public class Student
{
    @NotBlank private String sid;
    @NotBlank private String password;
    private String name;
    private String email;
    private int grade;
    private String major;
```

비밀번호는 SHA-1 암호화 방식을 이용하기 때문에, getHash 메서드를 통해서 비밀번호의 해시 값을 알 수 있다.

```
public String getHash( String password )
{
    String hashStr = null;

    try
    {
        MessageDigest md = MessageDigest.getInstance( "SHA-1" );
        md.update( password.getBytes() );

        BigInteger hash = new BigInteger( 1 , md.digest() );
        hashStr = hash.toString( 16 );

        return hashStr;
    }
    catch( NoSuchAlgorithmException e )
    {
        e.printStackTrace();
    }

    return hashStr;
 }
```

Database Access Object는 아래와 같은 구조로 이루어져 있다.

```
public interface StudentDao
{
    public int insert( Student student );
    public int update( Student student );
    public int delete( String sid );

    public Student getOne( String sid );
    public List< Student > getAll();
}
```

```java
public class StudentDaoImpl extends SqlSessionDaoSupport implements StudentDao
{
    public int insert( Student student )
    {
        return getSqlSession().insert( "studentDao.insert" , student );
    }

    public int update( Student student )
    {
        return getSqlSession().update( "studentDao.update" , student );
    }

    public int delete( String sid )
    {
        return getSqlSession().delete( "studentDao.delete" , sid );
    }

    public Student getOne( String sid )
    {
        return ( Student ) getSqlSession().selectOne( "studentDao.getOne" , sid );
    }

    @SuppressWarnings( "unchecked" )
    public List< Student > getAll()
    {
        return getSqlSession().selectList( "studentDao.getAll" );
    }
}
```

StudentDao Interface를 위에서 SqlSessionDaoSupport를 상속받아서 구현했으며,

```xml
<mapper namespace="studentDao">
    <insert id="insert" parameterType="com.timetable.vo.Student">
        INSERT INTO students
        ( sid , password , name , email , grade , major )
        VALUES
        ( #{sid} , #{password} , #{name} , #{email} , #{grade} , #{major} )
    </insert>

    <update id="update" parameterType="com.timetable.vo.Student">
        UPDATE students SET
        password = #{password} ,
        name = #{name} ,
        email = #{email} ,
        grade = #{grade} ,
        major = #{major}
        WHERE sid = #{sid}
    </update>

    <delete id="delete" parameterType="String">
        DELETE FROM students
        WHERE sid = #{sid}
    </delete>

    <select id="getAll" resultType="java.util.Map">
        SELECT * FROM students
    </select>

    <select id="getOne" parameterType="String" resultType="com.timetable.vo.Student">
        SELECT * FROM students
        WHERE sid = #{sid}
    </select>
```

Mybatis Mapper를 이용해서 각 CRUD를 시행하는 구조로 작성되었다.

Timetable DAO도 비슷하게 구현되어있다.

특히, 검색 조건을 위해서 아래와 같은 Mapper가 설정되어 있다.

```xml
<select id="getBySubject" parameterType="String" resultType="com.timetable.vo.Timetable">
    SELECT * FROM timetables
    WHERE subject LIKE CONCAT( '%' , #{subject} , '%' )
</select>

<select id="getByPlace" parameterType="String" resultType="com.timetable.vo.Timetable">
    SELECT * FROM timetables
    WHERE place LIKE CONCAT( #{place} , '%' )
</select>
```

로그인을 처리하는 Controller에서는 /login으로 들어오는 요청에 대해서 처리해주게 되며, StudentDAO를 이용해서 서비스를 제공한다.

```java
@Controller
@RequestMapping( "/login" )
public class LoginController
{
    @Autowired
    StudentDao studentDao;
```

GET으로 들어오는 요청에 대해서는 Login Form을 보여주고, POST로 들어오는 요청은 로그인 처리를 해주는 부분이다.

```java
@RequestMapping( value = "" , method = RequestMethod.POST )
public String loginProcess( @Valid @ModelAttribute( "loginStudent" ) Student student ,
                            Errors errors , HttpServletRequest request , Model model )
{
    if( errors.hasErrors() )
        return "/member/login";
    else
    {
        Student loginStudent = studentDao.getOne( student.getSid() );

        if( loginStudent == null )
        {
            model.addAttribute( "errorMessage" , "학번을 확인해주세요." );

            return "/member/login";
        }
        else
        {
            String chkPasswd = student.getHash( student.getPassword() );
            boolean check = chkPasswd.equalsIgnoreCase( loginStudent.getPassword() );

            if( check == true )
            {
                request.getSession().setAttribute( "student" , loginStudent );

                return "redirect:/";
            }
            else
            {
                model.addAttribute( "errorMessage" , "비밀번호를 확인해주세요." );

                return "/member/login";
            }
        }
    }
}
```

에러가 있다면 해당 에러 메시지를 각 각 출력해주는 구조이고 비밀번호를 검증해서 해시 값이 동일하면 로그인 처리를 하며 세션을 생성한다. 이를 이용해서 회원 인증을 검증한다.

```java
@RequestMapping( value = "/logout" , method = RequestMethod.GET )
public String logoutPage( HttpSession session )
{
    session.setAttribute( "student" , null );
    session.invalidate();

    return "redirect:/";
}
```

로그 아웃 시에는 세션을 파기해주고 메인 페이지로 리다이렉트 시켜주는 구조이다.

/join으로 들어오는 요청에 대해서는 JoinController가 처리해주게 된다.

```
@Controller
@RequestMapping( "/join" )
public class JoinController
{
    @Autowired
    StudentDao studentDao;
```

Select(선택 상자) 필드에 값을 Model에 Attribute로 설정해주어서 Foreach를 이용해서 출력 효율을 높였다.

```
private List< Integer > gradeList = new ArrayList< Integer >( Arrays.asList( 1 , 2, 3 , 4 ) );
private List< String > majorList = new ArrayList< String >(
       Arrays.asList(
           "신문방송학과" , "디이나믹미디어학과" , "융합미디어콘텐츠학과" , "커뮤니케이션문화학부" , "영미어문학과 " ,
           "패션디자인학부" , "산업디자인학부" , "시각영상디자인학부" , "실내디자인학과" , "회화학과" ,
           "영어학과" , "경영경제학부" , "국제통상·문화학부" ,
           "경찰학과" , "행정복지학부" , "문헌정보학과" , "유아교육과" , "자율전공학부" ,
           "특성기술융합학과" , "나노전자기계공학과" , "컴퓨터공학과" , "스포츠과학부" ,
           "의생명융학과" , "생명공학과" , "식품생명과학부" , "의학공학부" , "간호학과"
       )
);
```

```
model.addAttribute( "gradeList" , gradeList );
model.addAttribute( "majorList" , majorList );
```

```
<tr>
    <td><label>학년</label></td>
    <td>
        <select name="grade" id="grade">
        <c:forEach items="${gradeList}" var="data">
            <option value="${data}">${data} 학년</option>
        </c:forEach>
        </select>
    </td>
</tr>
<tr>
    <td><label>전공</label></td>
    <td>
        <select name="major" id="major">
        <c:forEach items="${majorList}" var="data">
            <option value="${data}">${data}</option>
        </c:forEach>
        </select>
    </td>
</tr>
```

마지막으로, /timetable로 들어오는 요청에 대해서는 Timetable Controller에서 처리해주게 된다. 이 부분이 본 어플리케이션의 주 목적이 되는 빈 강의실 검색 부분에 속한다.

```
@Controller
@RequestMapping( "/timetable" )
public class TimetableController
{
    @Autowired
    TimetableDao timetableDao;
```

먼저, 시간표 데이터베이스는 아래와 같은 구조로 작성된다.

idx	subject	professor	week	time	place
1	사고와글쓰기	조미숙	1	1-2	종202-2
2	사고와글쓰기	조미숙	5	2-3	상401
3	삶의철학과성신의	이동기	1	9-11	종202-2
4	세계속의한국사	홍성화	4	4-9	종202-2

여기서 week의 숫자는 일요일(0)부터 토요일(6)까지 해당 요일의 숫자 코드를 의미하고, time은 해당 과목의 시간을 의미한다. 이는 교시로 표시되어, 시작-끝의 구조로 작성되어 –를 구분자로해 시간 표를 출력시키는 구조이다.

각 time에 따라서 시간 대를 미리 정의해 두고 그에 해당하는 시간대를 출력시켜주는 구조다.

```
private String[] timeSlot = {
        "08:00-08:30" , "08:30-09:00" , "09:00-09:30" , "09:30-10:00" ,
        "10:00-10:30" , "10:30-11:00" , "11:00-11:30" , "11:30-12:00" ,
        "12:00-12:30" , "12:30-13:00" , "13:00-13:30" , "13:30-14:00" ,
        "14:00-14:30" , "14:30-15:00" , "15:00-15:30" , "15:30-16:00" ,
        "16:00-16:30" , "16:30-17:30" , "17:30-18:00" , "18:00-18:30"
};
```

검색 결과를 보여주기 위해서 POST로 넘어온 파라미터들을 분석하는 부분은 아래와 같다.

```
@RequestMapping( value = "/search" , method = RequestMethod.POST )
public String searchForm( HttpServletRequest request , Model model )
{
    String collage = request.getParameter( "collage" );
    int floor = Integer.parseInt( request.getParameter( "floor" ) );
    int start = Integer.parseInt( request.getParameter( "start" ) );
    int end = Integer.parseInt( request.getParameter( "end" ) );
```

```
List< Timetable > timetableList = timetableDao.getByPlace( collage );

List< Object > timetable = new ArrayList< Object >();
for( Timetable obj : timetableList )
{
    String place2floor = obj.getPlace().split( "-" )[ 0 ].replaceAll( "[^\\.0123456789]" ,"" );
    int selectFloor = Integer.parseInt( place2floor ) / 100;

    if( floor != selectFloor )
        continue;

    int classStart = Integer.parseInt( obj.getTime().split( "-" )[ 0 ] );
    int classEnd = Integer.parseInt( obj.getTime().split( "-" )[ 1 ] );

    if( ( classStart <= start && classEnd >= start ) ||
        ( classStart <= end && classEnd >= end ) )
        continue;

    timetable.add( Arrays.asList(
            obj.getSubject() ,
            weekName[ obj.getWeek() - 1 ] ,
            timeSlot[ Integer.parseInt( obj.getTime().split( "-" )[ 0 ] ) - 1 ].split( "-" )[ 0 ]
                    + " - "
                    + timeSlot[ Integer.parseInt( obj.getTime().split( "-" )[ 1 ] ) - 1 ].split( "-" )[ 1 ] ,
            "<a href=\"" + obj.getPlace() + "\">" + obj.getPlace() + "</a>"
        ) );
}
```

들어온 요정에 따라서 데이터베이스를 질의해 값을 가져오고 이를 다시 재 가공하는 구조이다.

```
String place2floor = obj.getPlace().split( "-" )[ 0 ].replaceAll( "[^\\.0123456789]" ,"" );
int selectFloor = Integer.parseInt( place2floor ) / 100;
```

Place 필드에서 층 수를 뽑아오기 위해서 데이터 베이스 구조를 변경하지 않고 작업하기 위해서 숫자 부분만 뽑아온 후, 100으로 나눠서 계산해주는 구조로 작성했다.

```
int classStart = Integer.parseInt( obj.getTime().split( "-" )[ 0 ] );
int classEnd = Integer.parseInt( obj.getTime().split( "-" )[ 1 ] );
```

수업의 시작시간과 끝 시간은 위와 같이 –를 구분자로 계산해 분리하여 이용하였다.

검색 조건에 맞는 리스트를 다시 생성해서 출력시키는 구조이다.

세부 시간표는 RESTful하게 PathVariable을 이용해서 처리해 주었다. 이를 위해서 URI를 UTF-8로 인코딩하게 톰캣에서 설정해 주어야 한다.

```
@RequestMapping( value = "/{place}" , method = RequestMethod.GET )
public String detailPage( @PathVariable String place , Model model )
```

```
for( Timetable item : timetableList )
{
    int week = item.getWeek();
    int startTime = Integer.parseInt( item.getTime().split( "-" )[ 0 ] );
    int endTime = Integer.parseInt( item.getTime().split( "-" )[ 1 ] );

    int diff = endTime - startTime + 1;

    timetable[ startTime - 1 ][ week ] = "<td rowspan=\"" + diff + "\" class=\"bordered\">"
                                        + item.getSubject() +
                                        "<br /><span class=\"prof\">(" + item.getProfessor() + ")</span></td>";
    for( int i = startTime ; i < endTime ; ++i )
        timetable[ i ][ week ] = "";
}
```

데이터베이스에서 장소에 해당하는 값을 질의해 오고 이를 재 가공해 시간표 형태로 만들어주기 위해서 2차원 배열을 활용했다. 동일 시간표의 행 병합을 위해서 테이블을 직접 동적으로 생성해 주게 하였다. (HTML의 table 태그의 rowspan 어트리뷰트 이용)

세부 시간표 페이지에서 검색하는 부분은 POST 요청으로 처리하였다.

```
@RequestMapping( value = "/{place}" , method = RequestMethod.POST )
public String serachPage( @PathVariable String place , HttpServletRequest request , Model model )
{
    int start = Integer.parseInt( request.getParameter( "start" ) ) - 1;
    int end = Integer.parseInt( request.getParameter( "end" ) ) - 1;
    String weekday = request.getParameter( "week" );
```

각 파라미터들을 가져와서, 값을 재 가공하는 구조이다.

```
for( Timetable item : timetableList )
{
    int week = item.getWeek();
    int startTime = Integer.parseInt( item.getTime().split( "-" )[ 0 ] );
    int endTime = Integer.parseInt( item.getTime().split( "-" )[ 1 ] );

    int diff = endTime - startTime + 1;

    timetable[ startTime - 1 ][ week ] = "<td rowspan=\"" + diff + "\" class=\"bordered\">"
                                        + item.getSubject() +
                                        "<br /><span class=\"prof\">(" + item.getProfessor() + ")</span></td>";
    for( int i = startTime ; i < endTime ; ++i )
        timetable[ i ][ week ] = "";
}
```

```
boolean isEmpty = true;
for( int i = start ; i <= end ; ++i )
    if( temp[ i ][ weekNumber] != null )
        isEmpty = false;

if( isEmpty == true )
    for( int i = start ; i <= end ; ++i )
        if( temp[ i ][ weekNumber ] == null )
            timetable[ i ][ weekNumber + 1 ] = "<td class=\"highlight\"></td>";
```

값을 가져와서 비어있는 요일과 시간 대에는 css를 이용해서 시각적으로 표현해 준다.

3. 프로젝트 소감

프로젝트를 진행하면서 Spring Framework를 활용하여 MVC 구조를 이용해 프로그램을 설계하면서 더 복잡한 구조라도 MVC를 분리하면 편하게 작업을 할 수 있다는 것을 알 수 있게 되었다. 사실 이 어플리케이션은 학교에서 시간표나 학생 정보를 직접적으로 이용할 수 없기 때문에 약간 어색한 부분이 있지만 활용도는 높다 생각한다. 그 동안 프로젝트에서 잘 사용하지 않았던 MyBatis나 Hibernate를 추가적으로 활용해서 구현에 있어서 효율성을 최대한 증진시키고자 하였다. 아쉬운 것은 시간표를 생성해 줄 때 루프를 여러 번 돌기 때문에 효율적인 코드는 아니라 생각한다.

해당 프로젝트를 마무리 하면서 학사 정보를 보고 데이터베이스를 유추하고 설계하는 방식과 이를 활용하는 여러 가지 방법을 생각할 수 있었다. 주어진 정보가 어떻게 제공되느냐에 따라서 그에 맞게 데이터베이스를 수정하지 않고도 필요한 정보를 뽑아 파싱(Parsing)하는 구조가 효율적일 때가 있다고 생각해 이를 적극적으로 활용하고자 노력했다.

마지막으로, 타켓과 활용 기기에 따라서 사이트를 설계하면서 필요한 정보를 필요한 방식으로 제공해 편의성을 증진시킬 수 있음을 확인하였다. 모바일 환경에 맞춰서 그에 따라서 빠른 로딩을 위해서 쓸데 없는 이미지는 제거하고 간결하고 직관적으로 UX를 디자인 했다.