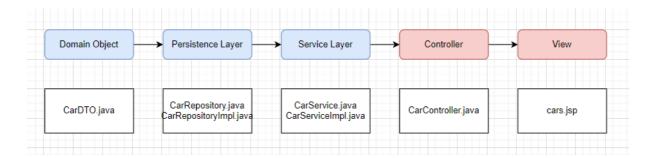
Spring Day 2 230228

Homework Review



<u>BoardDTO.java</u> <u>BoardRepository.java</u> <u>BoardService.java</u> <u>BoardController.java</u> boards.jsp

BoardRepositoryImpl.java BoardServiceImpl.java

board

boardDTO 제목btitle 내용bcontent 작성자bauthor 작성일 bdate

boards.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"</pre>
 pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>게시판 목록</title>
</head>
<body>
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
   rel="stylesheet"
   integrity="sha384-GLhlTQ8iRABdZLl603oVMWSktQ0p6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
   crossorigin="anonymous">
 <nav class="navbar navbar-expand navbar-dark bg-dark"
    aria-label="Second navbar example">
```

```
<div class="container-fluid">
    <a class="navbar-brand" href="#">CarShop</a>
    <button class="navbar-toggler" type="button"</pre>
      data-bs-toggle="collapse" data-bs-target="#navbarsExample02"
      aria-controls="navbarsExample02" aria-expanded="false"
      aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarsExample02">
      <a class="nav-link active"</pre>
         aria-current="page" href="/">喜</a>
        class="nav-item"><a class="nav-link" href="/cars">차량
           보기</a>
        class="nav-item"><a class="nav-link" href="/boards">게시판</a>
        <form role="search">
        <input class="form-control" type="search" placeholder="Search"</pre>
         aria-label="Search">
      </form>
    </div>
   </div>
 </nav>
 <div class="alert alert-dark">
   <div class="container">
    <h1>게시판</h1>
   </div>
 </div>
 <div class="container">
   <div class="row" align="center">
   <c:forEach items="${boardList}" var="board">
    <div class = "col-md-4">
      <h3>${board.btitle}</h3>
      ${board.bcontent}
      ${board.bauthor}
      ${board.bdate}
    </div> </c:forEach>
   </div>
 </div>
 <script
   crossorigin="anonymous"></script>
</body>
</html>
```

Spring Day 2 230228

2

BoardController.java

```
package com.carshop.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
@Controller
public class BoardController {
 @Autowired //DI
 private BoardService boardService;
 @RequestMapping("/boards")
 public String BoardList(Model model) {
   List<BoardDTO> list = boardService.getAllBoardList();
   model.addAttribute("boardList", list);
   return "boards";
 }
}
```

BoardService.java

```
package com.carshop.controller;
import java.util.List;
public interface BoardService {
   List<BoardDTO> getAllBoardList();
}
```

BoardServiceImpl.java

```
package com.carshop.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
@Service
public class BoardServiceImpl implements BoardService {

@Autowired //DI 의존성 주입 Ioc 제어의 역전
private BoardRepository boardRepository;

@Override
public List<BoardDTO> getAllBoardList() {

    // TODO Auto-generated method stub
    return boardRepository.getAllBoardList();
}
```

BoardRepository.java

```
package com.carshop.controller;
import java.util.List;
public interface BoardRepository {
   List<BoardDTO> getAllBoardList();
}
```

BoardRepositoryImpl.java

```
package com.carshop.controller;
import java.util.ArrayList;
import java.util.List;
```

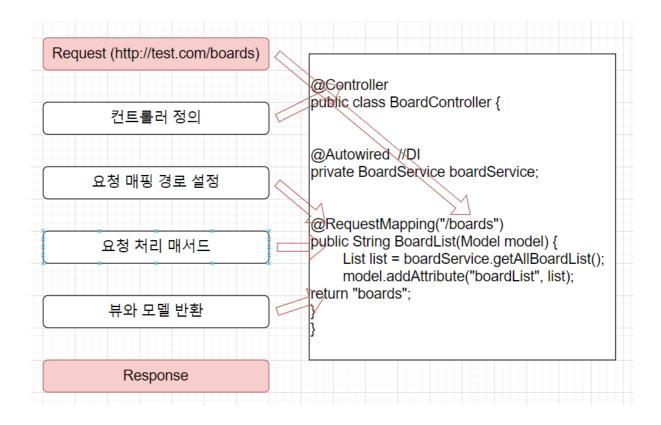
```
import\ org.springframework.stereotype.Repository;\\
@Repository
public class BoardRepositoryImpl implements BoardRepository {
  private List<BoardDTO> listOfBoards = new ArrayList<BoardDTO>();
 public BoardRepositoryImpl() {
   BoardDTO board1 = new BoardDTO("안녕하세요","반갑습니다 ","austin","02/28/2023");
   BoardDTO board2 = new BoardDTO("hi", "hello", "andy", "02/27/2023");
   BoardDTO board3 = new BoardDTO("방가", "신입회원입니다.", "신입", "02/26/2023");
   listOfBoards.add(board1);
   listOfBoards.add(board2);
   listOfBoards.add(board3);
 }
 // private String btitle, bcontent, bauthor, bdate;
 @Override
 public List<BoardDTO> getAllBoardList() {
   return listOfBoards;
}
```

BoardDTO.java

```
package com.carshop.controller;
public class BoardDTO {
  private String btitle, bcontent, bauthor, bdate;
  public String getBtitle() {
    return btitle;
  }
  public void setBtitle(String btitle) {
    this.btitle = btitle;
  }
  public String getBcontent() {
    return bcontent;
  }
  public void setBcontent(String bcontent) {
    this.bcontent = bcontent;
}
```

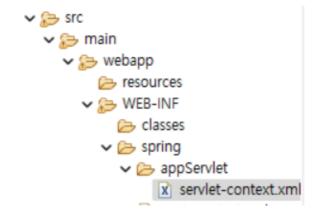
```
public String getBauthor() {
   return bauthor;
  public void setBauthor(String bauthor) {
   this.bauthor = bauthor;
 }
  public String getBdate() {
   return bdate;
 }
  public void setBdate(String bdate) {
   this.bdate = bdate;
 public BoardDTO(String btitle, String bcontent, String bauthor, String bdate) {
   super();
   this.btitle = btitle;
   this.bcontent = bcontent;
   this.bauthor = bauthor;
   this.bdate = bdate;
 }
}
```

Controller



컨트롤러는 매서드를 포한하고 있는 일반적인 자바 클래스가 아니라 브라우저에서 들어온 요청 request 를 처리하는 매서드를 포함하고 있는 특정 자바 클래스이다. 반드시 @Controller 를 선 언하여 이 클래스가 컨트롤러의 역할을 수행하는 클래스다 라는 것을 알려야 한다.

사이트 작성시 규모가 커지게 되면 여러개의 컨트롤러를 사용하게 된다. 따라서 컨트롤러 클래스들을 모두 각가 등록해주어야 하는데 편의상 *를 사용하여 한번에 모두 등록할 수 있다.



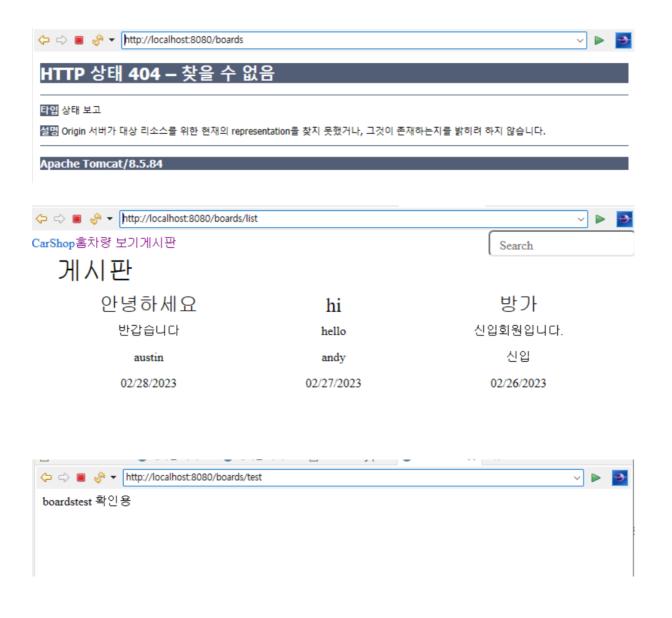
```
<context:component-scan base-package="com.carshop.*" />
```

하나하나 각각 등록해야하는 번거로움을 해결할수 있다.

@RequestMapping on Class

```
@Controller
@RequestMapping(value="/boards", method=RequestMethod.GET)
public class BoardController {
    @Autowired //DI
    private BoardService boardService;
    @RequestMapping("/list")
    public String BoardList(Model model) {
        List<BoardDTO> list = boardService.getAllBoardList();
        model.addAttribute("boardList", list);
        return "boards";
    }
    @RequestMapping("/test")
    public String BoardTest() {
        return "boardstest";
    }
}
```

boards 처리 매서드는 존재하지 않아서 예외 발생



@RequestMapping on Method

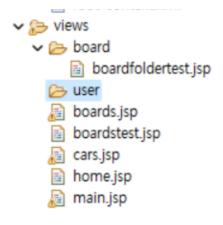
```
@Controller
@RequestMapping(value="/boards", method=RequestMethod.GET)
public class BoardController {
```

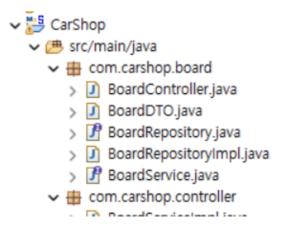
```
@RequestMapping("/list")
public String BoardList(Model model) {
    List<BoardDTO> list = boardService.getAllBoardList();
    model.addAttribute("boardList", list);
    return "boards";
}

@RequestMapping("/test")
public String BoardTest() {
    return "boardstest";
}

@RequestMapping("/boardfoldertest")
public String boardfoldertest() {
    return "boards/boardfoldertest";
}
```

클래스 단위의 매핑과 매서드 단위의 매핑을 섞어서 사용할 경우 (가독성이 좋다.) 뷰도 폴더를 사용하여 분류하게 되면 관리가 더 편하다.





마찬가지로 자바 클래스들도 종류별로 패키지를 분리하여 관리하는 것이 바람직하다.

@GetMapping on Method

GetMapping 은 spring version이 중요하다...

```
9
                                            <version>1.0.0-BUILD-SNAPSHOT</version>
> Maven Dependencies
∨ 🖶 src
                                   109
                                            properties>
  🗸 <table-cell-rows> main
                                                <java-version>1.6</java-version>
                                    11
    🗸 🐎 webapp
                                   12
                                                <org.springframework-version>3.1.1.RELEASE</org.spring</pre>
      resources
                                                <org.aspectj-version>1.6.10</org.aspectj-version>

✓ 

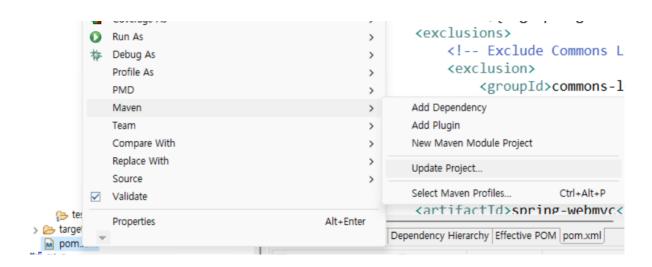
WEB-INF

                                               <org.slf4j-version>1.6.6</org.slf4j-version>
        classes
                                   14
       🗸 🗁 spring
                                   15
                                            </properties>
        16⊜
                                           <dependencies>
            x servlet-context.xml
                                  17
                                               <!-- Spring -->
          x root-context.xml
       189
                                                <dependency>
         🗸 📂 board
                                   19
                                                    <groupId>org.springframework</groupId>
            boardfoldertest.jsp
                                   20
                                                    <artifactId>spring-context</artifactId>
          user
                                   21
                                                    <version>${org.springframework-version}</version>
          boards.isp
                                   22⊝
                                                    <exclusions>
          boardstest.jsp
          cars.jsp
                                   23
                                                        <!-- Exclude Commons Logging in favor of SLF4j
          home.jsp
                                   24⊝
                                                         <exclusion>
          amain.jsp
        x web.xml
                                 Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml
  test
                                 Problems @ Javadoc 🚇 Declaration 😑 Console 🛭 🚜 Servers
> 🎘 target
 m pom.xml
```

```
30
31
31
32     public String boardfoldertest() {
33        return "boards/boardfoldertest";
34     }
35
```

기본 설정된 스프링 버전 3.1.1에서는 getMapping 을 지원하지 않을 뿐더러 앞으로도 많은 부분에서 3버전대에서는 지원하는 않는 기능이 많으므로 다양한 요소에서 예외를 발생 시키게 된다.

따라서 스프링의 기본 버전을 5.x.x 대 이상으로 올려줘야 한다. 최신 버전은 6.x.x 이지만 최신 버전은 다른 버전들과 호환성 면에서 떨어지는 부분이 오히려 더욱 많기 때문에 조금 낮은 버전을 사용하는 것이 바람직하다.



pom.xml 을 수정한 후에는 반드시 Update Project 를 눌러야만 한다. 중요!!!

```
12 @Controller
13 @RequestMapping(value="/board", method=RequestMethod. GET)
14 public class BoardController {
15
       @Autowired //DI
16⊜
17
       private BoardService boardService;
18
19
       @GetMapping("/list")
20⊝
       public String BoardList(Model model) {
21
           List<BoardDTO> list = boardService.getAllBoardList();
22
           model.addAttribute("boardList", list);
23
24
           return "boards";
25
       }
26
27⊝
       @GetMapping("/test")
28
       public String BoardTest() {
29
           return "boardstest";
30
       }
31
       @GetMapping("/boardfoldertest")
32⊝
       public String boardfoldertest() {
33
           return "boards/boardfoldertest":
34
35
       }
36
37 }
```

모델은 사용자의 요청을 처리한 결과 데이터를 관리하고 전달한다. 뷰는 모델을 받아서 브라우저에 출력하는 역할을 한다.

Model 데이터 (또는 객체) 정보를 저장

ModelMap 데이터 (또는 객체) 정보를 저장

ModelAndView 모델 정보 및 뷰 정보를 저장

Model 데이터 (또는 객체) 정보를 저장

```
@Controller
@RequestMapping("/study")
public class ModelExam {

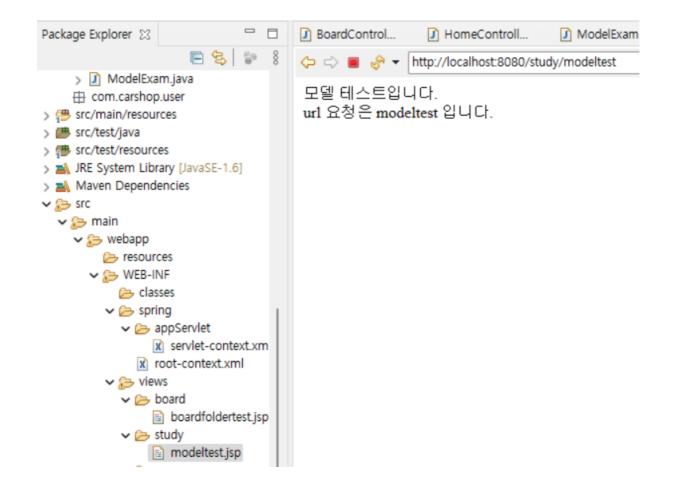
    @GetMapping("/modeltest")
    public String modeltest(Model model) {
        model.addAttribute("data1", "모델 테스트입니다.");
        model.addAttribute("data2", "url 요청은 modeltest 입니다.");
        return "study/modeltest";
    }
}
```

```
□ □ BoardControl... ☑ HomeControll... ☑ ModelExam.java 📓 modeltest.jsp 🛭 🍑 Insert titl...
Package Explorer 💢
                🖹 🕏 📳 🕴 1 <‰ page language="java" contentType="text/html; charset=UTF-8"
                                    pageEncoding="UTF-8"%>
    > 🚺 ModelExam.java
                             3 <!DOCTYPE html>
    > # src/main/resources
                            4⊖ <html>
 > # src/test/java
                             5⊖ <head>
> # src/test/resources
> M JRE System Library [JavaSE-1.6]
                            6 <meta charset="UTF-8">
> Maven Dependencies
                             7 <title>Insert title here</title>

✓ 

Src

                            8 </head>
  🗸 🐎 main
                            9⊜ <body>
    10
       resources
                           11 ${data1}
      classes
                             12 ${data2}
       🗸 🗁 spring
                            13 </body>
         14 </html>
            x servlet-context.xm
          x root-context.xml
        board
            boardfoldertest.jsp
         modeltest.jsp
```



ModelMap 데이터 (또는 객체) 정보를 저장

modelmap 은 model 과 완전히 동일하다. 따라서 model 을 사용하자

```
@Controller
@RequestMapping("/study")
public class ModelmapExam {

@GetMapping("/modelmaptest")
public String modelmap(ModelMap modelMap) {
    modelMap.addAttribute("data1", "모델 맵 테스트입니다.");
    modelMap.addAttribute("data2", "모델과 완전히 동일합니다.");
    return "study/modelmaptest";
}

}
```

ModelAndView 모델 정보 및 뷰 정보를 저장

```
@Controller
@RequestMapping("/study")
public class ModelAndViewExam {

    private BoardService boardService;

    @GetMapping("/modelandview")
    public ModelAndView modelandview() {

        ModelAndView modelAndView = new ModelAndView();
        List<BoardDTO> list = boardService.getAllBoardList();
        modelAndView.addObject("test", list);
        modelAndView.setViewName("study/modelandview");
        return modelAndView;
}
```

위의 두가지 방법 과는 달리 modelandview 는 객체 형태로 리턴한다. 따라서 뷰도 속성 형태로 설정하여 함께 리턴하게 된다.

Spring 한글 처리 방법

web.xml 제일 하단에 추가