|  |
| --- |
| MSF (Mobigen Spring Framework) |
| UI 개발 가이드 |
|  |
| Author: 박세종  Version: 1.6  Version Date: 2016/11/01 |

변경 이력

|  |  |  |  |
| --- | --- | --- | --- |
| 날짜 | 작성자 | 버전 | 변경 내용 |
| 2015/08/03 | 박세종 | 0.1 | 신규 생성 |
| 2015/09/08 | 박세종 | 0.4 | AngularJS 내용 추가 |
| 2016/03/30 | 박세종 | 1.5 | MSF 1.5 버전 업그레이드 |
| 2016/11/01 | 박세종 | 1.6 | MSF 1.6 버전 업그레이드 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

목차

[2. Overview 4](#_Toc447130741)

[2.1. 소개 4](#_Toc447130742)

[2.2. 개발 환경 및 Resource 4](#_Toc447130743)

[3. 코드 작성 규칙 5](#_Toc447130744)

[3.1. Code Conventions 5](#_Toc447130745)

[3.1.1. Java 5](#_Toc447130746)

[3.1.2. JSP & HTML5 & Javascript 5](#_Toc447130747)

[3.2. Naming Conventions 6](#_Toc447130748)

[3.2.1. Java 6](#_Toc447130749)

[3.2.2. HTML 7](#_Toc447130750)

[3.2.3. Javascript 7](#_Toc447130751)

[3.2.4. AngularJS 8](#_Toc447130752)

[3.2.5. URI 8](#_Toc447130753)

[4. Project Package 10](#_Toc447130754)

[4.1. java 10](#_Toc447130755)

[4.2. resources 10](#_Toc447130756)

[4.3. webapp 11](#_Toc447130757)

[5. Programming Guide 13](#_Toc447130758)

[5.1. Back-End 13](#_Toc447130759)

[5.1.1. Controller Class 13](#_Toc447130760)

[5.1.2. Service(Business) Class 17](#_Toc447130761)

[5.1.3. Persistent Class 18](#_Toc447130762)

[5.2. Front-End 21](#_Toc447130763)

[5.2.1. tiles3 21](#_Toc447130764)

[5.2.2. JSP 22](#_Toc447130765)

[5.2.3. RequireJS 23](#_Toc447130766)

[5.2.4. Moment.js 23](#_Toc447130767)

[5.2.5. AngularJS 24](#_Toc447130768)

[5.2.6. Jquery.validation.js 27](#_Toc447130769)

[5.2.7. jquery.layout.js 27](#_Toc447130770)

[5.2.8. jquery.mdi.js 27](#_Toc447130771)

# Overview

## 소개

본 문서는 MSF UI 개발 가이드 문서 입니다.

UI 개발에 필요한 코드 작성 규칙 및 규약에 대한 내용을 포함 하고 있습니다.

## 개발 환경 및 Resource

|  |  |  |
| --- | --- | --- |
| **항목** | **버전정보** | **비고** |
| 지원 Browser | Chrome 43.x 버전 이상 | *http://www.google.co.kr/chrome/browser/desktop/* |
| 화면 해상도 | HD (1920 ⅹ1024) | *https://en.wikipedia.org/wiki/Display\_resolution* |
| Eclipse | LUNA (v4.4x) + STS | *https://www.eclipse.org/* |
| Web/WAS | Tomcat 8.x | *https://tomcat.apache.org/download-70.cgi* |
| Java | JDK8 | *http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html* |
| HTML | HTML5 | *http://www.w3.org/TR/html5/* |
| MSF | v1.6 | *http://www.mobigen.com* |
| Lombok | v1.16.10 | *https://projectlombok.org/* |
| Maven | v4.x | *https://maven.apache.org/* |
| Spring | v4.3.3.RELEASE | *https://spring.io/* |
| Spring Security | v4.1.3.RELEASE | *http://projects.spring.io/spring-security/* |
| JSP | Servlet v3.x | *http://www.oracle.com/technetwork/java/javaee/jsp/index.html* |
| Tiles | v3.x | *https://tiles.apache.org/framework/whats-new.html* |
| RequireJS | v2.3.2 | *http://www.requirejs.org/* |
| Moment.js | v2.15.1 | *http://momentjs.com/* |
| jQuery | v3.1.1 | *https://jquery.com/* |
| jQuery UI | v1.12.1 | *https://jqueryui.com/* |
| jQuery.layout | v1.5 | *http://www.mobigen.com* |
| jQuery.mdi | v1.2 | *http://www.mobigen.com* |
| jQuery.validation | V1.0 | *http://www.mobigen.com* |
| AngularJS | v1.4.3 | *https://angularjs.org/* |
| HightCharts | v4.1.7 (commercial license) | *http://www.highcharts.com/* |
| Angular UI Grid | v3.0.1 | *http://ui-grid.info/* |
| Bootstrap | V3.3.5 | *http://getbootstrap.com/* |

# 코드 작성 규칙

## Code Conventions

### Java

* Eclipse의 Java code formatter 기능을 사용.
* 프로젝트의 /@styleguide/eclipse-java-google-style.xml 파일을 Preferences > Java > Code Style > Formatter > Import 기능으로 적용.
* 코드 작성 후, Ctrl + Shift + f로 Code formatting 처리.
* 파일은 항상 UTF-8로 저장.
* 그 외 내용은 Google java code style([*http://google.github.io/styleguide/javaguide.html*](http://google.github.io/styleguide/javaguide.html)) 참조.

### JSP & HTML5 & Javascript

* Eclipse의 Javascript code formatter / Code Indent 기능을 사용.
* 프로젝트의 /@styleguide/eclipse-javascript-google-style.xml 파일을 Preferences > Java > Code Style > Formatter > Import 기능으로 적용.
* Preferences > Web > HTML Files > Editor 에서 Line width를 100으로 조정.
* 코드 작성 후, Ctrl + Shift + f / Ctrl + i로 Code formatting 처리.
* 파일은 항상 UTF-8로 저장.
* JSP 파일을 생성 시, 반드시 javascript코드는 따로 .js 파일로 만들어서 include.
* HTML tag에는 가급적 이벤트를 선언 하지 않는다.
* 그 외 내용은 아래의 규약을 참조
  + Google javascript code style *(*[*https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml*](https://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml)*)*
  + JQuery style guide *(*[*http://contribute.jquery.org/style-guide/js/*](http://contribute.jquery.org/style-guide/js/)*)*
  + Google angularjs code style *(*[*https://google-styleguide.googlecode.com/svn/trunk/angularjs-google-style.html*](https://google-styleguide.googlecode.com/svn/trunk/angularjs-google-style.html)*)*
  + Code Guide by @mdo *(http://mdo.github.io/code-guide/#html-attribute-order)*

## Naming Conventions

### Java

* 기본적으로 코드 내에서 사용되는 모든 클래스, 함수, 변수에는 특수문자, 숫자를 가급적 허용하지 않음. (상수의 \_ 는 제외)
* 그 외의 기본 규칙은 아래와 같다.

|  |  |  |
| --- | --- | --- |
| **타입** | **명명 규칙** | **예제** |
| Packages | 전부 소문자로 입력 한다. | com.mobigen.troi  com.mobigen.msf |
| Classes | 항상 대문자로 시작 해야 함 (UpperCamelCase)  가급적 축약된 단어는 사용하지 않고, 명사를 이용한다. | class Raster  class ImageSprite; |
| Interfaces | Classes 명명 규칙과 동일 | interface RasterDelegate; |
| Methods | 항상 소문자로 시작 해야 함 (LowerCamelCase)  동사를 이용한다. | run(); runFast(); |
| Local  variables | 항상 소문자로 시작 해야 함(LowerCamelCase)  변수는 의미를 알 수 있는 단어를 조합  단, 반복문의 인덱스나 기타 임시적인 변수는 한단어로 된 짦은 단어 사용 | Int i;  float myWidth; |
| Type  variables | 항상 소문자로 시작 해야 함(LowerCamelCase)  가급적 타입명과 동일하게 작성. | ImageSprite imageSprite; |
| Constants | 항상 모든 문자를 대문자로 | static final int MIN\_WIDTH = 4; |
| Non Constants | 항상 소문자로 시작 해야 함 (LowerCamelCase) | static int ,minWidth = 4; |
| Parameters | 항상 소문자로 시작 해야 함 (LowerCamelCase)  가급적 한글자로 된 단어는 사용하지 말 것. | Int getData(int option); |

* MSF에서 사용되는 method 명 및 sql명은 아래와 같은 규칙을 따른다.

|  |
| --- |
| *prefix+기능명 (camel 표기법)*  *ex) getData, addData* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Controller** | **Service** | **Dao** | **Sql (Mybatis)** |
| **페이지 요청** | view |  |  |  |
| **조회** | get | get | get | get |
| **등록** | add | add | add | add |
| **수정** | update | update | update | update |
| **삭제** | delete | delete | delete | delete |
| **상태** | is | is |  |  |

### HTML

* 기본적으로 모든 html-tag, attribute 는 소문자로 작성.
* HTML의 attribute에 값을 적용 할 때는 double-quote를 사용.

|  |
| --- |
| *<table class="table striped">* |

* Image 태그에는 항상 alt 값을 입력 하고, 가급적 이미지 크기를 지정.

|  |
| --- |
| *<img src="html5.gif" alt="HTML5"****style="width:128px;height:128px***"*>* |

* class명은 Bootstrap에서 지정한 class 외에는 개발자가 절대 임의로 부여 하지 않음.
* Id는 다음과 같은 규칙을 따름.

|  |
| --- |
| *prefix-name*  *ex) btn-submit, chk-use* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **타입** | **prefix** |  | **타입** | **prefix** |
| Edit | edit |  | Button | btn |
| CheckBox | chk |  | RadioButton | rdo |
| Text | txt |  | ComboBox | cb |
| TextArea | list |  | Chart | cht |
| Select | sel |  | Image | img |
| Form | frm |  | ... |  |

* HTML 내의 모든 문자열은 AngularJS의 translate 를 사용 하여 표기 하고, 절대 임의로 입력 하지 않는다.

|  |
| --- |
| *<a class="navbar-brand" href="">****{{ "MENU.TITLE" | translate }}****</a>* |

### Javascript

* 기본적으로 아래와 같은 방식으로 이름을 만든다.

|  |
| --- |
| *functionNamesLikeThis, variableNamesLikeThis, ClassNamesLikeThis, EnumNamesLikeThis,*  *methodNamesLikeThis, CONSTANT\_VALUES\_LIKE\_THIS, foo.namespaceNamesLikeThis.bar, filenameslikethis.js.* |

* 그 외의 기본 규칙은 아래와 같다.

|  |  |  |
| --- | --- | --- |
| **타입** | **명명 규칙** | **예제** |
| Classes | 항상 대문자로 시작 해야 함 (UpperCamelCase)  가급적 축약된 단어는 사용하지 않고, 명사를 이용한다. | *var ClassNamesLikeThis* |
| Methods | 항상 소문자로 시작 해야 함 (LowerCamelCase)  동사를 이용한다. | *methodNamesLikeThis()* |
| Local  variables | 항상 소문자로 시작 해야 함(LowerCamelCase)  변수는 의미를 알 수 있는 단어를 조합  단, 반복문의 인덱스나 기타 임시적인 변수는 한단어로 된 짦은 단어 사용 | var i;  var *variableNamesLikeThis* |
| Constants | 항상 모든 문자를 대문자로 | var *CONSTANT\_VALUES\_LIKE\_THIS = “123”* |
| Parameters | 항상 소문자로 시작 해야 함 (LowerCamelCase)  가급적 한글자로 된 단어는 사용하지 말 것. | Int getData(int option); |

### AngularJS

* 컨트롤러의 이름은 컨트롤러의 이름을 기준으로 지어야 하며(예를 들어 shopping cart, homepage, admin panel), 이름의 끝에는 Ctrl을 붙임. 컨트롤러 이름은 UpperCamelCase를 사용해 작성. (HomePageCtrl, ShoppingCartCtrl, AdminPanelCtrl, etc.).
* 디렉티브 / 필터는 lowerCamelCase를 사용
* 서비스는 UpperCamelCase를 사용
* broadcase / emit 에서 사용하는 이벤트 명은 문자열을 바로 쓰지 않고 상수를 사용. 상수는 \*\*\*\_EVENT형식으로 선언한다.
* 이벤트 핸들러 메소드 명은 on+기능명+Handler의 규칙으로 작성 한다.
* controllAs를 지정 할때의 이름은 controller의 이름을 lowerCamelCase 형식으로 사용 한다.

### URI

* URI에는 대문자는 허용 하지 않음.
* URI 확장자는 아래와 같은 규칙을 따름.

|  |
| --- |
| *http://도메인명:포트/업무분류1/업무분류2/../../기능명.확장자*  *example)*  *http://localhost:8080/board/qna/list.html*  *http://localhost:8080/user/login.do*  [*http://localhost:8080/common/menu/getMenu.json*](http://localhost:8080/common/menu/getMenu.json) |

|  |  |
| --- | --- |
| **구분** | **설명** |
| .html | 일반 view 화면 |
| .json | json 데이터 요청 |
| .do | form 데이터 호출 / 파일 업로드, 다운로드 / 엑셀 업로드, 다운로드 등 |

* REST API는 아래와 같은 규칙을 따름.

# Project Package

## java

**/src/main/java/com/mobigen/~**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **framework** |  |  |  |  |
|  | **excel** |  |  | 엑셀 파일 처리 |
|  | **exception** |  |  | MSF 예외 처리 |
|  | **file** |  |  | 파일 업로드/다운로드 |
|  | **result** |  |  | json 데이터 결과 반환에 대한 정의 |
|  | **security** |  |  | Spring Security를 이용한 web/restful 보안 처리 |
|  | **serializer** |  |  | json 결과 반환 시, 데이터 변환 어노테이션 |
|  | **validator** |  |  | json 호출 시, 입력값 검증 어노테이션 |
|  | **util** |  |  |  |
|  |  | rsa |  | public-key 암호화/복호화 도구 |
| **troi** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## resources

**/src/main/resources**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **mybatis** |  |  |  | mybatis 설정 |
| **spring** |  |  |  | spring 설정 |

**/src/main/resources-local**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **messages** |  |  |  | 예외처리, 인자값 검증 등 각종 사용자 정의 메세지 |
| **properties** |  |  |  | DB설정 및 각종 설정 |

## webapp

**/src/main/webapp**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **resources** |  |  |  |  | Spring-Security 에 적용 받지 않는 자원 경로 |
|  | **js** |  |  |  | 각종 Javascript 파일 |
|  |  | **library** |  |  | Jquery, angularJS등 liblary |
|  |  | **troi** |  |  | Troi 관련 js 파일 |
|  | **css** |  |  |  | 디자인을 위한 css |
|  | **fonts** |  |  |  | 디자인 폰트 |
|  | **images** |  |  |  | 각종 이미지 및 아이콘 |
|  | **messages** |  |  |  | 화면에서 사용되는 문자열 상수 |
|  | **constants** |  |  |  | AngularJS에서 사용되는 상수 |
| **WEB-INF** |  |  |  |  |  |
|  | **tiles** |  |  |  | tiles 3.0 처리 |
|  |  | **index** |  |  | 로그인 전 tiles 구성 |
|  |  | **common** |  |  | 로그인 후 tiles 구성 |
|  | **views** |  |  |  |  |
|  |  | **sample** |  |  | MSF 샘플 페이지 |

\* AngularJS directory 구조는 아래와 같은 형식으로 구성 한다. (예시)

기능 별로 상위 디렉터리를 나누고, 컴포넌트 타입 별로 하위 디렉터리를 구성.

|  |
| --- |
| ├── app  │ ├── app.js  │ ├── common  │ │ ├── controllers  │ │ ├── directives  │ │ ├── filters  │ │ └── services  │ ├── page1  │ │ ├── controllers  │ │ │ ├── FirstCtrl.js  │ │ │ └── SecondCtrl.js  │ │ ├── directives  │ │ │ └── directive1.js  │ │ ├── filters  │ │ │ ├── filter1.js  │ │ │ └── filter2.js  │ │ └── services  │ │ ├── service1.js  │ │ └── service2.js  │ └── page2  │ ├── controllers  │ │ └── ThirdCtrl.js  │ ├── directives  │ │ ├── directive2.js  │ │ └── directive3.js  │ ├── filters  │ │ └── filter3.js  │ └── services  │ └── service3.js  ├── partials  ├── lib  └── test |

# Programming Guide

## Back-End

### Controller Class

* 명명 규칙: 기능명+Controller (*ex:사용자 처리 UserController*)
* Controller 내의 함수는 URL 경로를 반환하거나 service 클래스를 통해 비지니스 로직을 호출 하여 결과를 반환 하는 매핑 역활 에만 국한한다.
* 로그 처리를 위한 logger를 따로 선언할 필요 없이 lombok class를 이용.

|  |
| --- |
| ***Example: View 연결***  *@Slf4j* ***// lombok Class 사용. 클래스에 따로 log 를 설정 할 필요 없이 자동 생성 됨***  *@Controller*  *@RequestMapping(value = "/sample")* ***// 상위 url을 미리 선언***  *public class SampleController {*    *@Autowired*  *private SampleService sampleService;*  *@Value("#{prop['jdbc.username']}") String username;* ***// properties에 미리 선언된 자원을 가져올 수 있다***  *@RequestMapping(value = { "/sample\_file.html" }, method = RequestMethod.GET)*  *public String viewLogin() {*  *log.info(“SAMPLE\_FILE”);* ***// lombok class log 사용***  *return "sample/sample\_file";*  *}*  *}* |

* JSON 형식의 데이터를 반환 해야 하는 경우, 항상 JsonResult 클래스를 사용.
* 별도로 try-catch 문을 선언 하지 않고 대신 반드시 메소드에 throws Exception 문을 추가 하도록 한다.

|  |
| --- |
| ***Example: Json 호출***  *@Slf4j*  *@Controller*  *@RequestMapping("/sample")*  *public class SampleLoginController {*  *@Autowired*  *private SampleLoginService sampleService;*  ***// JSON 형식으로 결과를 반환 시, @ResponseBody 를 선언 하고, 리턴은 JsonResult를 사용 한다.***  *@RequestMapping(value = "/login.json")*  *@ResponseBody*  *public JsonResult login(@ModelAttribute @Valid UserModel login, HttpServletRequest request,*  *HttpServletResponse response) throws Exception {*  *log.info("Login Start:" + login.getUsername() + " / " + login.getPassword());*  *JsonResult js = new JsonResult();*  ***// 처리결과가 좋지 않을때는 GlobalExceptionHandler가 처리 할 수 있도록 예외를 발생 시킨다***  *if (sampleService.login(login, request, response) == 0)*  *throw new Exception("Login Error");*  *log.info("Login Success:" + login.getUsername() + " / " + login.getPassword());*  ***// JSON 결과는 JsonResult의 setData 메소드를 이용한다.***  *js.setData(JsonResult.RESULT\_SUCCESS);*  *return js;*  *}*  *}* |

* + - 1. **Exception**
* 예외가 발생 했거나, 결과가 실패로 끝나서 임의로 예외를 발생 시킨 경우, GlobalExceptionHandler에서 사전에 정의된 메세지 맵의 메세지를 자동으로 JsonResult의 errorMessage에 설정하여 사용자에게 반환 하도록 되어 있다.
* 아래의 규칙에 따라 exception.xml 파일에 사용자 정의 메시지를 추가.

|  |
| --- |
| ***package-path + “.” + class-name + “.” + method-name***  ***Example: 예외처리용 메세지 선언***  *<?xml version="1.0" encoding="UTF-8" standalone="no"?>*  *<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">*  *<properties>*  *<entry key="com.mobigen.framework.exception.GlobalExceptionHandler">시스템 오류가 발생 하였습니다. 자세한 내용은 웹 로그를 확인 해 주시기 바랍니다.</entry>*  *<entry key="com.mobigen.msf.sample.json.SampleJSonController.getDeviceList1">리스트 조회를 실패 했음</entry>*  *<entry key="com.mobigen.msf.sample.login.SampleLoginController.login">로그인을 실패하였습니다. 사용자 정보를 다시 한번 확인 해 주시기 바랍니다.</entry>*  *</properties>* |

* 만일 메세지 자동 처리를 사용하지 못하는 경우 try-catch 문을 선언.

|  |
| --- |
| **Example: 예외 처리 구문 추가**  @RequestMapping(value = "/login.json")  @ResponseBody  public JsonResult login(@ModelAttribute @Valid UserModel login, HttpServletRequest request,  HttpServletResponse response) throws Exception {  log.info("Login Start:" + login.getUsername() + " / " + login.getPassword());  JsonResult js = new JsonResult();    try {  if (sampleService.login(login, request, response) == 0)  throw new Exception("Login Error");  js.setData(JsonResult.RESULT\_SUCCESS);  } catch (Exception e) {  rs.setErrorMessage(e);  }  return js;  }  } |

* + - 1. Validation
* front-end의 모든 요청(Json)에 대한 인자값은 validation 처리를 해야 함.
* back-end validation은 데이터 자체를 체크하고 front-end validation은 입력 자체를 통제.
* back-end Validation은 JSR 303 Bean Validation 방식으로 처리하며 기본적으로는 Spring에서 제공하는 Validation 클래스를 이용.

|  |
| --- |
| ***TestController***  *@RequestMapping(value = { "test.json" }, method = RequestMethod.POST)*  *@ResponseBody*  *public ResultSet testModel(****@Valid*** *@RequestBody Test test) {*  *ResultSet rs = new ResultSet();*  *return rs;*  *}*  ***Test.java***  *public class Test{*  ***@Size(min=1, max=5)***  *private String data1;*    ***@NotEmpty***  *private String data2;*  *public String getData1() {*  *return data1;*  *}*  *...* |

* validation 오류 발생 시, 사전에 정의된 메세지 맵의 메세지를 자동으로 JsonResult의 errorMessage에 설정하여 사용자에게 반환 하도록 되어 있음. 아래의 규칙에 따라 validation.xml 파일에 사용자 정의 메세지를 추가.

|  |
| --- |
| ***code + "." + object name + "." + field***  *ex) Test.java의 property data1에 Size Validation 을 걸었다*  *Size.test.data1 = “{0}의 길이는 {1}보다 작고 {2} 보다 커야 함”* |

* JSR-303 형식 외에 사용자 정의 validation 클래스를 제공.

|  |  |
| --- | --- |
| **클래스명** | **형식** |
| UserAccount | 로그인 시, 계정/패스워드에 대한 validation 처리 |
| UserRSAAccount | 로그인 시, 암호화된 계정/패스워드에 대한 validation 처리 |

* + - 1. Serialize
* back-end 에서 front-end로 JSON형식을 반환 하는 경우 , 필요 시 @JsonSerialize를 이용하여 데이터를 자동으로 변환 할 수 있음.

|  |
| --- |
| *public class SampleJsonModel {*  *@JsonSerialize(using = NumberSerializer.class)*  *private String number;*  *@JsonSerialize(using = ByteToMegaByteSerializer.class)*  *private String megaBytes;*  *...*  *}* |

* 다음과 같은 사용자 정의 Serialize 클래스를 제공 .

|  |  |
| --- | --- |
| **클래스명** | **형식** |
| ByteToMegaByteSerializer | 바이트를 메가바이트로 변환 ex) 1048576 -> 1 |
| NumberSerializer | 숫자를 세 자리 묶음으로 처리 ex) 1048576 -> 1,048,576 |

### Service(Business) Class

* 명명 규칙: 기능명+Service (ex: 사용자 처리 UserService)
* Controller에서 요청한 비지니스 로직을 처리하고 반환.
* 일반적으로는 인터페이스를 먼저 만들고 그에 따른 구현 클래스를 따로 작성 하지만, 1:1 사용되는 클래스의 경우 인터페이스는 생략.
* 만일 사용자 정보를 다루는 클래스 처럼 여러 곳에서 사용 되는 기능은 인터페이스를 반드시 작성.
* Service에서 다루는 모든 메소드의 exception은 바깥으로 던져서 Controller가 처리 할 수 있도록 함.

|  |
| --- |
| *@Service*  *public class UserService {*  *@Autowired*  *private UserMapper userMapper;*    ***// exception은 controller에서 처리 할 수 있도록 throws Exception 구문을 추가한다***  *public List<MenuEntity> getMenuList(String userid) throws Exception*  *{*  ***// 만일 DAO에 파라메터를 hash로 넘겨야 한다면 service레이어에서 만들어 처리 한다.***  ***// control layer에서 만들지 않는다.***  *HashMap<String> param = new HashMap<String>();*  *param.put("userid", userid);*    *List<MenuEntity> list = userMapper.getMenuList(param);*  *return list;*  *}*  *}* |

### Persistent Class

* + - 1. Dao Interface
* 명명 규칙: 기능명+Mapper (ex: 사용자 처리 UserMapper)
* Service에서 요청한 쿼리를 연결하여 처리하고 반환.
* DAO는 따로 구현체를 만들지 않고 interface만 정의 하여 MyBatis의 클래스 자동 생성 기능을 이용.
* Interface 정의 시 모든 메소드에 throws exeption을 선언 하여 dao->service->controller 의 예외 처리 프로세스가 성립되도록 함.

|  |
| --- |
| ***// Repository value 값은 클래스명을 사용한다 (소문자로 시작)***  *@Repository(value="****customerHighMainMapper****")*  *public interface* ***CustomerHighMainMapper*** *{*  ***// 반환값은 반드시 DB의 필드와 대응되는 entity 클래스를 선언한다.***  ***// 가급적 반환값으로 hash를 사용하지 않는다***  *public CustUseInfoEntity* ***getCustUseInfo****(HashMap<String,Object> aParam) throws Exception;*  ***// 클래스명은 Mybatis에 선언한 id 값과 정확히 일치 해야 한다***  *public CntrDiffCommentEntity getCntrDiffComment(HashMap<String,Object> aParam) throws Exception;*  *public List<WeatherTimeEntity> getWeatherTime(HashMap<String,Object> aParam) throws Exception;*  *}* |

* + - 1. sqlmap
* 명명 규칙: Dao Interface의 클래스명과 동일 하게 작성
* id 는 Dao Interface의 메소드명과 일치하게 1:1로 작성

|  |
| --- |
| *<!DOCTYPE mapper*  *PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"*  *"http://mybatis.org/dtd/mybatis-3-mapper.dtd">*    *<mapper namespace="com.mobigen.kepri.customer.high.main.****CustomerHighMainMapper****">*    *<select id="****getCustUseInfo****" resultType="com.mobigen.kepri.customer.high.main.entity.CustUseInfoEntity">*  *SELECT*  *A.\**  *, B.mr\_dd as mr\_dd*  *, '1' as SELBILL\_CD*  ***...*** |

* resultType의 경로가 너무 길 경우, TypeAlias를 이용한다. TypeAlias는 sqlMapConfig.xml 에 모두 등록.
* 작성된 mapper는 sqlMapConfig.xml 에 등록.

|  |
| --- |
| *<!-- In SQL Mapping XML file -->*  *<select id="selectUsers" resultType="UserModel">*  *select id, username, hashedPassword*  *from some\_table*  *where id = #{id}*  *</select>*  *...*  *<!-- sqlMapConfig.xml -->*  *<configuration>*  *<settings>*  *<setting name="cacheEnabled" value="true" />*  *<setting name="lazyLoadingEnabled" value="false" />*  *<setting name="multipleResultSetsEnabled" value="true" />*  *...*  *</settings>*  *<typeAliases>*  *<typeAlias type="com.mobigen.msf.sample.login.model.UserModel" alias="UserModel" />*  *</typeAliases>*  *<mappers>*  *<mapper resource="mybatis/test/test-mapper.xml" />*  ***<mapper resource="mybatis/common/common-mapper.xml" />***  *...* |

* sql mapper는 패키지 구조와 동일하게 계층 구조로 경로를 만들어 위치 시키도록 함.
  + - 1. Model Class
* Controller에서 Form 혹은 Json 형태의 인자를 받으면 반드시 Model 객체를 생성.
* DB Table의 데이터를 반환 시 반드시 Model 객체를 생성.
* 각각의 객체는 아래와 같이 패키지를 생성.

|  |
| --- |
| *ex) com.mobigen.tbic.test.model / com.mobigen.tbic.test.model* |

* 명명 규칙: *클래스명+Model (ex: TestModel.java*
* Model에는 별도로 getter/setter를 사용하지 않고, lombok 클래스를 이용하여 @Data 를 대신 사용.

|  |
| --- |
| *@Data*  *public class SampleJsonModel {*  *private String number;*  *private String megaBytes;*  *private String userName;*  *}* |

* + - 1. Transaction
* 데이터 처리 시, transaction이 필요할 때는 mapper를 호출하는 비지니스 레이어에서 @Transactional 어노테이션을 이용하여 처리.

|  |
| --- |
| *@Transactional*  *public int addData(String value) throws Exception {*  *…*  *}* |

## Front-End

### tiles3

* 모든 화면은 tiles로 구성.

|  |
| --- |
| ***tiles-definitions.xml***  *<!DOCTYPE tiles-definitions PUBLIC*  *"-//Apache Software Foundation//DTD Tiles Configuration 3.0//EN"*  *"http://tiles.apache.org/dtds/tiles-config\_3\_0.dtd">*  *<tiles-definitions>*  *<definition name="defaultTemplate" template="/WEB-INF/tiles/default/serviceLayout.jsp">*  *<put-attribute name="title" value="" />*  *<put-attribute name="meta" value="/WEB-INF/tiles/default/meta.jsp" />*  *<put-attribute name="header" value="/WEB-INF/tiles/default/header.jsp" />*  *<put-attribute name="footer" value="/WEB-INF/tiles/default/footer.jsp" />*  *</definition>*  *<definition name="indexTemplate" template="/WEB-INF/tiles/index/serviceLayout.jsp">*  *<put-attribute name="title" value="" />*  *<put-attribute name="meta" value="/WEB-INF/tiles/index/meta.jsp" />*  *<put-attribute name="header" value="/WEB-INF/tiles/index/header.jsp" />*  *<put-attribute name="footer" value="/WEB-INF/tiles/index/footer.jsp" />*  *</definition>*  *</tiles-definitions>*  ***serviceLayout.jsp***  *<%@ taglib uri="http://tiles.apache.org/tags-tiles" prefix="tiles"%>*  *<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>*  *<!DOCTYPE html>*  *<html xmlns="http://www.w3.org/1999/xhtml">*  *<head>*  *<title><tiles:insertAttribute name="title" /></title>*  *<tiles:insertAttribute name="meta" />*  *</head>*  *<body>*  ***<div id="body">***  ***<tiles:insertAttribute name="body" />***  ***</div>***  *<div id="footer">*  *<tiles:insertAttribute name="footer" />*  *</div>*  *</body>*  *</html>*  ***test.jsp***  *<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>*  *<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>*  *<%@ taglib prefix="tiles" uri="http://tiles.apache.org/tags-tiles" %>*  *<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>*  ***<tiles:insertDefinition name="defaultTemplate">***  ***<tiles:putAttribute name="body">***  *<h1> Hello!! World </h1>*  *</tiles:putAttribute>*  *</tiles:insertDefinition>* |

### JSP

* jsp는 기본적으로 아래와 같은 tablib를 선언.
* c: jstl core
* sec: spring security (세션의 사용자 정보 등을 출력 할때 사용. 필요 시 선언)
* tiles: tiles 선언

|  |
| --- |
| *<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>*  *<%@ taglib prefix="sec" uri="http://www.springframework.org/security/tags" %>*  *<%@ taglib prefix="tiles" uri="http://tiles.apache.org/tags-tiles" %>*  *<%@ page pageEncoding="UTF-8" contentType="text/html; charset=UTF-8"%>* |

* html 코드는 html5 를 기준으로 함.
* html 코드에 사용되는 내부 resource에 접근하는 모든 url은 c:url을 사용.

|  |
| --- |
| *…*  *<script src="****<c:url value='/resources/js/jquery/jquery-ui.min.js'/>****"></script>*  *…*  *function onUsePatternClickHandler ( )*  *{*  *$("#modal\_main").load ( "****<c:url value='/customer/low/analysis/useAnalyPatternMain.html'/>****", ajaxErrorHander);*  *}* |

* 세션 정보가 필요할 때는 spring security를 이용.

|  |
| --- |
| *<c:set var="userType" scope="page">*  *<sec:authentication property='principal.userType' />*  *</c:set>*  *…*  *<span class="fl banner b01"><a rel="leanModal" href="#modal\_main"onclick="onUsePatternClick)">HELP</a></span>*  *<c:choose>*  *<c:when test="****${userType == '210'}****">*  *<span class="fr banner b02\_h"><a rel="leanModal" href="#modal\_main" onclick="onCompareRatesClickHandler()">다른요금제와 비교하기</a></span>*  *</c:when>*  *...* |

### RequireJS

* 각 JS 파일간의 dependency 처리를 위해 RequireJS를 사용
* 새로운 javascript 파일을 include 하려면 requirejs의 설정 파일에 키와 경로를 등록. 자세한 정보는 requires 사이트를 참조

|  |
| --- |
| *<script type="text/javascript" src="require.js"></script>*  *<script type="text/javascript">*  *require(["http://code.jquery.com/jquery-1.6.2.min.js"], function() {*  *alert($().jquery);*  *});*  *</script>* |

### Moment.js

* 모든 시간에 관련된 변수는 moment.js를 사용 하여 구현한다. (Date 는 가급적 사용하지 않는다.)
* 현재 지역의 정확한 값 (GMT)을 얻기 위해서는 항상 local() 로 값을 처리 해야 한다.

|  |
| --- |
| *$scope.currentDate = moment(dt.getTime()).add(6, 's');* |

### AngularJS

* + - 1. module
* 모듈 선언 시 아래와 같은 방식으로 사용 한다.

|  |
| --- |
| ***angular***  *.module('app', [*  *'ngAnimate',*  *'ngRoute',*  *'app.shared',*  *'app.dashboard'*  *]);*  ***angular***  *.module('app')*  *.controller('SomeController',* ***SomeController****);*  *function* ***SomeController****() { }* |

* + - 1. Controller
* Controller 코드는 아래와 같이 requireJS 형식을 기본으로 한다.

|  |
| --- |
| *define(["app"], function(app){*  *app.controller("SearchCtrl", ["$rootScope", function($rootScope) {*  *"use strict";*    *}]);*  *});* |

* Controller 사용 시 controllerAs를 사용 한다

|  |
| --- |
| *<!-- recommended -->*  *<div ng-controller="Customer as customer">*  *{{ customer.name }}*  *</div>*  *function Customer() {*  *this.name = {};*  *this.sendMessage = function() { };*  *}*  *function Customer() {*  *var vm = this;*  *vm.name = {};*  *vm.sendMessage = function() { };*  *}*  *<input ng-model="vm.title"/>* |

* controller에는 화면을 처리 하는 코드만을 선언 하고, Business-logic은 server를 사용 한다(Spring을 생각하라ㅋ)
  + - 1. Server & Factory
* factory 는 common-module 작성 시 사용 한다.
* service 는 business-module 작성 시 사용 한다.
  + - 1. Directives
* 항상 custom element 와 custom attribute를 사용 한다. ({ restrict: 'EA' })

|  |
| --- |
| *<my-directive></my-directive>*  *<div my-directive></div>* |

* HTML-template 코드를 생성 할 때는 String + 를 하지 말고.. Array.join(‘’)을 사용 한다. (성능차이가 많이난다)
  + - 1. Filters
* 전역 필터는 angular.filter() 만을 사용하여 선언 한다.

|  |
| --- |
| *function startsWithLetterA () {*  *return function (items) {*  *return items.filter(function (item) {*  *return /^a/i.test(item.name);*  *});*  *};*  *}*  *angular*  *.module('app')*  *.filter('startsWithLetterA', startsWithLetterA);* |

* + - 1. ETC
* 최초의 초기화 이후 내용 변경이 필요 없는 변수는 bindonce를 사용

|  |
| --- |
| *<p>*  *{{ ::vm.user }}*  *</p>* |

* 다음과 같은 사용을 권장함

|  |
| --- |
| *setTimeout 대신 $timeout*  *setInterval 대신 $interval*  *window 대신 $window*  *document 대신 $document*  *$.ajax 대신 $http* |

* 이벤트는 선언 후 반드시 제거 한다

|  |
| --- |
| *var unbind = [*  *$rootScope.$on('customEvent1'[, callback]),*  *$rootScope.$on('customEvent2'[, callback]),*  *$rootScope.$on('customEvent3'[, callback])*  *];*  *$scope.$on('$destroy', function () {*  *unbind.forEach(function (fn) {*  *fn();*  *});*  *});* |

* 코드 블록을 이용하여 코드별 구역을 나누어 처리 한다.

|  |
| --- |
| *// property*  *// event-handler*  *// method* |

### Jquery.validation.js

* 값 자체는 Back-End에서 처리하므로, Front-End에서는 입력 제한 방식을 사용.
* 입력 제한은 jquery.validation.js를 사용.

|  |
| --- |
|  |

### jquery.layout.js

* HTML 화면 구성 시, Flex 스타일로 코딩 가능하게 해주는 플러그인.

|  |
| --- |
| *<script>*  *$(document).ready(function() {*  *$("#content").layout({*  *gap : 5*  *});*  *});*  *</script>*  *<div id="content" layout="vertical" width="100%" height="100%" vertical-align="middle" horizontal-align="center">*  *<h1 width="50%" height="80px">로그인 테스트</h1>*  *<div width="50%" height="180px" layout="vertical" vertical-align="middle" horizontal-align="left" gap="10px">*  *<div width="100%" height="100%" layout="horizontal" vertical-align="middle">*  *<span width="80px">아이디</span>*  *<input id="username" type="text" width="100%"></input>*  *</div>*  *<div width="100%" height="100%" layout="horizontal" vertical-align="middle">*  *<span width="80px">패스워드</span>*  *<input id="userpass" type="password" width="100%"></input>*  *</div>*  *<button class="btn" onclick="login()">로그인</button>*  *</div>*  *</div>* |

### jquery.mdi.js

* HTML 화면 구성 시, MDI로 화면을 구현 가능하게 하는 플러그인.
* jquery.layout.js / jquery.ui 가 필수적으로 필요.

|  |
| --- |
| *<script>*  *$(document).ready(function() {*  *var layout = $("#content").layout();*  *var mdi = $("#mdiContainer").mdi();*  *$(".add-window").click(function() {*  *mdi.addWindowByHtml(this.value, this.value, "<p>Hello! World!!" + this.value, 800, 600);*  *});*  *});*  *</script>*  *<div id="content" layout="vertical" width="100%" height="100%">*  *<div id="mdiContainer" width="100%" height="100%" gap="0px"></div>*  *<div id="example" layout="horizontal" width="100%" height="60px" vertical-align="middle">*  *<input type="button" class='add-window' value="창1번"></input>*  *<input type="button" class='add-window' value="창2번"></input>*  *<input type="button" class='add-window' value="창3번"></input>*  *<input type="button" class='add-window' value="창4번"></input>*  *</div>*  *</div>* |