Smart Green Platform 개발

개발표준

문서번호 입력

Ver. 1.0

관리부서 : 통합팀

개 정 이 력

버전	작성일	변경내용[1]	작성자	승인자
1.0.0	2013-11-25	기존 개발표준 v1.0.0 에서 이관 및 플랫폼 기반 개발 기준으로 내용 수정	최윤종	

[1] 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.)

목 차

1. 개요

개발시 준수해야 할 표준에 대해 가이드한다. 본 문서에 제시된 내용 외에 별도 제공되는 문서들을 참고하도록 한다.

- Java 코딩 가이드
- 웹 개발 보안 가이드
- 웹 표준 및 접근성 가이드

2. 업무코드

모델링 담당자가 제공하는 모델링 가이드에 따른다.

3. 폴더구조

3.1 로컬/개발/운영 환경

개발자 로컬 PC 는 개발자 개발환경 구성가이드에 따르며, 개발/운영 서버의 구조는 설치되는 경로에 따른다.

3.2 모듈 기본구조

폴더 구조				용도
{프로젝트 모듈 경로}	/build			빌드파일 생성 폴더
	/log			로그파일 생성 폴더
	/src			java 소스
	/resources	/META-INF	/config	Spring xml 설정 파일
			/properties	프로퍼티 파일
			/sqlmap_db2	iBATIS SQL Map xml 파일(DB2)
			/sqlmap_mssql	iBATIS SQL Map xml 파일(MSSQL)

폴더 구조				용도
			/sqlmap_oracle	iBATIS SQL Map xml 파일(Oracle)
			/sqlmap_postgre	iBATIS SQL Map xml 파일(Postgre)
			/validation	Validation Rule 설정 파일
	/WebContent	/css		css 파일
		/images		image 파일
		/js		javascript 파일
		/swf		swf 파일
		/download		사용자 다운로드용 파일(apk 등)
		/jspf	/tiles	Tiles 레이아웃 템플릿
			/include	included jsp 파일
		/plugin	/xxx	웹 클라이언트, 플러그인
		/WEB-INF	/config	Spring MVC xml 설정 파일
			/lib	라이브러리 파일
			/tld	태그 라이브러리 설정 파일
			/views	업무별 jsp 파일

3.3 Java 패키지 구조

java 패키지 구조는 기본적으로 모델설계 구조를 따르며, 업무 코드명과 컴포넌트 구분에 따라 아래와 같다.
(SGP 서비스 개발에서는 공통 패키지 구조로서 'com.lgcns.sgs'를 사용한다.)
[예시]

[공통 패키지].[업무 코드명].[1 레벨 업무명].[2 레벨 업무명].[컴포넌트 구분].*.java
(컴포넌트 구분: dao, model, service, controller, util 등)

com.lgcns.sgs.dzh.bizcommon.cmmn.dao.XXXDao.java
com.lgcns.sgs.dzh.bizcommon.XXX.controller.XXXController.java
com.lgcns.sgs.dzh.bizcommon.XXX.service.XXXService.java

3.4 Spring xml 파일

Spring xml 파일은 config 폴더에서 역할별로 구분되는 개별 파일을 만든다. 하위에 별도로 framework 폴더를 작성하여 framework 의 bean 설정들을 관리한다.

3.5 프로퍼티 파일

프로퍼티 파일은 properties 폴더에 업무별로 파일을 분리하여 작성하며, 별도의 하위 폴더를 만들지 않는다.

[예시]

resources/META-INF/properties/[업무 코드명].properties

3.6 iBATIS SQL Map xml 파일

iBATIS SQL Map xml 파일은 sqlmap_[DBMS 타입] 폴더에서 업무에 따라 아래와 같이 하위 폴더를 작성하여 관리한다.

[예시]

resources/META-INF/sqlmap_oracle/[업무 코드명] /[1 레벨 업무명]/[EBC 명]Query.xml

3.7 css 파일

css 파일은 css 폴더를 사용하고, 업무에 따라 하위 폴더를 생성한 후 작성한다.

[예시]

WebContent/css/[업무 코드명] /*.css

3.8 image 파일

image 파일 경로는 images 폴더를 사용하고, 업무에 따라 하위 폴더를 생성한 후 작성한다. 업무코드 하위 폴더 생성은 업무별 구성에 따른다.

[예시]

WebContent/images/[업무 코드명] /[image 파일]

3.9 include jsp 파일

include jsp 파일 경로는 jspf/include 폴더를 사용하고 별도의 하위 폴더를 만들지 않으며 용도별로 파일만 분리한다. 파일 확장자는 jspf 를 사용한다.

3.10 javascript 파일

javascript 파일은 js 폴더에 업무에 따라 하위 폴더를 생성한 후, 작성한다. 업무코드 하위 폴더 생성은 업무별 구성에 따른다.

[예시]

WebContent/js/[업무 코드명] /*.js

3.11 Flex 용 swf 파일

Flex 로 개발된 swf 파일은 swf 폴더에 업무에 따라 하위 폴더를 생성한 후, 작성한다. 업무코드 하위 폴더 생성은 업무별 구성에 따른다.

[예시]

WebContent/swf/[업무 코드명] /*. swf

3.12 Spring MVC xml 파일

Spring MVC xml 파일은 WEB-INF/config 폴더에 용도별로 파일만 분리하여 작성한다.

3.13 업무별 jsp 파일

업무별 jsp 파일은 WEB-INF/views 폴더에 업무에 따라 하위 폴더를 생성한 후, 작성한다.

[예시]

WEB-INF/views/[업무 코드명]/[1 레벨 업무명]/[2 레벨 업무명]/*.jsp

4. 명명규칙

java 파일명을 제외하고 모든 파일 이름은 소문자로 시작한다. 두 개 이상의 단어가 연결될 경우에는 '_' 를 사용하지 말고, Camel Case 기법을 사용한다. 즉, 두 번째 단어부터 각 단어의 첫 글자만 대문자로 작성한다. 폴더 구분 없이 업무별로 파일만 구분되는 파일의 경우는 업무명을 파일명으로 사용하도록 한다.

4.1 Java 파일명

java 파일명은 첫글자를 반드시 영문 대문자로 시작하며, '[의미있는 이름][컴포넌트 postfix]'의 형태로 작성한다. 컴포넌트 postfix 는 Dao, Service, Controller, Util 등과 같이 특성이 반영된 문자열을 의미한다.

Java 구분	컴포넌트 구분	패키지명	컴포넌트 Postfix
Controller	Controller	controller	Controller
Action	Action	service.action	Action
Service	PBI	service	Service
	PBC	service	ServiceImpl
	CBI	cmmn.util	Util
	CBC	cmmn.util	UtilImpl
	EBI	cmmn.dao	Dao
	EBC	cmmn.dao	DaoImpl
Model	Model	cmmn.model	Model

4.2 Java 메소드명

java 메소드명은 의미적인 식별이 용이하도록 '[의미있는 동사][명사]'의 형태로 작성한다.

4.3 Action 명

Action 클래스의 어노테이션으로 선언되는 Action 명은 내용이 잘 드러날 수 있도록 '[업무 코드명].[1 레벨 업무명].[2 레벨 업무명].[의미있는 이름]'으로 작성한다.
예) dzh.bizcommon.OO.OOOAction

4.4 iBATIS SQL Map xml 파일명

SQL Map xml 파일은 2 레벨 업무별로 파일을 생성하며, 파일명은 '[EBC 명]Query.xml'로 한다.

4.5 iBATIS SQL Map namespace 및 query id

SQL Map xml 파일의 namespace 명은 '[업무 코드명].[파일명]'으로 작성한다. query id 는 '[동사(insert/update/delete/select)][명사]'로 작성한다.

4.6 jsp 파일명

jsp 파일명은 화면 구성이 하나의 파일로 이뤄지는 경우에는 해당 업무를 표현할 수 있는 의미 있는 이름으로 구성하고, 여러 개의 파일로 나뉘어지면 List/Edit/Page 등의 특성이 반영될 수 있도록 postfix 를 붙이도록 한다.

4.7 javascript 파일명

javascript 파일명은 업무에 따라 의미있는 이름으로 작성한다.

4.8 javascript 함수명

javascript 함수명은 의미적인 식별이 용이하도록 '[의미있는 동사][명사]'의 형태로 작성한다.

4.9 swf 파일명

swf 파일명은 업무에 따라 의미있는 이름으로 작성한다.

5. 설정 표준

5.1 프로퍼티 설정

프로퍼티 설정은 업무별로 관리하는 것을 표준으로 하며, 프로퍼티 값의 중복 가능성을 제거하기 위해 업무 코드명을 명시한다. 프로퍼티명은 '_'없이 Camel Case 로 표기하고, 의미구분이 필요한 경우 '.'으로 구분한다.

[예시]

```
[업무 코드명].[1 레벨 업무명].[프로퍼티명]=[프로퍼티값]
dzh.codemanage.icon.noImage=/images/dzh/noimage.jpg
```

5.2 Bean 설정

Bean 설정의 변경 및 추가는 기존 Spring xml 파일을 수정하여 적용하는 것을 표준으로 한다. Bean id 는 인터페이스명의 첫글자를 소문자로 바꾼 것을 기본으로 하며, 중복이나 의미구분이 필요한 경우 인터페이스명 앞에 추가로 명시한다.

[예시]

```
(인터페이스명이 MultipartResolver 인 경우)
<bean id="multipartResolver"
 class="com.lgcns.sgs.dzh.security.web.xecure.resolver.XecureDecryptMultipartResolver"
 />
```

업무별로 사용범위가 한정되는 Bean id 는 업무 코드명을 명시한다. 단, SGP 프레임워크 컴포넌트에서 별도의 명명규칙을 가지는 경우 컴포넌트 개발가이드의 명명규칙을 따른다.

[예시]

```
[업무 코드명].[의미구분][인터페이스명]
<bean id="dzh.execHmiScheduleTrigger"
 class="org.springframework.scheduling.quartz.CronTriggerBean"
 />
```

6. Java 코딩 표준

6.1 기본사항

java 코딩에 대한 전반적인 사항은 별도로 제공되는 LG CNS Java 코딩 가이드에 따른다. 단, comment 및 명명규칙에 대한 표준은 본 가이드에서 제공하는 프로젝트 표준에 따른다.

6.2 comment

6.2.1 파일 comment

SGP 내부에서 개발하는 소스에서는 패키지명 선언 전에 다음과 같은 comment 를 공통으로 작성한다. 라이센스 부분은 수정 없이 Copy & Paste 하도록 하며, 수정 이력부분은 코드의 주요 수정사항 발생시마다 추가하도록 한다. 성명 부분은 개발 담당자명을 기술하도록 한다.

```
/*                                                                                    *
*                  S M A R T    G R E E N    P L A T F O R M    S Y S T E M          *
*                                                                                    *
*                                    LG CNS Co., Ltd.                                *
*                                  PUBLIC/SOC BIZ DIVISION                           *
*                                                                                    *
*                  All rights reserved.  No part of this publication may be          *
*                  reproduced,  stored in a retrieval system  or transmitted         *
*                  in any form or by any means  -  electronic,  mechanical,          *
*                  photocopying, recording, or otherwise, without the prior          *
*                  written permission of LG CNS Co., Ltd.                            *
* ***********************************************************************************
*     수정이력 :
*      날 짜          성 명                    수정  내용
*  -------------    -----------    ---------------------------------------------------
*    2011/01/01       홍 길 동          Initial Release
```

```
*    2011/02/01       홍 길 동          M0001 - Xxxxxx를 수정함.
 ********************************************************************************************/
```

6.2.2 클래스 comment

클래스 혹은 인터페이스 선언 전에는 해당 클래스에 대한 간략한 설명과 정보를 다음과 같은 양식으로 작성한다.

```
/**
 * <pre>
 * 기능 : Pattern1-2 SingleEdit sample Controller
 * 설명 : SingleEdit기능 구현시 참고할 수 있는 Controller의 sample을 보여준다
 *
 * References :  설계서 SGP-XXX-XX-01
 *
 * Special Logic : NONE
 * </pre>
 */
```

6.2.3 메소드 comment

메소드의 선언 전에는 해당 메소드에 대한 간략한 설명과 파라미터 및 리턴값을 다음과 같은 양식으로 작성한다. 설명은 해당 메소드의 처리 절차를 기술한다.

```
    /**
     * 수정을 위한 employee data를 조회하고 화면에 리턴함.<br />
     * - employee data 조회<br />
     * - 조회값의 null여부 판별하여 view 결정하여 리턴
     *
     * @param searchEmpNum  조회 EmpNum값
     * @return 결과 view url
     */
```

7. jsp 코딩 표준

jsp 소스는 comment, page directive, scriptlet 및 tag library 등으로 구성된다.

7.1 기본사항

jsp 는 scriptlet 코드가 많을수록 가독성이 떨어져 개발 및 유지보수가 어려워진다. 그러므로 jsp 에서는 데이터의 구조와 내용 부분만 담당하도록 하고 처리 로직은 다른 컴포넌트에서 구현한다.

- 모든 태그는 영문 소문자로 작성한다.
- 들여쓰기는 4 spaces 로 한다. 단, html 코드와의 가독성을 위해 jsp 스크립틀릿 및 태그 라이브러리는 상위 html 태그 요소와 동일한 컬럼에 들여쓰기 한다.
- scriptlet 태그(<% ... %>)는 첫 컬럼에서 시작해서 첫 컬럼에서 끝나도록 한다. 단, 값을 출력하기 위한 '<%=...%>'는 예외로 한다.
- 값을 출력할 때는 '<% out.print(varName) %>' 대신 '<%=varName %>'을 사용한다.
- jsp 에는 내장 객체로 request, response, out 등이 있으므로 이에 관련된 정보를 얻기 위해 필요없는 코드를 작성하는 일은 지양한다.

7.2 jsp comment

jsp 소스에서만 볼 수 있고, 클라이언트 웹 브라우저의 '소스보기' 에서는 확인되지 않는 주석이다. 소스에 대한 정보, 로직 설명 등 사용자에게 보여서는 안되는 주석을 이러한 형태로 작성한다. jsp comment(<%-- ... --%>)와 scriptlet 안의 일반적인 java 주석(/*...*/, //...)이 여기에 속한다.

SGP 내부에서 개발하는 소스에서는 파일의 맨 처음에 jsp comment 로 다음과 같이 파일 주석을 작성한다. 성명 부분은 개발 담당자명을 기술한다.

```
<%--------------------------------------------------------------------------------------------
                    S M A R T    G R E E N    P L A T F O R M    S Y S T E M

                                    LG CNS Co., Ltd.
                                PUBLIC/SOC BIZ DIVISION

                    All rights reserved.  No part of this publication may be
                    reproduced,  stored in a retrieval system  or transmitted
                    in any form or by any means  -  electronic,  mechanical,
                    photocopying, recording, or otherwise, without the prior
                    written permission of LG CNS Co., Ltd.
--------------------------------------------------------------------------------------------
      파일명 : form.jsp
```

```
      설  명 : xxx정보 저장을 위해 form 입력하는 화면
      수정이력 :
      날  짜          성 명                        수정  내용
 ------------    ----------    ------------------------------------------------------------
    2011/01/01       홍 길 동        Initial Release
    2011/02/01       홍 길 동        M0001 - Xxxxxx를 수정함.
-------------------------------------------------------------------------------------------%>
```

scriptlet 내부에서 선언되는 변수와 로직에 대한 주석은 Java 주석과 동일한 방식으로 작성한다.

7.3 page directive

- jsp 파일 주석 뒤에 한 라인 공백을 두고 페이지 directive 를 작성한다.
- contenType, import 등에 대해 각각 별도의 태그 안에 차례로 작성한다.
- import 에 2 개 이상의 클래스가 포함될 경우, 한 라인에 한 개의 클래스를 쓴다.
- import 문에는 '*'를 사용하지 않고, 반드시 클래스까지의 전체 경로를 기술한다.
- jsp 에서는 WAS 제품에 따른 호환성을 고려하여 static import 를 사용하지 않도록 한다.
- page directive 사이에는 공백 라인을 사용하지 않는다.

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8" %>
<%@ page import="java.sql.Connection" %>
<%@ page import="java.util.Vector" %>
<%@ page import="java.util.ArrayList" %>
```

7.4 선언문

- page directive 아래에 한 라인 공백을 두고 작성하는데, 이 부분은 꼭 필요한 경우에만 사용한다.
- jsp 내에서 공통으로 사용할 변수나 메소드를 정의해 사용할 수 있다.
- 변수를 모든 쓰레드에서 공유해야 하는 특별한 경우에만 선언한다.
- 2 개 이상의 변수와 메소드가 필요할 경우, 하나의 선언문(<%! ... %>)에 모아서 선언한다.
- 변수와 메소드 선언은 Java 코드와 동일한 방식을 따른다.

```
<%!
    private int hitCount;
    private Date today;

    public int getHitCount() {
        return hitCount;
    }
%>
```

7.5 scriptlet 및 tag library

- html 코드가 시작되기 전에 필요한 scriptlet 은 jsp 선언문 아래에 한 라인 공백을 두고 작성한다. jsp 선언문이 없으면 page directive 아래에 한 라인 공백을 두고 작성한다.
- 일반적인 html 코드 사이에 동적인 처리가 필요한 부분은 scriptlet 및 tag library 를 이용해 작성한다.
  -image, css, script 와 같은 링크 생성은 tag library 의 c 태그를 사용한다.
  -form 태그 작성은 Spring tag library 의 form 태그를 사용한다.
- scriptlet 기호는 첫 컬럼에 쓰고, 코드는 다음 라인에 작성한다.
- scriptlet 시작시 들여쓰기는 인접한 상위의 html 태그 요소에 맞추고, html 코드 전에 나오는 scriptlet 은 최초 컬럼에서 4 spaces 들여쓰기하여 작성한다.
- scriptlet 이후의 html 코드 들여쓰기는 scriptlet 들여쓰기와 상관없이 가장 인접한 상위의 html 코드에 맞추어 적절히 들여쓰기한다.

8. html 코딩 표준

8.1 기본사항

- html 은 크게 head 태그 영역과 body 태그 영역으로 구성된다.
- head 태그 영역에는 title, meta, link, script 태그 등이 포함된다.
- body 태그 영역에는 화면에 표시될 컨텐츠 구조와 내용이 포함된다.
- html 태그는 모두 영문 소문자로 작성하며 들여쓰기는 4 spaces 로 한다.

8.2 DOCTYPE

html 의 최상단에 DOCTYPE 을 작성한다. 프로젝트 표준 DOCTYPE 으로서 웹 표준 스펙 중 하나인 XHTML1.0 을 사용한다. DOCTYPE 은 반드시 html 의 최상단에 위치해야 하며 html 주석, 태그 등 모든 html 요소보다 앞에 있어야만 한다.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

8.3 comment

html 파일 설명을 위한 주석은 파일 크기를 줄이기 위하여 되도록이면 사용하지 않는다. 단 내용 설명이 꼭 필요한 경우에는 html 주석(<!-- -->)을 사용한다.

```
<!-- html Comment -->
```

html 문서의 가독성을 위해 의미있는 컨텐츠 영역별로 구분을 위한 html 주석을 삽입한다.

8.4 head 태그 영역

- DOCTYPE 및 html 태그를 작성 후 다음 라인에 들여쓰기 하여 head 태그를 작성한다.
- title 태그에는 화면 제목을 지정한다.
- meta 태그는 화면에 대한 정보나 charset, cache 등에 관한 제어 관련 정보를 기록한다.
- link 태그는 이 화면에 사용할 css 파일에 대한 정보를 기록하며, 2 개 이상 기록할 수 있다.
- script 태그는 화면에서 사용할 javascript 를 작성하며, 외부 js 파일을 사용할 수 있고, 본 화면에서만 사용할 javascript 코드를 작성할 수도 있다.
- 본 화면에서만 사용할 javascript 코드는 외부 js 파일 정보 다음에 작성하도록 한다.
- css, script 등을 링크할 때는 웹서버 상의 논리적인 절대경로를 사용한다.
- 태그 내의 모든 속성값(name, value, link, src 등)은 "..."와 같이 double quotation marks 를 사용한다.

```
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <title>Virtual Library</title>
    <link href="/resource/css/common.css" rel="stylesheet" type="text/css" />
    <script type="text/javascript" src="/resource/js/common.js"></script>
    <script type="text/javascript">
    // <![CDATA[
        ....
    // ]]>
    </script>
</head>
```

8.5 body 태그 영역

- head 태그 다음에 한 라인 공백을 두고 작성한다.
- 화면에 표시하고자 하는 컨텐츠의 구조와 내용을 작성한다.
- img 등의 링크시 웹서버 상의 논리적인 절대경로를 사용한다.
- 태그 내의 모든 속성값(name, value, link, src 등)은 "..."와 같이 double quotation marks 를 사용한다.

```
<body>
    <p>Moved to <a href="http://vlib.org/">vlib.org</a>.</p>
    <img src="/sp/img/arrow.gif" alt="초기화면으로 이동" />
</body>
```

9. javascript 코딩 표준

9.1 기본사항

- 특정 html 문서 내에서만 사용되는 javascript 를 제외하고는 가급적 별도의 js 파일에 저장하여 사용한다.
- javascript 의 들여쓰기는 4 spaces 로 한다.
- javascript 를 링크하거나 html 페이지 내에 작성하는 경우, script 태그에는 type 속성값을 "text/javascript"로 지정한다.

- script 태그 내에는 XML 표준문법에 맞춘 주석표시를 하고 javascript 를 작성한다.
  - '// <![CDATA['와 '//]]>'를 사용

```
<script type="text/javascript" src="/resource/js/common.js"></script>
<script type="text/javascript">
// <![CDATA[
     ..............
// ]]>
</script>
```

9.2 comment

javascript 에서는 '/* ... */'와 '// ...' 형태의 주석을 지원하며, javascript API document 를 만들기 위한 주석 처리는 '/** ... */' 로 처리한다.

9.2.1 파일 comment

SGP 내부에서 개발하는 소스에서는 js 파일에서 파일 설명을 위한 주석을 다음과 같이 표시하며 파일 최상단에 작성한다. 성명 부분은 개발 담당자명을 기술하도록 한다.

```
/**************************************************************************************************
*                    S M A R T    G R E E N    P L A T F O R M    S Y S T E M                    *
*                                                                                                 *
*                                        LG CNS Co., Ltd.                                         *
*                                      PUBLIC/SOC BIZ DIVISION                                    *
*                                                                                                 *
*                    All rights reserved.  No part of this publication may be                     *
*                    reproduced,  stored in a retrieval system  or transmitted                    *
*                    in any form or by any means  -  electronic,  mechanical,                     *
*                    photocopying, recording, or otherwise, without the prior                     *
*                    written permission of LG CNS Co., Ltd.                                       *
* ************************************************************************************************
*      파일명 : calendar.js
*      설 명 : 달력을 사용하기 위한 공통 스크립트 모음
*      수정이력 :
*       날  짜          성 명                        수정  내용
*  -------------    -----------    ---------------------------------------------------------------
*     2011/01/01       홍 길 동          Initial Release
*     2011/02/01       홍 길 동          M0001 - Xxxxxx를 수정함.
**************************************************************************************************/
```

9.2.2 함수 comment

javascript documentation 으로 자동화하여 API 문서로 생성될 수 있도록 정해진 포맷에 따른 주석을 작성하여야 한다. 모든 함수는 반드시 함수에 대한 설명과 필요 요구사항에 대한 주석을 작성하도록 한다.

```
    /**
     * 날짜에 대한 유효성검사와 포맷 적용
     * @param {String} date 날짜
     * @param {String} format 포맷
     * @type String
     * @return {String} value 변경된 값
     */
    function (date, format) {
        return value;
    }
```

9.2.3 클래스 comment

javascript 에 작성된 Object 타입의 객체들은 New 키워드를 통해 객체로 생성할 수 있기 때문에 클래스로 간주하여 주석을 작성한다. 클래스 대상이 되는 Object 의 형태는 아래의 예제를 참고하며, Object 내부에 정의된 function 또한 함수 주석에 대한 가이드에 따라 작성한다.

```
/**
  * @class 포맷을 적용하는 클래스
  */
SGPFormatter = Class.create(
        /**
         * @lends SGPFormatter.prototype
         */
    {
        /**
         * SGPFormatter 를 초기화 한다.
         * 초기화 대상 defaultDateFormat, defaultNumberFormat
         * @constructs 생성자
         */
    initialize: function() {
    },
    /**
     * 숫자에 대한 유효성검사와 포맷 적용
     * @param {String} number 숫자
```

```
     * @param {String} target element in Document
     * @type String
     * @return {String} convertNumber 변경된 값
     */
    numberFormat: function (number, target) {
    });
```

9.2.4 comment keyword

javascript API document 생성을 위한 키워드는 아래와 같은 것들이 있으며, 기본적으로 필요한 키워드 외에 추가로 선언하여 API 문서를 더 풍부하게 하는 것이 가능하다.

@class – 클래스에 대한 정보를 제공하고, 문서의 생성 부분에서 정의되어 진다.

- 문법

> @class description
>
> description - 부가정보: 클래스에 대한 description

- example

```
/**
    Creates a new Person.
    @class Represents a person.
 */
Person = function() {
}
```

@constructor – 클래스의 생성자로서 함수를 정의한다.

- 문법

> @constructor

- example

```
/**
    Creates a new Person.
    @constructor
*/
Person = function() {
}

var p = new Person();
```

@author – 코드의 author 를 지정한다.

- 문법

@author authorDescription

authorDescription – 작성자에 대한 어떠한 정보도 상관없다.

- example

```
/**
 * @author <a href="mailto:micmath@gmail.com">Michael Mathews</a>
 */
```

@augments – 이 클래스가 상속받은 클래스를 의미한다.

- 문법

@augments otherClass

otherClass – 필수항목 : 상속받는 클래스의 명칭

- example

```
/**
 * @constructor
 */
function GeneralWriter() {
}

/**
 * @constructor
 * @augments GeneralWriter
 */
function SpecialWriter() {
}
```

@returns – return 타입과 내용을 정의한다.

- 문법

@returns {returnType} returnDescription

returnType – 리턴타입 .

returnDescription – 리턴 description

- example

@type – 함수의 return type 을 정의한다.

- 문법

@type typeName

typeName – 타입명

- example

```
/**
 * @type Color
 */
```

```
function getColor() {
}
```

@param – 파라미터의 타입과 내용을 정의한다.

- 문법

```
@param {paramType} paramName paramDescription
        paramType – 파라미터 타입
        paramName – 파라미터 이름
        paramDescription – 파라미터 설명
```

- Example

```
/**
 * @param userName The name of the user.
 */
function logIn(userName) {
    // ...
}
```

@lends – Object 내의 범위에 있음을 표시한다.

- 문법

```
@lends symbolAlias
    symbolAlias – lending 하고 있는 Object 의 full name path
```

- Example

```
var Person = makeClass(
  /**
    @lends Person.prototype
  */
  {
    /** @constructs */
    initialize: function(name) {
      this.name = name;
    },
    say: function(message) {
      return this.name + " says: " + message;
    }
  }
);
```

@throws – exception 에 대한 내용을 정의한다.

- 문법

```
@throws {exceptionType} exceptionDescription
```

exceptionType – exception 타입

exceptionDescription – exception 설명

- Example

```
/**
 * @throws {OutOfMemeory} If the file is too big.
 */
function processFile(path) {

}
```

@link – 특정클래스를 바라보는 html link 를 생성한다.

- 문법

```
@anyTag This is some text {@link otherSymbol}.
```

anyTag – 어떤 태그에서도 적용이 가능하다.

otherSymbol – HTML link 를 원하는 Object 의 namepath

- Example

```
/**
 * This is similar to a {@link Printer} object but can also print to files.
 */
function Writer() {
}
```

9.3 변수 선언

- 전역 변수는 꼭 필요한 경우에만 사용하며, javascript 소스 맨 처음에 한 번에 모아서 선언한다.
- 지역 변수는 javascript 함수 시작 부분에 한 번에 모아서 선언한다.
- javascript 는 변수를 선언없이 사용할 수 있지만 유지보수의 용이성을 위해 반드시 미리 선언하고 사용한다.
- 변수 선언시 반드시 var 키워드를 붙이도록 한다.

```
var layerCnt = 1;

function fncSave() {
    var initialCount = 0;
    ....
}
```

9.4 함수 선언

함수에 리턴타입과 파라미터 타입을 선언하지 않는다는 것을 제외하고는 java 의 메소드 선언과 유사하다.

```
function fncSaveMemberInfo(form) {
    form.userId = "7284738";
    form.xxx();
}
```