I. Liferay 개발 환경 구축

1. MySql 설치

에디슨 포털에서 사용하는 MySQL 데이터베이스 관리 시스템은 버전 5.5 이상이어야 한다.

우선 MySQL 다운로드 사이트로부터 최신 버전 소스를 다운로드 받는다.

```
shell> wget
http://dev.mysql.com/get/Downloads/MySQL-5.6/mysql-5.6.12.tar.gz/from/http://cdn.mysql.com/
```

MySQL 5.6 버전에서는 소스를 컴파일 하기 위해서 cmake를 사용한다. 서버에 cmake가 설치되어 있는지를 확인하고, 만약 cmake가 설치되어 있지 않으면 다음의 명령어를 사용하여 설치한다.

```
shell> yum install cmake
```

다운로드 받은 소스를 압축을 풀고 다음 과정을 시작한다.

```
# Preconfiguration setup
shell> groupadd mysql
shell> useradd -r -g mysql mysql
# Beginning of source-build specific instructions
shell> tar zxvf mysql-VERSION.tar.gz
shell> cd mysql-VERSION
shell> cmake .[1)]
shell> make
shell> make install
# End of source-build specific instructions
# Postinstallation setup
shell> cd /usr/local/mysql
shell> chown -R mysql .
shell> chgrp -R mysql .
shell> scripts/mysql_install_db --user=mysql[2)]
```

---

1) 이 명령어를 수행할 때 다음과 같은 오류를 발생시킬 수 있다.

```
Could NOT find Curses (missing: CURSES_LIBRARY CURSES_INCLUDE_PATH)
```

이 때는 다음과 같이 ncurses-devel 라이브러리를 설치하고, CMakeCache.txt 파일을 삭제한 후에 다시 실행한다.

```
shell> #yum -y install ncurses-devel
```

2) 이 명령어는 '/usr/sbin/mysqld: unknown variable '--defaults-char-set=utf8'이란 오류 메시지와 함께 실패할 수 있다. 이 때는 다음의 명령어로 다시 실행한다.

```
scripts/mysql_install_db --no-defaults --character-set-server=utf8 --user=mysql
```

```
shell> chown -R root .
shell> chown -R mysql data
# Next command is optional
shell> cp support-files/my-medium.cnf /etc/my.cnf
shell> bin/mysqld_safe --user=mysql &
shell> cp support-files/mysql.server /etc/init.d/mysql.server
shell> cp bin/mysql /usr/bin/
```

MySQL을 설치한 후에는 반드시 UTF8로 chracter set이 지정되었는 지를 확인해야 한다.

```
mysql> select charset('한글');
+------------------+
| charset('한글')        |
+------------------+
| utf8                   |
+------------------+
mysql> show global variables like 'character_set_%';
+----------------------+------+
| Variable_name        | Value  |
+----------------------+------+
| character_set_client  | utf8     |
| character_set_connection | utf8     |
| character_set_database    | utf8     |
| character_set_filesystem   | binary   |
| character_set_results       | utf8     |
| character_set_server | utf8     |
| character_set_system| utf8     |
+----------------------+------+
```

만약 character_set_filesystem이 binary인 것을 제외하고 utf8로 설정되어 있지 않다면 mysql 설정파일인 my.cnf 파일의 [mysqld] 섹션에 다음과 같이 설정한다.

```
[mysqld]
init_connect=SET collation_connection = utf8_general_ci
init_connect=SET NAMES utf8
character-set-server = utf8
collation-server=utf8_general_ci
character-set-client-handshake = false
```

만약 수정하지 않고 다음 단계를 진행한다면 깨진 한글 때문에 라이프레이가 제대로 작동하지 않는다.

2. 라이프레이 설치

□ 단계 1: 라이프레이 다운받기

라이프레이 다운로드 페이지에서 적당한 버전을 다운받는다. 이 예제에서는 톰캣과 통합된 패키지인 버전 6.1.1을 사용한다.

```
shell> wget
http://sourceforge.net/projects/lportal/files/Liferay%20Portal/6.1.1%20GA2/liferay-portal-tomcat-6.1.1-ce-ga2-20120731132656558.zip/download
```

□ 단계 2: 라이프레이 설치 디렉토리 정의

파일 다운로드가 완료되면 압축을 푼다. 압축을 풀면 기본적으로 라이프레이의 버전이 디렉토리 이름으로 저장되는데, 사용하기 쉽도록 이 예제에서는 그냥 edison-portal이라고 라이프레이 설치 디렉토리 이름을 정의한다.

```
shell> unzip liferay-portal-tomcat-6.1.1-ce-ga2-20120731132656558.zip
shell> mv liferay-portal-6.1.1-ce-ga2 edison-portal
```

이제 라이프레이는 설치되었지만, 실질적으로는 사용할 준비가 끈난 것은 아니다. 라이프레이를 사용하기 위해서는 몇 가지 사전 절차가 필요하다.

□ 단계 3: 라이프레이가 사용할 데이터베이스 정의

우선 라이프레이와 EDISON에서 사용할 데이터베이스를 정의해야 한다. 이 예제에서는 edison이라고 정의한다.

```
shell> mysql
Welcome to the MySQL monitor.  Commands end with ; or ₩g.
Your MySQL connection id is 5
Server version: 5.6.12 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '₩h' for help. Type '₩c' to clear the current input statement.

mysql>  create database edison character set utf8;
```

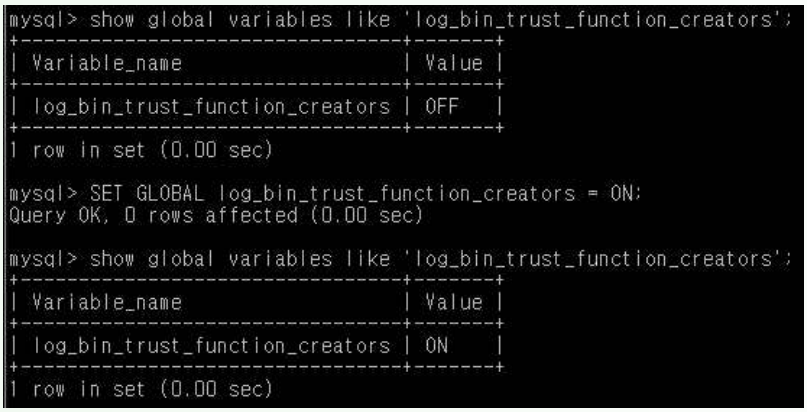
```
Query OK, 1 row affected (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| edison             |
| edison_cc          |
| mysql              |
| performance_schema |
| test               |
+--------------------+
6 rows in set (0.00 sec)

mysql> GRANT ALL ON edison.* to 'edison'@'%' identified by 'edison2013!!';
Query OK, 0 rows affected (0.00 sec)
```

위의 예제에서 마지막 명령은 edison DB에 edison이란 사용자가 모든 권한을 가지고 로컬 호스트 또는 원격 접속할 수 있도록 허용해주는 명령이다.

EDISON 포털을 위한 데이터베이스에는 필요한 Function들을 정의하여 사용하고 있다. 이 함수들을 사용하기 위한 옵션을 켜고 확인한다.



이제 edison 계정으로 서버에 접속해서 mysql에 접속해보면, edison DB에 edison 계정으로 잘 접근할 수 있다는 것을 볼 수 있다.

```
[edison@xenvm1 ~]$ mysql -u edison -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.6.12 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| edison             |
| test               |
+--------------------+
3 rows in set (0.00 sec)
```

이제 EDISON 포털을 설치하기 위해서는 EDISON 포털 개발팀에서 개발된 포틀릿이 필요로 하는 데이터베이스 테이블을 구성해야 한다. 원래 이 과정은 포틀릿을 사용하기 시작하였을 때 자동으로 필요한 데이터 테이블을 작성하고 시작되는 것이 옳은 과정인 것으로 생각되지만, 아직은 그 과정까지 개발 능력이 도달하지 못하여 일단 시작 전에 필요한 테이블 구조를 수동으로 구성하도록 한다. 필요한 SQL 문장들은 부록에 수록되어 있다. 차례로 수행하여 이미 생성해 놓은 edison DB에 구성하도록 한다.

```
mysql> source Table.sql
```

테이블 생성이 끝나면 작성된 프로시저들을 생성한다. 프로시저 정의는 부록에 수록되어 있다.

```
mysql> source Procedure-Function.sql
```

마지막으로 시스템에서 공통적으로 사용하는 데이터를 로딩한다.

```
mysql> source Sys_common_cd.sql
```

□ 단계 4: Tomcat 설정

Tomcat은 이미 앞에서 다운 받은 라이프레이에 버전 7.0.26이 포함되어 있다. 톰캣을 구동하기 전에 미리 몇 가기 설정을 해야 한다.

일단 포트는 80 포트로 한다. 톰캣의 기본 포트는 8080이므로 server.xml을 열

어서 수정한다.

```
<Connector port="80" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" URIEncoding="UTF-8" />
```

만약 톰캣을 루트 계정으로 실행시키는 것이 아니라면 포트 1~1023은 오로지 루트 계정으로 오픈 가능하기 때문에 위의 수정은 허락되지 않는다. 이런 경우에는 다음과 같은 해결 방법이 있다.

- Apache HTTPD를 루트로 실행한 뒤 접속 요구를 톰캣 포트로 연결한다.
- jsvc를 사용한다.

다음은 이메일을 설정한다. 이 예제에서 모든 이메일은 gmail을 이용해서 주고받은 것으로 정의한다. edison 포털을 위한 gmail 계정은 다음과 같이 이미 생성되어 있다.

```
ID: portal.edison.kisti@gmail.com
passwd: edison2013!!
```

$CATALINA_HOME/conf/Catalina/localhost/ROOT.xml을 열어서 다음을 입력한다.

```
<Resource
  name="mail/MailSession"
  auth="Container"
  type="javax.mail.Session"
  mail.imap.host="localhost"
  mail.pop.host="localhost"
  mail.store.protocol="imap"
  mail.transport.protocol="smtp"
  mail.smtp.host="smtp.gmail.com"
  mail.smtp.port="465"
  mail.smtp.auth="true"
  mail.smtp.starttls.enable="true"
  mail.smtp.user="portal.edison.kisti@gmail.com"
  password="edison2013!!"
  mail.smtp.socketFactory.class="javax.net.ssl.SSLSocketFactory"
/>
```

ROOT.xml의 첫줄을 다음과 같이 변경하여 reloadable 속성을 추가한다.

```
<Context path="" crossContext="true" reloadable="true">
```

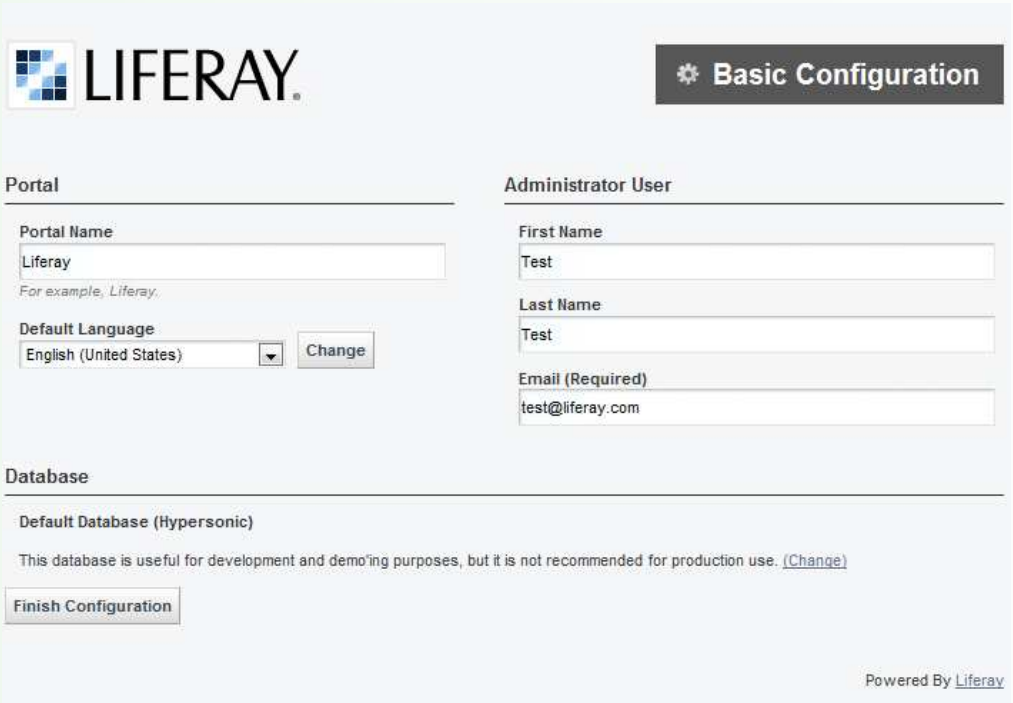
만약 MySql의 버전이 5.6.x라면 톰캣과 MySql의 연결을 위해서 추가적인 설정이 필요하다. 톰캣 7.0.x 버전에 포함된 MySql 연결자는 MySql 버전 5.6을 지원하지 않는다. 따라서 MySql 5.6 버전을 지원하는 연결자를 다운받아서 톰캣 디렉토리에 있는 연결자를 대체해주어야 한다.

```
shell> wget
http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.25.tar.gz/from/http://cdn.
mysql.com/
shell> tar zxvf mysql-connector-java-5.1.25.tar.gz
shell> cp mysql-connector-java-5.1.25/mysql-connector-java-5.1.25-bin.jar
edison-portal/tomcat-7.0.27/lib/ext/mysql.jar
```

톰캣을 실행시키면 라이프레이의 초기 설정을 위해서 웹 브라우저가 실행된다.

3. 라이프레이 초기 설정

Tomcat을 실행하고 웹 브라우저를 통해서 처음 접속하면 라이프레이의 초기 설정 화면이 나타난다. 이 설정 화면에서는 포털의 명칭을 정의하고, 관리자를 정의하며, 데이터베이스 연결을 설정할 수 있다.



[그림 3] 라이프레이의 초기 설정 화면

[Portal Name] 항목에는 구축하려고하는 포털의 이름을 정의한다. 이 책의 예에서는 Edison이다. [Default Language] 항목에서는 언어를 선택하면 되는데, 그냥 영어로 정의해서 놓는다. 언어는 한국어도 지원되지만, 이는 번역기를 돌린 언어지원으로, 매우 어색하며 맞지 않는다.

[Administrator User] 섹션에는 포털의 슈퍼 유저를 정의한다. 슈퍼 유저는 이 포털에 관한한 신과 다름없다. 따라서 신중해야 한다.

아래 부분의 [Database] 섹션에서는 사용하고자 하는 데이터베이스를 정의할 수 있다. 아무 것도 정의하지 않으면 기본적으로 라이프레이에서 제공하는 Hypersonic이란 DBMS를 사용한다. 이 책에서는 MySQL을 사용할 예정이므로 이 섹션을 잘 정의하도록 한다.

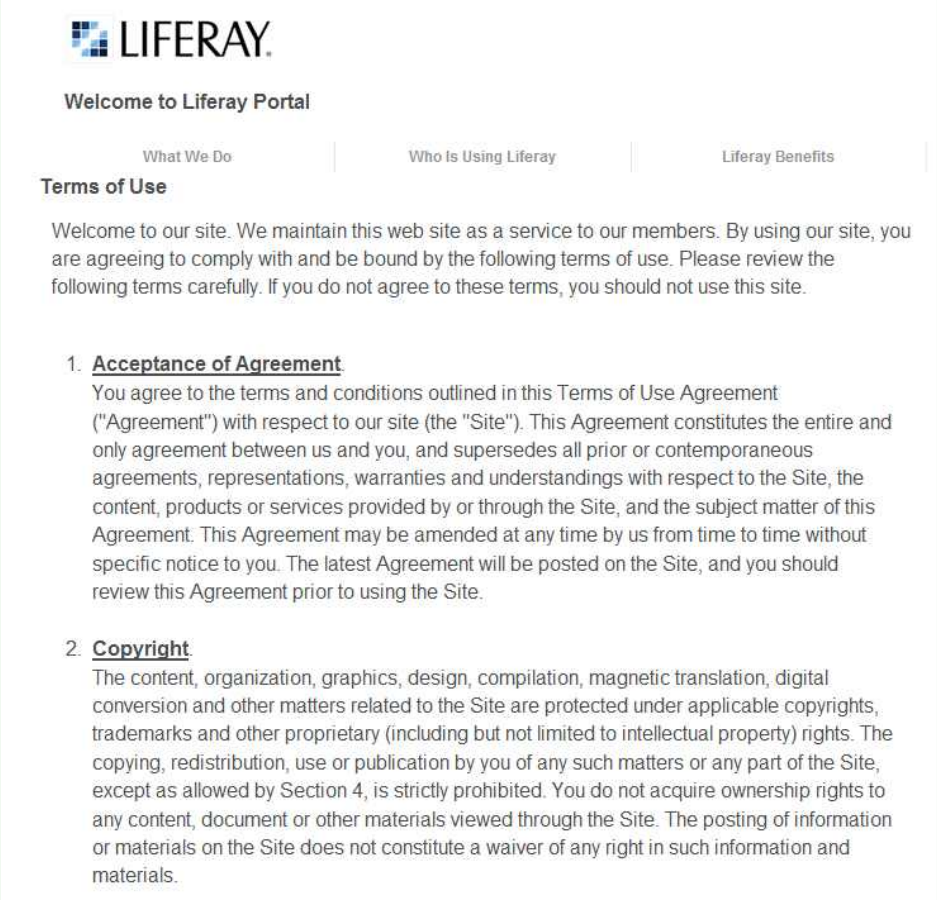
한 가지 기억할 것은 MySQL에 Edison 포털에서 사용할 DB와 DB의 사용자가 미리 정의되어 있어야 한다는 점이다. MySQL에 DB를 생성하고 사용자를 정의하는 방법은 이 책의 범위에서 벗어나기 때문에 구체적으로 설명하지 않지만, MySQL의 사용법과 SQL 사용법을 통하여 쉽게 알 수 있다.

[그림 4] 라이프레이에서 사용할 DBMS 선택 및 DB 정의

[JDBC URL] 항목에 보면, [jdbc:mysql://localhost/lportal?...]이 되어 있다. 이 내용은 사용할 DB 이름과 문자 셑 등을 정의한다. 이 책에서는 Edison 포털이 사용할 DB를 edison으로 이미 정의해 놓았다. 따라서 문장 중의 lportal을 edison으로 수정해야 한다. [Finish Configuration] 버튼을 누르면 다음과 같은 성공 메시지가 나타난다.

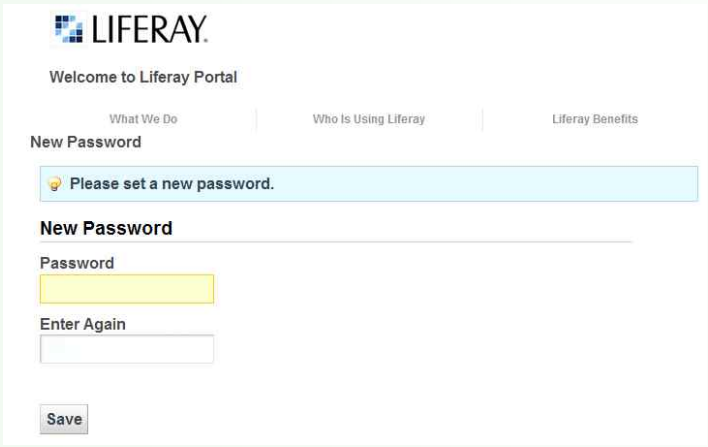
[그림 5] MySQL에 Edison 포털 DB 생성을 성공적으로 끝낸 메시지

[Go to My Portal] 버튼을 누르면 라이센스 관련한 화면이 나타나는데, [I Agree] 버튼을 눌러서 통과한다.



[그림 6] 라이프레이의 사용권 계약 동의 화면

사용권 계약에 동의하면 마지막 단계로 슈퍼 유저의 패스워드를 정의하는 화면이 나타난다. 이 암호는 앞에서 정의한 슈퍼 유저가 로그인할 때 사용하는 것이므로 절대로 잊어버려서는 안되는 중요한 정보이다. 따라서 신중하게 결정해야 한다.



[그림 7] 슈퍼 유저의 패스워드 정의 화면

패스워드를 정의하고 저장하면 다음 단계로 패스워드를 잊어버렸을 때 기억해 내기 쉽도록 패스워드 기억 정의 화면이 나타난다. 단 답을 할 때 오로지 영어로 답해야 한다는 것을 명심하자. 한글로 답을 쓰면 라이프레이는 한없이 긴 오류를 발생시킨다.

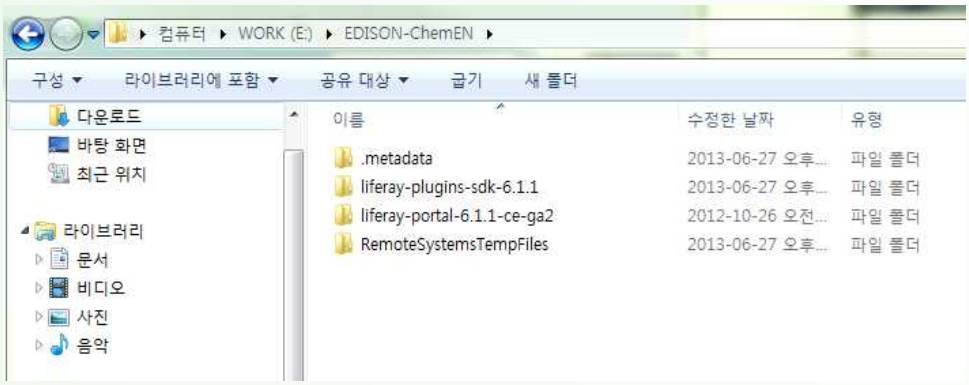
[그림 8] 패스워드 기억 정의 화면

위의 단계를 지나면 비로소 라이프레이 기반의 새로운 포털, Edison의 첫 페이지를 볼 수 있다. 이 첫 화면은 라이프레이에서 제공하는 예제 콘텐츠들을 포함하여 보여준다.

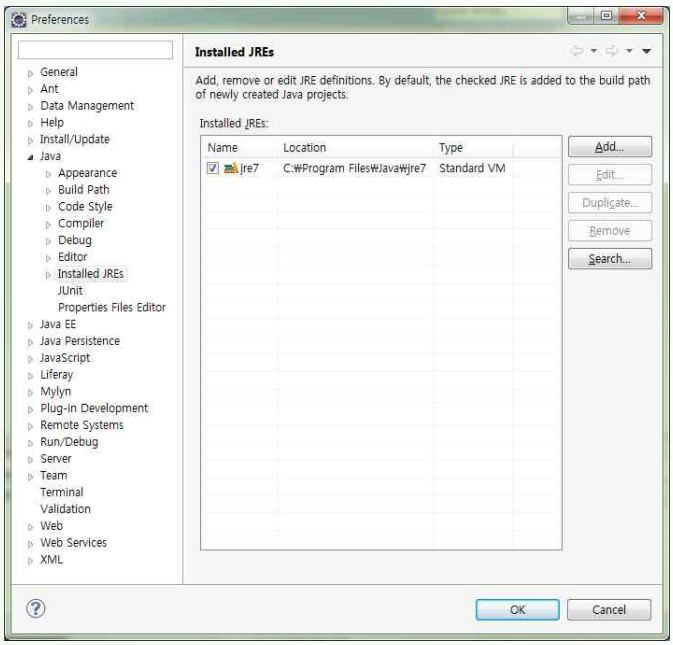
설정이 끝났으면 톰캣을 종료한다.

4. EDISON 개발환경 구성

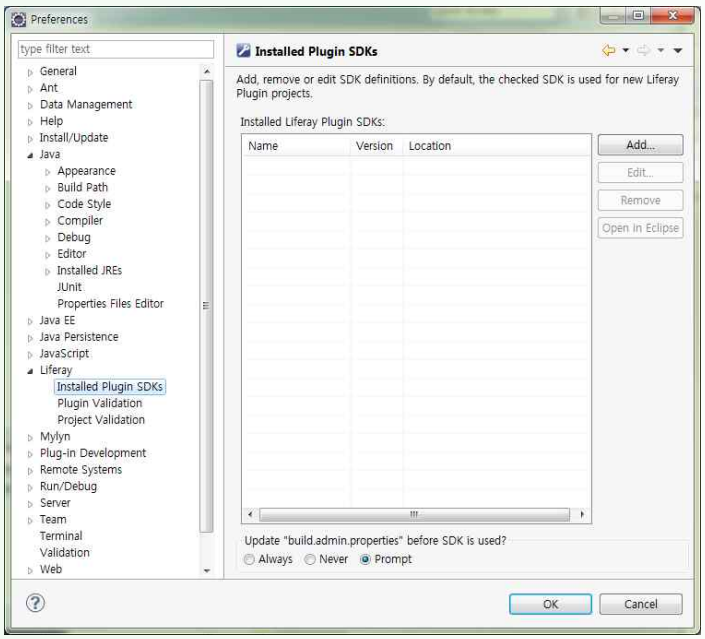
라이프레이의 초기 구성이 끝났으면 라이프레이 IDE를 이용해서 개발 환경을 구성한다. 새로운 포탈이 될 폴더를 하나 생성하고, 새로운 폴더 아래에는 라이프레이 SDK를 다운받아 저장하도록 한다. IDE를 실행시키면서 워크스페이스를 생성한 폴더로 지정한다. 이 예제에서는 라이프레이 포탈을 아예 워크스페이스 아래로 포함시켜버렸다.



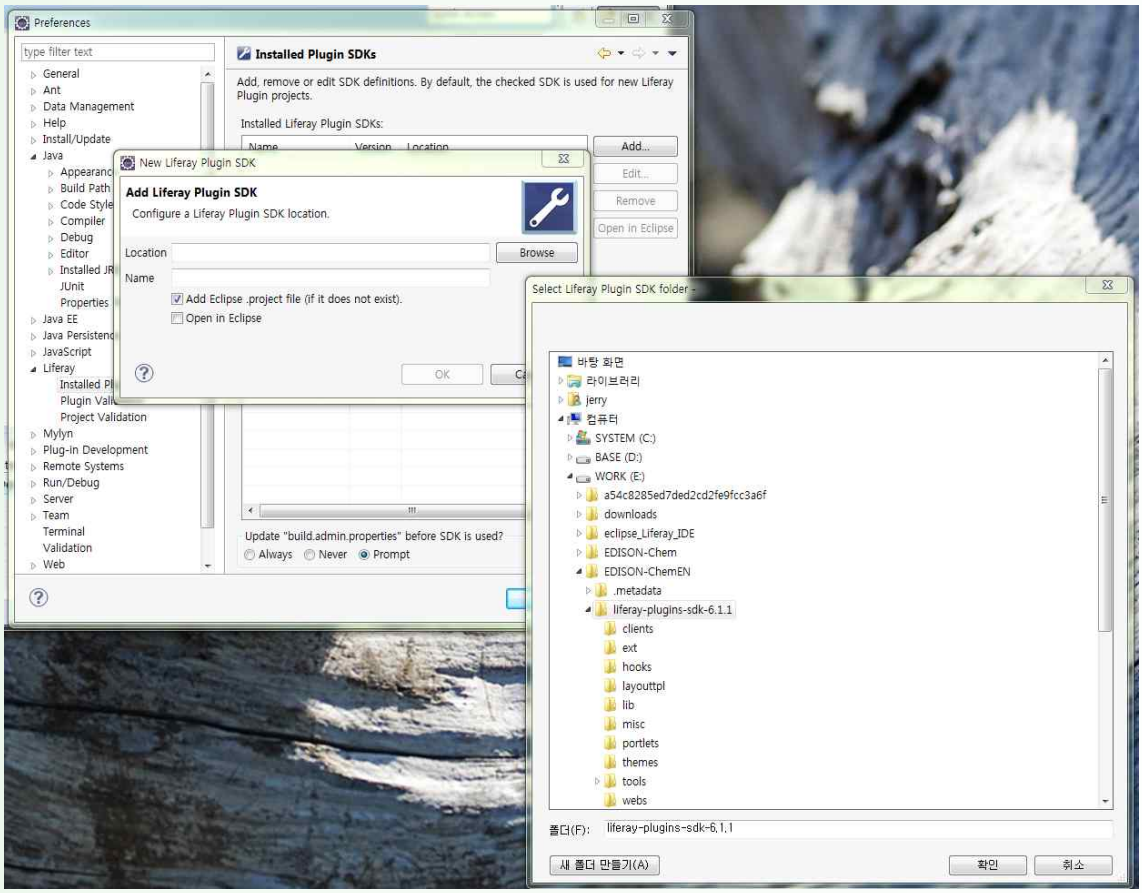
IDE를 실행한 후에는 여러가기 설정이 필요하다. 우선 Java 환경이 어떻게 되어 있는 지 확인한다. [Window => Preferences] 메뉴를 눌러 대화 창을 열고 [Java => Installed JREs] 항목을 확인한다.



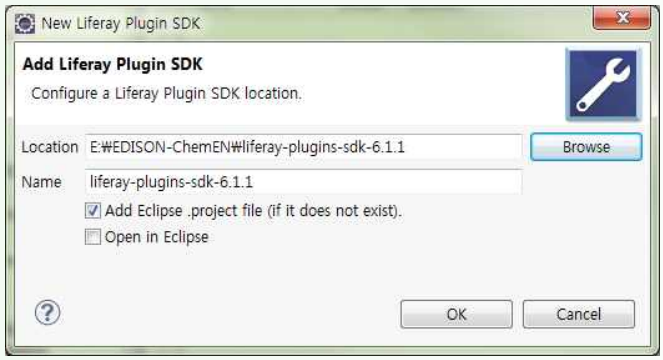
다른 버전의 JAVA를 사용할 수도 있다. 이 과정은 개발자 매뉴얼에서 더 자세히 논의한다. 다음은 [Liferay => Installed Pligin SDKs]를 눌러서 설치한 SDK를 추가하도록 한다.



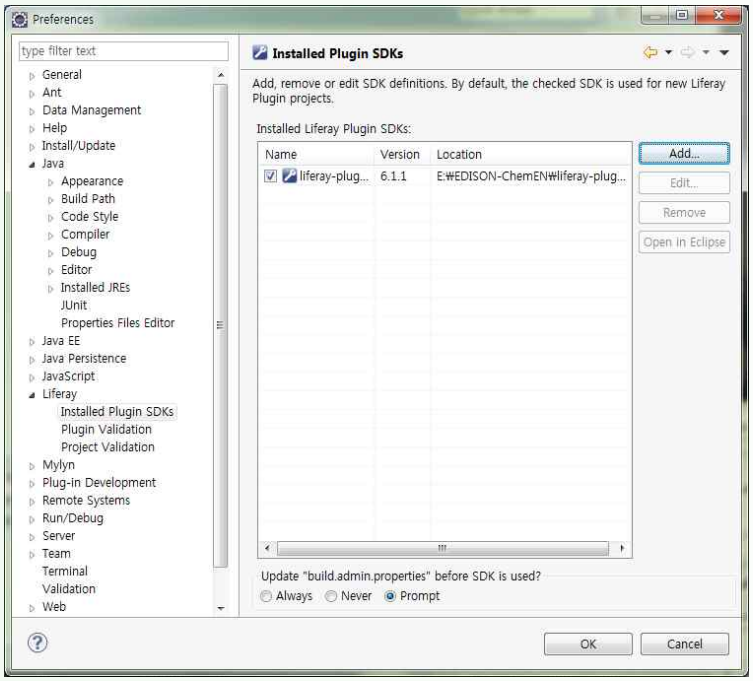
[Add] 버튼을 누른다.



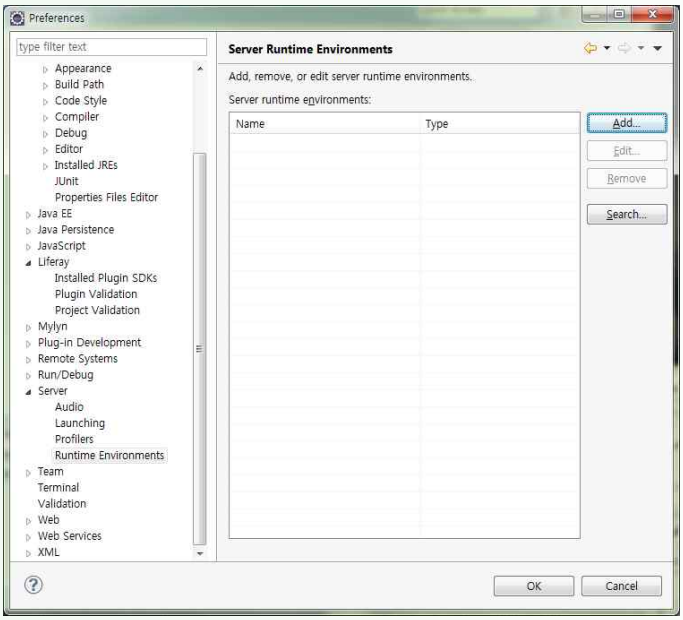
대화창에서 [Browse] 버튼을 누르고 이미 저장해둔 SDK 폴더를 선택한 뒤 확인을 누른다.



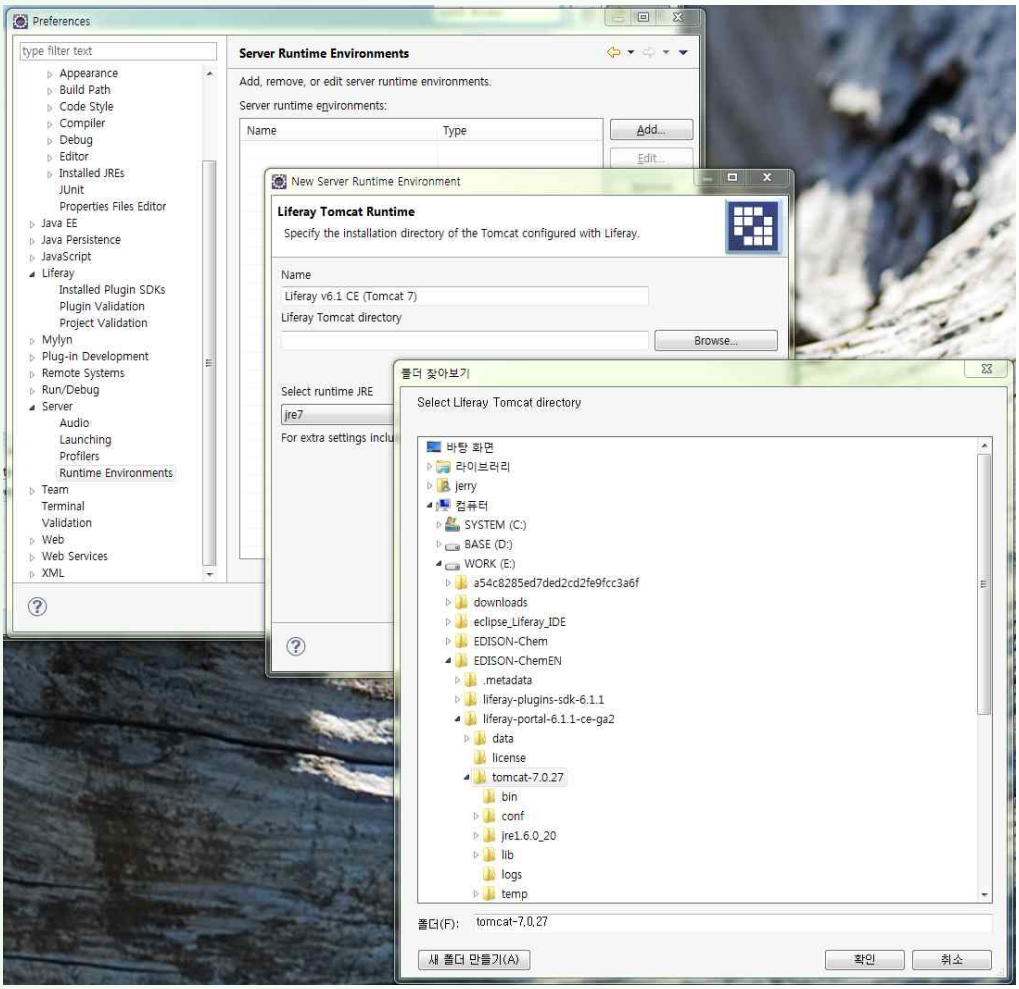
[OK] 버튼을 누르면 SDK 추가가 완료된다.



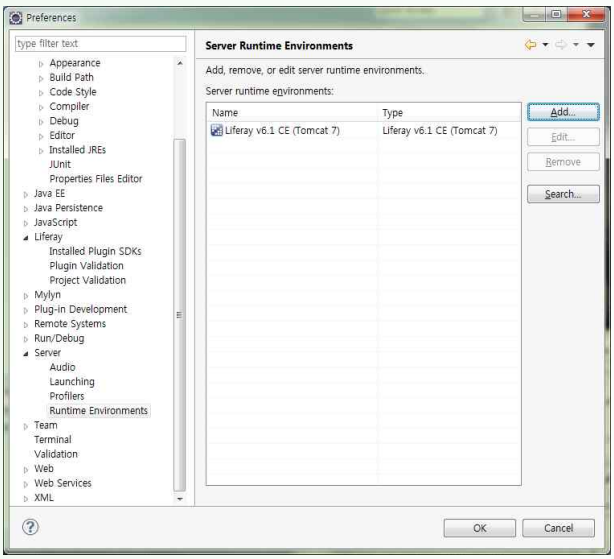
마지막으로 [Server => Runtime Environments]를 선택하고 실행 환경을 추가한다.



[Add...] 버튼을 누르고 [Liferay, Inc. => Liferay v6.1 CE(Tomcat 7) 항목을 선택한 뒤 [Next >] 버튼을 누른다. 대화창에서 [Browse...] 버튼을 누르고 Liferay Tomcat directory를 지정한다.

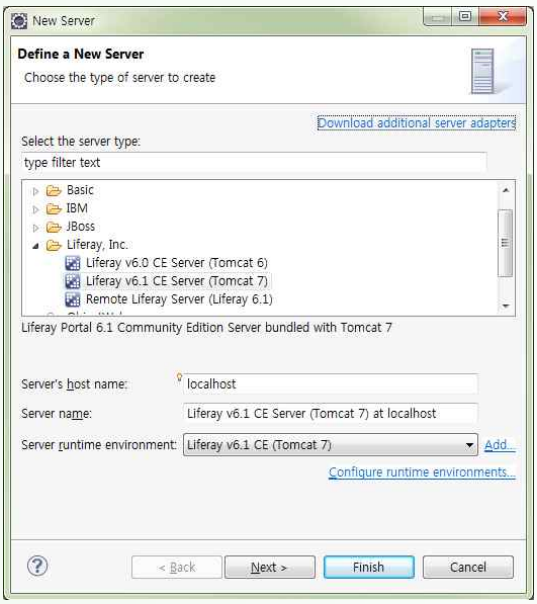


[확인]과 [Finish] 버튼을 차례로 눌러 실행환경 추가를 완료한다.

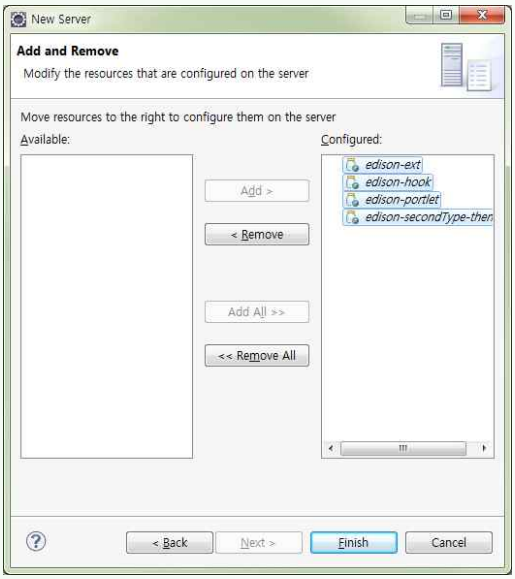


라이프레이 IDE에서는 설치된 톰캣 서버를 IDE 상에서 구동하고 종료할 수 있다. 개발을 하는 데에 있어 IDE 상에서 톰캣을 구동시킬 수 있는 기능은 대단히 편리하므로, 다음과 같이 IDE와 톰캣을 연결한다.

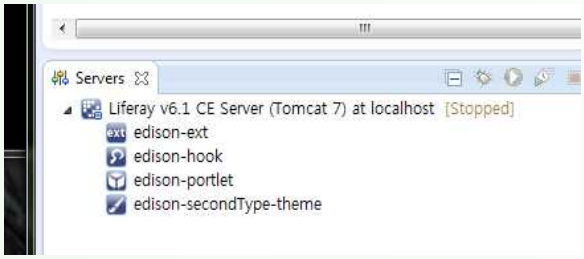
IDE의 라이프레이 Perpective는 6 서브 창으로 구분되어 있는 데, [Servers] 창에서 [new server wizard..] 링크를 클릭한다.



[Next>] 버튼을 누른다.



[Available]에 있는 모든 프로젝트를 [Configured]로 추가하고 [Finish] 버튼을 누른다.



이제 라이프레이 IDE에서 톰캣을 실행과 중단할 수 있으며, 네 개의 프로젝트들은 실시간 디플로이된다.

II. Science Platform

III. SciencePlatform Hook

1. 프로퍼티 설정

2. Hook을 이용한 라이프레이 기능 수정

□ 커스텀 필드를 가진 계정 생성 및 사용자 관리

3. 자산(Assets)

자산은 라이프레이의 협업 기능의 기반이다. 자산 기능을 통하여 엔터티를 포털 전체에 게시할 수 있다. 자산은 사교 API와 같은 피드 패턴을 사용한다. 웹 어플리케이션에 자산 기능을 사용하기 위해서는 다음과 같은 절차를 따르면 된다.

- 참조 엔터티와 필드를 service.xml에 추가
- 서비스 레이어에 자산추가
- 엔터티를 자산으로 번역 해주는 AssetRenderer 생성

1) service.xml 수정

① 엔터티 선언부에 uuid=“true” 속성을 추가한다.

```
<entity name="ScienceApp" local-service="true" uuid="true">
```

② 자산과 관련한 엔터티 클래스를 마지막 finder 엘리먼트 다음에 삽입한다. 이 때 사용되는 엘리먼트는 reference이며, 여기에 정의된 클래스는 서비스 빌더에 의하여 생성되는 ScienceAppLocalServiceBaseImpl 클래스에 삽입되어 확장 클래스인 ScienceAppLocalServiceUtil 클래스에서 인스턴스로 접근할 수 있다.

```
<reference package-path="com.liferay.portlet.asset" entity="AssetEntry" />
<reference package-path="com.liferay.portlet.asset" entity="AssetLink" />
```

③ 사이언스앱을 상태에 따라 검색할 수 있도록 finder를 정의한다. 여기에서 정의된 finder는 ScienceAppPersistence 클래스에 findScienceAppByStatus() 멤버 함수로 자동 생성된다.

```
<finder name="Status" return-type="Collection">
  <finder-column name="status"></finder-column>
</finder>
```

④ 서비스 빌더 실행

2) ScienceAppLocalServiceImple.java 수정

① 앞에서 언급했듯이 상태에 따라 사이언스앱을 검색하는 함수는 findScienceAppByStatus()라는 이름으로 퍼시스턴스 클래스에 생성된다. 그러나 사용자는 가능한 한 유틸 클래스로 모든 작업을 수행할 수 있기를 희망하고 있다. 따라서 ScienceAppLocalServiceImpl 클래스에 다음의 함수를 추가한다.

```
public List<ScienceApp> getScienceAppByStatus(int status) throws SystemException{
  return super.scienceAppPersistence.findByAppStatus(status);
}
```

IV. ScienceApp

V. ScienceWorkflow

1. 개요

2. 시나리오

VI. Science Platform Service APIs

사이언스 플랫폼 서비스는 근본적으로 라이프레이(Liferay) 웹 포털 플랫폼을 기반으로 개발되었다. 라이프레이는 워크벤치를 통하여 기본적인 서비스 API들을 자동으로 생성해준다. 이 장에서는 라이프레이에서 자동으로 생성해주는 API들에 대한 문서는 라이프레이에 맡기고, 사이언스 플랫폼 서비스를 위하여 새롭게 정의한 API들에 대하여만 설명한다.

사이언스 플랫폼에 데이터를 저장하기 위해서는 데이터 모델들을 생성하여 이요하는 것이 필요하다. 이를 위하여 라이프레이는 기본적으로 ServiceUtil 클래스에 create() 함수를 생성하여 제공한다. 기본 API들을 사용하여 콘트롤러에서 데이터 핸들링을 하는 것도 문제없지만 사이언스 플랫폼에서는 프로그래밍의 일관성 유지와 편의를 위하여 추가적인 API를 제공한다.

모델을 생성하고 저장하는 주 API는 create()와 addxxxxxx()함수이며 모든 ServiceUtil에서 동일한 패턴의 API를 제공한다.

create() 함수는 기본 정보로 데이터 모델을 생성하여 반환한다. 이 때 반환된 데이터 모델 인스턴스에 저장된 모든 데이터는 아직 데이터베이스에 저장되지 않고, addxxxxxx() 함수를 호출해야 비로소 저장된다. 따라서 콘트롤러에서는 create()함수를 호출하여 데이터 인스턴스를 생성한 뒤 저장할 데이터를 인스턴스에 우선 저장하고, addxxxxxx() 함수를 호출하여 저장하여야 한다.

1. ScienceAppLocalServiceUtil

□ createScienceApp (String appName, String appVersion, ServiceContext sc) throws SystemException

○ 설명

ScienceApp 모델 인스턴스를 생성하여 반환한다.

○ 인수

인수 명	데이터 형	설명
appName	String	사이언스 앱 이름. 영문 대소문자, 숫자, '_', '-','.' 만 허용
appVersion	String	사이언스앱 버전. xxx.xxx.xxx 3개 섹션으로 구성
sc	ServiceContext	ScienceApp 모델에 대한 서비스 콘텍스트

○ Return

- 앱이 이미 존재하거나 앱 이름이 규정된 규칙에 맞지 않으면 null 인스턴스를 반환
- 버전이 작성 규칙에 맞지 않거나 기존의 버전보다 낮으면 null 인스턴스를 반환

- 그렇지 않으면 ScienceApp 인스턴스를 반환
- 반환된 인스턴스에는 다음과 같은 데이터가 초기 저장되어 있다.

속성 명	초기 데이터값
scienceAppId	자동 증가 값
name	앱 이름
version	앱 버전
stage	ScienceAppConstants.EMPTY
authorId	현재 사용자 아이디
createDate	현 일시
modifiedDate	현 일시
userId	현재 사용자 아이디
recentModifierId	현재 사용자 아이디
groupId	Scope Group Id
companyId	company id

○ Exception

SystemException

□ copyScienceApp (long scienceAppId, ServiceContext sc) throws SystemException

○ 설명

ScienceApp 아이디가 scienceAppId인 인스트너스를 복제하여 반환한다.

○ 인수

인수 명	데이터 형	설명
scienceAppId	long	사이언스 앱 아이디
sc	ServiceContext	ScienceApp 모델에 대한 서비스 콘텍스트

○ Return

- ScienceApp 인스턴스
- 반환된 복제 인스턴스에는 모든 데이터가 원 인스턴스의 데이터와 동일하나, 다음의 속성들은 초기화 되어 있다.

속성 명	초기 데이터값
scienceAppId	자동 증가 값
stage	ScienceAppConstants.EMPTY
createDate	현 일시
modifiedDate	현 일시
userId	현재 사용자 아이디
recentModifierId	현재 사용자 아이디
groupId	Scope Group Id
companyId	company id

○ Exception

SystemException

□ addScienceApp(ScienceApp scienceApp, ServiceContext sc) throws SystemException,

PortalException

○ 설명

ScienceApp 인스턴스를 데이터베이스에 저장한다.

○ 인수

인수 명	데이터 형	설명
scienceApp	ScienceApp	사이언스 앱 인스턴스
sc	ServiceContext	ScienceApp 모델에 대한 서비스 콘텍스트

○ Return

- 저장된 ScienceApp 인스턴스

○ Exception

SystemException, PortalException

□ boolean verifyScienceAppName(String appName) throws SystemException

○ 설명

ScienceApp 이름이 정해진 규칙에 맞는 지와 이미 데이터베이스에 동일한 이름이 있는지를 조사한다.

○ 인수

인수 명	데이터 형	설명
scienceAppName	String	사이언스 앱 이름

○ Return

- 사이언스앱 이름이 정해진 규칙에 맞지 않거나, 데이터베이스에 이미 존재하면 false
- 그렇지 않으면 true

○ Exception

SystemException

□ boolean existAppName(String appName) throws SystemException

○ 설명

ScienceApp 이름이 이미 데이터베이스에 동일한 이름이 있는지를 조사한다.

○ 인수

인수 명	데이터 형	설명
scienceAppName	String	사이언스 앱 이름

○ Return

- 사이언스앱 이름이 데이터베이스에 이미 존재하면 false
- 그렇지 않으면 true

○ Exception

SystemException

□ boolean existApp(String appName, String appVersion) throws SystemException

○ 설명

동일한 버전의 사이언스앱이 이미 데이터베이스에 동일한 이름이 있는지를 조사한다.

○ 인수

인수 명	데이터 형	설명
scienceAppName	String	사이언스 앱 이름
appVersion	String	사이언스앱 버전. xxx.xxx.xxx 3개 섹션으로 구성

○ Return

- 동일한 버전의 사이언스앱이 데이터베이스에 이미 존재하면 false
- 그렇지 않으면 true

○ Exception

SystemException

□ ScienceApp getLatestVersion(String appName) throws SystemException

○ 설명

가장 최근 버전의 appName 사이언스앱을 검색

○ 인수

인수 명	데이터 형	설명
scienceAppName	String	사이언스 앱 이름

○ Return

- ScienceApp 인스턴스
- 지정한 이름을 가진 사이언스앱이 없으면 null

○ Exception

SystemException

□ boolean verifyVersionNumber(String appName, String appVersion) throws SystemException

○ 설명

사이언스앱의 버전이 정해진 규칙에 따라 작성되어 있고 유효한 지를 검사한다.

○ 인수

인수 명	데이터 형	설명
scienceAppName	String	사이언스 앱 이름
appVersion	String	사이언스앱 버전. xxx.xxx.xxx 3개 섹션으로 구성

○ Return

- 버전이 규칙에 따라 생성되었거나, 이전 버전보다 크면 true
- 그렇지 않으면 false

○ Exception

SystemException

□ ScienceApp deleteScienceApp(long scienceAppId) throws SystemException, PortalException

○ 설명

데이터베이스로부터 지정된 사이언스앱 정보를 삭제

○ 인수

인수 명	데이터 형	설명
scienceAppId	long	사이언스 앱 아이디

○ Return

- 삭제된 사이언스앱 인스턴스

○ Exception

SystemException, PortalException

□ ScienceApp deleteScienceApp(ScienceApp scienceApp) throws SystemException, PortalException

○ 설명

데이터베이스로부터 지정된 사이언스앱 정보를 삭제

○ 인수

인수 명	데이터 형	설명
scienceApp	ScienceApp	사이언스 앱 인스턴스

○ Return

- 삭제된 사이언스앱 인스턴스

○ Exception

SystemException, PortalException

□ ScienceApp updateScienceApp(ScienceApp scienceApp, ServiceContext sc) throws SystemException, PortalException

○ 설명

데이터베이스에 사이언스앱 정보 수정

○ 인수

인수 명	데이터 형	설명
scienceApp	ScienceApp	사이언스 앱 인스턴스
sc	ServiceContext	ScienceApp 모델에 대한 서비스 콘텍스트

○ Return

- 삭제된 사이언스앱 인스턴스

○ Exception

SystemException, PortalException

□ String getBinPath(long scienceAppId) throws PortalException, SystemException

○ 설명

지정된 사이언스 앱의 실행 파일 경로를 생성

○ 인수

인수 명	데이터 형	설명
scienceAppId	long	사이언스 앱 아이디

○ Return

- String: 사이언스앱 실행 파일 경로

○ Exception

SystemException, PortalException

□ String getSrcPath(long scienceAppId) throws PortalException, SystemException

○ 설명

지정된 사이언스 앱의 소스 파일 경로를 생성

○ 인수

인수 명	데이터 형	설명
scienceAppId	long	사이언스 앱 아이디

○ Return

- String: 사이언스앱 소스 파일 경로

○ Exception

SystemException, PortalException

VII. Science Platform Model APIs

2. ScienceApp

□ boolean isApproved()

○ 설명

사이언스 앱이 서비스 중인 지 아닌지를 조사한다.

○ Return

- 사이언스앱이 서비스 중이면 true
- 그렇지 않으면 false

□ String getBinPath()

○ 설명

사이언스 앱의 실행 파일 경로를 생성. 생성된 경로는 다음의 규칙을 따른다.
{사이언스앱 이름}/{사이언스앱 버전}/{BIN_DIR}/{사이언스앱 실행 파일 이름}

○ Return

- String: 사이언스앱 실행 파일 경로

□ String getSrcPath()

○ 설명

사이언스 앱의 소스 파일 경로를 생성. 생성된 경로는 다음의 규칙을 따른다.
{사이언스앱 이름}/{사이언스앱 버전}/{SRC_DIR}/{사이언스앱 소스 파일 이름}

○ Return

- String: 사이언스앱 소스 파일 경로

VIII. Science Platform Javascript Objects

1. InputPort

□ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>

var portMap = new PortMap();
var port = portMap.createInputPort("inp");
```

□ JSON Format

```
{
   name:{#port-name},
   mandatory:{#true-or-false},
   port-type-id:{#port-type-id},
   default-editor-id:{#editor-id},
   order:{#order} //deprecated
}
```

□ getName

○ 설명

인풋포트의 이름을 얻는다.

○ Return

데이터 형	설명
String	포트 이름

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort("inp");
var portName = port.getName();
```

□ setName

○ 설명

인풋포트의 이름을 설정한다.

○ 인수

데이터 형	설명
String	포트이름

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
```

```
var portMap = new PortMap();
var port = portMap.getPort("inp");
port.setName("inp");
```

□ isMandatory

○ 설명

인풋포트가 사이언스앱 실행에 반드시 필요한가를 조회한다.

○ Return

데이터 형	설명
boolean	포트가 사이언스앱 실행에 반드시 필요하면 true, 그렇지 않으면 false.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort("inp");
var mandatory = port.isMandatory();
```

□ setMandatory

○ 설명

인풋포트가 사이언스앱 실행에 필수적인가의 여부를 설정한다.

○ 인수

데이터 형	설명
boolean	포트가 사이언스앱 실행에 필수적이면 true, 그렇지 않으면 false

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort("inp");
port.setMandatory(true);
```

□ getPortTypeId

○ 설명

인풋포트의 포트 타입을 ID를 얻는다.

○ Return

데이터 형	설명
Number	포트타입이 정의되어 있으면 포트타입의 ID, 그렇지 않으면 "undefined"를 반환한다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort("inp");
var id = port.getPortTypeId();
```

□ setPortTypeId

○ 설명

인풋포트에 포트타입 ID를 설정한다.

○ 인수

데이터 형	설명
Number	포트타입 ID.

□ getDefaultEditorId

○ 설명

인풋포트에 정의된 데이터를 편집할 수 있는 기본 편집기 ID를 얻는다.

○ Return

데이터 형	설명
Number	기본 편집기가 정의되어 있으면 편집기의 ID, 그렇지 않으면 “undefined”를 반환한다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort(“inp”);
var editorId = port.getEditorId();
```

□ setDefaultEditorId

○ 설명

인풋포트에 정의된 데이터를 편집할 수 있는 기본 편집기 아이디를 설정한다.

○ 인수

데이터 형	설명
Number	편집기 ID.

□ getOrder

○ 설명

Deprecated. 인풋포트의 순서를 얻는다.

○ Return

데이터 형	설명
number	순서가 정의되어 있으면 순서를, 그렇지 않으면 “undefined”를 반환한다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort(“inp”);
var order = port.getOrder();
```

□ setOrder

○ 설명

Deprecated. 인풋포트의 순서를 정의한다.

○ 인수

데이터 형	설명
number	인풋포트의 순서

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort("inp");
port.setOrder(1);
```

□ verifyPortName

○ 설명

포트 이름이 명명 규칙에 따랐는 지를 확인한다.

○ 인수

데이터 형	설명
String	검사하려는 포트 이름

○ Return

데이터 형	설명
String	포트 이름이 명명 규칙을 따랐으면 true, 그렇지 않으면 PortErrors.INVALID_PORT_NAME 오류를 반환

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var result = portMap.verifyPortName("inp");
```

□ setData

○ 설명

인풋포트에 데이터를 저장한다.

○ 인수

데이터 형	설명
JSON Object	데이터를 저장하고 있는 JSON 객체. 일반적으로 데이터베이스로부터 얻어온다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
vvar portMap = new PortMap();
var port = portMap.getPort("inp");
var jsonData = yourDatabase.getJsonData();
port.setData(jsonData);
```

2. OutputPort

□ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>

var portMap = new PortMap();
var port = portMap.createOutputPort("out");
```

□ JSON Format

```
{
   name:{#port-name},
   mandatory:{#true-or-false},
   port-type-id:{#port-type-id},
   default-analyzer-id:{#analyzer-id},
   file-name:{#file-name}
}
```

□ getName

○ 설명

@See Inputport.getName()

□ setName

○ 설명

@See Inputport.setName()

□ isMandatory

○ 설명

@See Inputport.isMandatory()

□ setMandatory

○ 설명

@See Inputport.setMandatory()

□ getPortTypeId

○ 설명

@See Inputport.getPortTypeId()

□ setPortTypeId

○ 설명

@See Inputport.setPortTypeId()

□ getDefaultAnalyzerId

○ 설명

아웃풋포트에 정의된 데이터를 가시화하거나 분석할 수 있는 기본 분석기 ID를 얻는다.

○ Return

데이터 형	설명
Number	기본 분석기가 정의되어 있으면 분석기의 ID, 그렇지 않으면 “undefined”를 반환한다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort(“out”);
var analyzerId = port.getAnalyzerId();
```

□ setDefaultAnalyzerId

○ 설명

아웃풋포트에 정의된 데이터를 분석할 수 있는 기본 분석기 ID를 설정한다.

○ 인수

데이터 형	설명
Number	분석기 ID.

□ setData

○ 설명

아웃풋포트에 데이터를 저장한다.

○ 인수

데이터 형	설명
JSON Object	데이터를 저장하고 있는 JSON 객체. 일반적으로 데이터베이스로부터 얻어온다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.getPort(“out”);
var jsonData = yourDatabase.getJsonData();
port.setData(jsonData);
```

3. PortMap

□ 사용 예제

객체에 인풋 포트 또는 아웃풋 포트를 (포트 이름, 포트 객체) 쌍으로 저장하고 관리한다.

```
var portMap = new PortMap();
  var port = portMap.createInputPort("inp");
  if( port == null){
       alert("Port name "+ "inp" + " is not valid...");
  }
  else{
       port.setName("inp");
       port.setMandatory(true);
       var portTypeId = 12345;
```

```
        var editorId = “6758”;
        port.setPortTypeId(portTypeId);
        port.setDefaultEditorId(editorId);
        port.setOrder(1);
    }

    var port2 = portMap.createInputPort("inpTwo");
    if( port2 == null){
        alert("Port name "+ "inpTwo" + " is not valid...");
    }
    else {
        port2.setName("inpTwo");
        port2.setMandatory(false);
        var portTypeId = 54321;
        var editorId = 9876;
        port2.setPortTypeId(portTypeId);
        port2.setDefaultEditorId(edistorId);
        port2.setOrder(2);
    }

    var ary = portMap.getPortArray()
    console.log(JSON.stringify(portMap));
    console.log(JSON.stringify(ary));
    var names = portMap.getPortNameArray();
    console.log(JSON.stringify(names));
    portMap.removePort("inp");
    console.log(JSON.stringify(portMap));
```

□ JSON Format

```
{
  {#port-name} : {#InputPort or OutputPort 객체},
  ......
}
```

□ getPort

○ 설명

포트 이름으로 저장된 포트를 얻는다.

○ 인수

데이터 형	설명
String	얻고자 하는 포트의 이름

○ Return

데이터 형	설명
InputPort or OutputPort	지정된 이름을 가진 포트가 있으면 포트의 객체, 그렇지 않으면 “undefined”를 반환한다.

○ 사용 예제

```
<script src=’path-to-js’/in_out_ports.js></script>
var portMap = new PortMap();
var port = port.getPort(“inp”);
```

□ addPort

○ 설명

포트 객체를 저장한다.

○ 인수

데이터 형	설명
InputPort or OutputPort	저장하고자 하는 포트 객체

○ Return

데이터 형	설명
boolean	포트 객체에 정의된 포트 이름이 유효하고 저장에 성공하면 true, 그렇지 않으면 false를 반환한다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = new Port();
//do something with port
var result = port.addPort(port);
```

□ createInputPort

○ 설명

InputPort 포트 객체를 생성하여 저장한 뒤 반환한다.

○ 인수

데이터 형	설명
String	생성하려는 포트 이름

○ Return

데이터 형	설명
InputPort	지정된 포트 이름이 유효하고 생성 및 저장에 성공하면 생성된 InputPort 객체를, 그렇지 않으면 null을 반환한다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var port = portMap.createInputPort("inp");
//do something with port
```

□ createOutputPort

○ 설명

@see createInputPort()

□ removePort

○ 설명

지정된 이름을 가진 포트를 삭제한다.

○ 인수

데이터 형	설명
String	삭제하려는 포트 이름

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
portMap.removePort("inp");
```

□ getPortArray

○ 설명

저장된 모든 포트 객체를 배열로 얻어온다.

○ Return

데이터 형	설명
Array	저장된 모든 InputPort 또는 OutputPort 객체

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var portArray = portMap.getPortArray();
//do something with portArray
```

□ getPortNameArray

○ 설명

저장된 모든 포트 객체의 이름을 배열로 얻어온다.

○ Return

데이터 형	설명
Array	저장된 모든 포트의 이름

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var portNameArray = portMap.getPortNameArray();
//do something with portNameArray
```

□ isPortNameDulicated

○ 설명

이미 동일한 이름을 가진 포트가 없는 지를 확인한다.

○ 인수

데이터 형	설명
String	검사하려는 포트 이름

○ Return

데이터 형	설명
String	사이언스앱 내에 지정된 포트 이름이 유일하면 true, 그렇지 않으면 PortErrors.DUPLICATED_PORT_NAME 오류를 반환

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var result = portMap.isPortNameDuplicated("inp");
```

□ setData

○ 설명

JSON 데이터를 포트맵 객체에 저장한다.

○ 인수

데이터 형	설명
JSON Object	데이터를 저장하고 있는 JSON 객체. 일반적으로 데이터베이스로부터 얻어온다.

○ 사용 예제

```
<script src='path-to-js'/in_out_ports.js></script>
var portMap = new PortMap();
var jsonData = yourDatabase.getJsonData();
portMap.setData(jsonData);
```

4. InputdeckForm

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>

var inputdeckForm = new InputdeckForm();
```

□ JSON Format

```
{
    "vector-form" :
    {
        "brace-char" : "#{brace-char}",
        "space" : "#{true-or-not}",//deprecated
        "delimiter" : "#{delimiter-char}"
        "sample" : "#{sample}"
    },
    "line-form" :
    {
        "value-delimiter" : "#{delimiter-char}",
        "space" : "#{true-or-not}", //deprecated
        "line-delimiter" : "#{delimiter-char}",
        "comment-char" : "#{comment-char}"
    },
```

```
    "available-language-ids" :
    [
        "language-id",
        ##..... any more data .....##
    ],
    "default-language" : "{#language-id}",
    variable-map : {#VariableMap Object}
}
```

□ Attribute Definition

카테고리	인수 명	데이터 형	기본값	설명
vector-form	brace-char	String	SQUARE, SQUARE-SPACE, ROUND, ROUND-SPACE	vector bracket 기호
	space	Boolean	TRUE or FALSE	스페이스 사용 여부, deprecated 예정
	delimiter	String	SPACE or COMMA	vector 아이템 구분자
	sample	String	-	vector form이 적용된 샘플 문자열
line-form	value-delimiter	String	SPACE or EQUAL	대입 연산 기호
	space	Boolean	TRUE or FALSE	스페이스 사용 여부, deprecated 예정
	line-delimiter	String	₩n, SEMICOLON, +₩n₩r	line 구분자
	comment-char	String	-	주석 기호
available-language-ids	language-id	Number	-	사용 가능한 언어 ID
default-language	language-id	Number	-	기본 사용 언어 ID
variable-map (key)	variable-name	String	-	variable-map의 key 값이 되는 이름, variable-map에 속한 value의 하위 노드인 name과 값이 동일
variable-map (value)	name	String	-	파라미터 이름, 해당 노드부터 variable-map의 속한 value로 사용
	name-text-map (language-id)	Number	-	파라미터 이름에 Locale 정보를 반영하기 위한 Language ID
	name-text-map (text)	String	-	ID에 따른 디스플레이 텍스트 정보
	type	String	String, Number, Vector, List, Group, Comment	파라미터의 타입 정보, 타입 정보의 따라 variable-map에서 사용되는 노드 정보가 변동
	activate-conditio	Array	-	변수활성화조건을 담고있는 배열 형식의 컨테이너,

	n-container			담고있는 배열의 조건을 충족해야지만 연관된 파라미터가 활성화
activate-condition-container	variable-name	String	-	변수활성화조건 이름
	list-item-value	String	-	파라미터 Type정보가 List일 경우, 리스트이 선택 값과 list-item-value가 동일해야만 변수활성화
	min	Number	-	파라미터 Type정보가 Number일 경우, Number 값이 min값의 min-operand의 조건을 충족시켜야만 변수활성화
	max	Number	-	파라미터 Type정보가 Number일 경우, Number 값이 max값의 max-operand의 조건을 충족시켜야만 변수활성화
	operand	String	==,<=,=>,<>	최소값,최대값의 operand 조건
variable-map (value)	value	String	-	파라미터의 기본값
	value-domain (low-limit)	Number	-	파라미터 Type정보가 Number일 경우, 기본값 value의 최소값
	value-domain (upper-limit)	Number	-	파라미터 Type정보가 Number일 경우, 기본값 value의 최대값
	value-domain (operand)	String	==,<=,=>,<>	파라미터 Type정보가 Number일 경우, 기본값 value의 limit operand 조건
	list-map	Array	-	파라미터 Type정보가 List일 경우, 리스트의 각 Item 이름과 value 정보를 가진다
	list-map (value)	String	-	List Item의 값
	list-map (localized-text-map)	String	-	List Item의 이름으로 Locale 정보를 표현하기 위한 Language ID와 Text 정보를 가진다.
	list-map (activate-condition-container)	String	-	파라미터의 activate-condition-container와 동일, List Item에 각각 적용
	sweep-domain	Array	-	파라미터 Type정보가 Number일 경우, Sweep의 대한 정보를 가진다
	sweep-domain (low-limit)	Number	-	파라미터 Type정보가 Number일 경우, Sweep의 최소값
	sweep-domain (upper-limit)	Number	-	파라미터 Type정보가 Number일 경우, Sweep의 최대값
	sweep-domain (operand)	String	==,<=,=>,<>	파라미터 Type정보가 Number일 경우, Sweep의 최소값, 최대값에 대한 operand 정보. 연산기호 2개로 이루어져있다. 앞열은 최소값에 대한 기호이며, 뒷열은 최대값의 대한 기호이다. 만약 "=>"의 경우 최소값은 이상을 말하고, 최대값은 미만을 뜻한다. "=" 여부에 따라 해당 값을 포함한 범위인지를 판별한다.
	order	Number	-	파라미터 순서를 위한 값, deprecated 예정

	description-map (language-id)	Number	-	Description 정보에 Locale 정보를 반영하기 위한 Language ID
	description-map (text)	String	-	ID에 따른 디스플레이 텍스트 정보
	dimension	Number	1	벡터의 기본 크기

□ getVectorForm

○ 설명

InputdeckForm 객체에 저장된 VectorForm 객체가 있으면 기존의 객체를 반환하고, 만약 존재하지 않으면 새로운 객체를 생성하여 저장한 뒤 반환한다. 특별한 상황이 아니면 직접적인 사용을 권장하지 않음.

○ Return

데이터 형	설명
VectorForm	VectorForm 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
```

□ setVectorForm

○ 설명

VectorForm 객체를 InputdeckForm 객체에 저장. 특별한 상황이 아니면 직접적인 사용을 권장하지 않음.

○ 인수

데이터 형	설명
VectorForm	VectorForm 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = new VectorForm();
inputdeckForm.setVectorForm(vectorForm);
```

InputdeckForm 메소드인 getVectorForm()을 사용하여 VectorForm 객체를 얻으면, 생성된 객체는 InputdeckForm 객체에 자동으로 저장되어 반환된다. 이렇게 반환된 객체에 작업을 수행하면 모든 작업 내용은 모두 InputdeckForm 객체에 자동으로 저장된다. 따라서 기본적으로 setVectorForm() 메소드를 사용할 특별한 상황이 아니면 이 메소드 사용을 권장하지 않는다.

□ setVectorFormValues

○ 설명

InputdeckForm 객체에 저장된 VectorForm 객체의 프로퍼티 값을 설정한다. 만일 InputdeckForm 객체에 저장된 VectorForm 객체가 없으면 새로운 VectorForm 객체를 생성한 뒤 프로퍼티들을 저장한다.

○ 인수

데이터 형	설명
String	괄호 문자
String	벡터값을 구분하는 구분문자
String	샘플 문자열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
inputdeckForm.setVectorFormValues("{", ",", "{1,2,3}");
```

□ getVectorFormBraceChar

○ 설명

벡터 표현에 사용할 괄호 문자를 얻는다.

○ Return

조건	데이터 형	설명
괄호 문자가 정의되어 있을 때	String	괄호 문자
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
var braceChar = inputdeckForm.getVectorFormBraceChar();
```

□ setVectorFormBraceChar

○ 설명

벡터 표현에 사용될 괄호 문자를 지정한다.

○ 인수

데이터 형	설명
String	필터링할 타입 이름

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
inputdeckForm.setVectorFormBraceChar("{");
```

□ getVectorFormDelimiter

○ 설명

벡터 표현에서 값을 구분할 구분 문자를 얻는다.

○ Return

조건	데이터 형	설명
구분 문자가 정의되어 있을 때	String	구분 문자
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
```

```
var braceChar = inputdeckForm.getVectorFormDelimiter();
```

□ setVectorFormDelimiter

○ 설명

벡터 표현에서 값을 구분할 구분문자를 지정한다.

○ 인수

데이터 형	설명
String	구분문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
inputdeckForm.setVectorFormDelimiter(",");
```

□ isVectorFormSpace

○ 설명

객체 표현에 공백 문자를 사용할지 여부를 얻는다. 공백문자를 사용한 벡터 표현은 "{ 1, 2, 3 }" 이며, 공백 문자를 사용하지 않은 벡터 표현은 "{1,2,3}" 이다.

○ Return

조건	데이터 형	설명
스페이스 사용여부가 정의되어 있거나, 공백을 사용하면	boolean	true
그렇지 않으면	boolean	false

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
var space = InputdeckForm.isVectorFormSpace();

if( space == false ){
   //vector form does not use space such as {1,2,3}.
}
else if( space == true ){
   //vector form use space such as { 1, 2, 3 }.
}
```

□ setVectorFormSpace

○ 설명

벡터 표현에서 값을 공백문자를 사용할 지 여부를 지정한다.

○ 인수

데이터 형	설명
boolean	공백문자 사용여부

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
inputdeckForm.setVectorFormSpace(true);
```

□ getVectorFormSample

○ 설명

벡터 표현에서 샘플표현을 얻는다.

○ Return

조건	데이터 형	설명
샘플 표현이 있을 때	String	샘플 문자열
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
var sample = inputdeckForm.getVectorFormSample();
```

□ setVectorFormSample

○ 설명

벡터 표현의 샘플을 저장한다.

○ 인수

데이터 형	설명
String	샘플 벡터 표현

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
inputdeckForm.setVectorFormSample(“{1,2,3}”);
```

□ getLineForm

○ 설명

InputdeckForm 객체에 저장된 LineForm 객체가 있으면 기존의 객체를 반환하고, 만약 존재하지 않으면 새로운 객체를 생성하여 저장한 뒤 반환한다. 특별한 상황이 아니면 직접적인 사용을 권장하지 않음.

○ Return

데이터 형	설명
LineForm	LineForm 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
```

□ setLineForm

○ 설명

LineForm 객체를 InputdeckForm 객체에 저장. 특별한 상황이 아니면 직접적인 사용을 권장하지 않음.

○ 인수

데이터 형	설명
LineForm	LineForm 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = new LineForm();
inputdeckForm.setLineForm(lineForm);
```

InputdeckForm 메소드인 getLineForm()을 사용하여 LineForm 객체를 얻으면, 생성된 객체는 InputdeckForm 객체에 자동으로 저장되어 반환된다. 이렇게 반환된 객체에 작업을 수행하면 모든 작업 내용은 모두 InputdeckForm 객체에 자동으로 저장된다. 따라서 기본적으로 setLineForm() 메소드를 사용할 특별한 상황이 아니면 이 메소드 사용을 권장하지 않는다.

□ setLineFormValues

○ 설명

InputdeckForm 객체에 저장된 LineForm 객체의 프로퍼티 값을 설정한다. 만일 InputdeckForm 객체에 저장된 LineForm 객체가 없으면 새로운 LineForm 객체를 생성한 뒤 프로퍼티들을 저장한다.

○ 인수

데이터 형	설명
String	변수와 값을 구분하는 구분문자
String	라인을 구분하는 구분문자
String	코멘트를 시작하는 라인의 첫문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
inputdeckForm.setLineFormValues("=", ";", "#");
```

□ getLineFormValueDelimiter

○ 설명

인풋덱 파일에서 변수 명과 값 사이의 구분 문자를 얻는다.

○ Return

조건	데이터 형	설명
구분 문자가 정의되어 있을 때	String	구분 문자
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
var valueDelimiter = inputdeckForm.getLineFormValueDelimiter();
```

□ setLineFormValueDelimiter

○ 설명

인풋덱에서 변수 명과 값 사이에 구분 문자로 사용할 문자를 지정한다.

○ 인수

데이터 형	설명
String	구분문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
inputdeckForm.setLineFormValueDelimiter("=");
```

□ getLineFormLineDelimiter

○ 설명

인풋덱에서 라인을 구분할 구분 문자를 얻는다.

○ Return

조건	데이터 형	설명
구분 문자가 정의되어 있을 때	String	구분 문자
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
var lineDelimiter = inputdeckForm.getLineFormLineDelimiter();
```

□ setLineFormLineDelimiter

○ 설명

인풋덱에서 라인을 구분할 때 사용할 구분문자를 지정한다.

○ 인수

데이터 형	설명
String	구분문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
inputdeckForm.setLineFormLineDelimiter(";");
```

□ isLineFormSpace

○ 설명

인풋덱에서 변수명, 구분문자, 값 사이에에 공백 문자를 사용할지 여부를 얻는다. 공백문자를 사용한 (변수명, 값) 쌍의 표현은 "varName = val" 이며, 공백 문자를 사용하지 않은 표현은 "varName=val" 이다.

○ Return

조건	데이터 형	설명
스페이스 사용여부가 정의되어 있거나, 공백을 사용하면	boolean	true
그렇지 않으면	boolean	false

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
```

```
var space = InputdeckForm.isLineFormSpace();

if( space == false ){
   //line form does not use space such as 'varName=123'.
}
else if( space == true ){
   //line form use space such as 'varName = 123'.
}
```

□ setLineFormSpace

○ 설명

인풋덱에서 (변수명, 값) 상을 표현하는 데 공백문자를 사용할 지 여부를 지정한다.

○ 인수

데이터 형	설명
boolean	공백문자 사용여부

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
inputdeckForm.setLineFormSpace(true);
```

□ getLineFormCommentChar

○ 설명

인풋덱에서 사용할 코멘트 라인의 첫 시작 문자를 얻는다.

○ Return

조건	데이터 형	설명
시작문자가 있을 때	String	시작문자
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
var commentChar = inputdeckForm.getLineFormCommentChar();
```

□ setLineFormComentChar

○ 설명

인풋덱에서 사용할 코멘트 라인의 첫 시작 문자를 지정한다.

○ 인수

데이터 형	설명
String	코멘트 시작 문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inpuitdeckForm = new InputdeckForm();
inputdeckForm.setLineFormCommentChar("#");
```

□ getAvailableLanguageIds

○ 설명

InputdeckForm 객체에 저장된 사용 가능한 언어 아이디를 배열 객체로 반환한다.

○ Return

데이터 형	설명
Array	사용 가능한 언어 아이디를 문자열로 저장한 배열 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var availableLanguageIds = inputdeckForm.getAvailableLanguageIds();
```

□ setAvailableLanguageIds

○ 설명

InputdeckForm 객체에 사용 가능한 언어 아이디를 저장한다.

○ 인수

데이터 형	설명
LineForm	LineForm 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
<%
   Locale[] locales = LanguageUtil.getAvailableLocales();

   JSONArray jsonLocales = JSONFactoryUtil.createJSONArray();
   for(Locale al : locales){
        jsonLocales.put(al.toString());
   }
%>

var inputdeckForm = new InputdeckForm();
var availableLanguageIds = <%= jsonLocales.toString() %>;

inputdeckForm.setLineForm(lavailableLanguageIds);
```

□ getDefaultLanguageId

○ 설명

InputdeckForm 객체에 저장된 기본 언어 아이디를 얻는다.

○ Return

조건	데이터 형	설명
InputdeckForm 객체에 기본 언어가 저장되어 있을 때	String	기본 언어 아이디
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
```

```
var inputdeckForm = new InputdeckForm();
var defaultLanguageId = inputdeckForm.getDefaultLanguageId();
```

□ setDefaultLanguageId

○ 설명

InputdeckForm 객체에 기본 언어 아이디를 저장한다.

○ 인수

데이터 형	설명
String	언어 아이디

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
inputdeckForm.setDefaultLanguageId("ko_KR");
```

□ getVariableMap

○ 설명

InputdeckForm 객체에 저장된 변수를 저장한 맵 객체를 얻는다. 특별한 상황이 아니면 직접적인 사용을 권장하지 않음.

○ Return

조건	데이터 형	설명
InputdeckForm 객체에 변수맵 객체가 저장되어 있을 때	VariableMap	변수를 저장한 Map 객체
그렇지 않으면	VariableMap	{}, 빈 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
```

□ setVariableMap

○ 설명

InputdeckForm 객체에 저장된 변수를 저장한 맵 객체를 저장한다. 특별한 상황이 아니면 직접적인 사용을 권장하지 않음.

○ 인수

데이터 형	설명
VariableMap	VariableMap 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = new VariableMap();
//Do something.....
inputdeckForm.setVariableMap(variableMap);
```

□ getVariable

○ 설명

InputdeckForm 객체에 저장된 변수 객체를 얻는다.

○ 인수

데이터 형	설명
String	변수 이름

○ Return

조건	데이터 형	설명
변수 이름을 가진 객체가 있을 때	Variable	변수 객체
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable(“var_1”);

if( variable == “undefined” )
   //there is no variable named “var_1”
else{
   //do something with variable
}
```

□ addVariable

○ 설명

InputdeckForm 객체에 변수 객체를 저장한다. 추가될 변수 객체의 이름이 명명 규칙에 맞지 않거나, 이미 동일한 이름의 변수 객체가 존재하면 변수를 추가하지 않는다.

○ 인수

데이터 형	설명
Variable	저장할 Variable 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = new Variable();

//do something with variable

inputdeckForm.addVariable(variable);
```

□ createVariable

○ 설명

InputdeckForm 객체에 변수 객체를 생성하여 저장한다.

○ 인수

데이터 형	설명
String	생성할 Variable 객체 변수 이름

○ Return

조건	데이터 형	설명
생성할 변수 객체 이름이 명명 규칙에 맞고 동일한 이름을 가진 변수 객체가 존재하지 않을 때	Variable	변수 객체
그렇지 않으면	null	

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.createVariable("var_1");

if( variable == null )
  //변수 이름이 잘못 명명되었음
else{
  //do something with variable
}
```

□ removeVariable

○ 설명

InputdeckForm 객체에 저장된 변수 객체를 삭제한다.

○ 인수

데이터 형	설명
String	삭제할 Variable 객체 이름

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
inputdeckForm.removeVariable("var_1");
```

□ cloneVariable

○ 설명

InputdeckForm 객체에 저장된 변수 객체를 정해진 횟수만큼 복제한다.

○ 인수

데이터 형	설명
Variable	복제할 Variable 객체
Number	복제 횟수

○ Return

조건	데이터 형	설명
복제에 성공했을 때	Array	복제된 Variable 객체 배열
그렇지 않으면	Object	null

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("varName");
var clones = inputdeckForm.cloneVariable(variable, 5);
```

☐ cloneVariableGroup

○ 설명

타입이 GROUP인 변수와 그룹에 속한 변수들을 복제한다

○ 인수

데이터 형	설명
Variable	복제할 Variable 객체

○ Return

조건	데이터 형	설명
복제에 성공했을 때	Variable	복제된 group 타입의 Variable 객체
그렇지 않으면	Object	null

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("varName");
var cloneGroup = inputdeckForm.cloneVariableGroup(variable);
```

☐ getVariableNames

○ 설명

InputdeckForm 객체에 저장된 모든 변수 객체의 이름을 문자열 배열로 얻는다.

○ Return

조건	데이터 형	설명
	Array	변수 이름 배열 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableNames = inputdeckForm.getVariableNames();
```

☐ getVariableNamesByType

○ 설명

InputdeckForm 객체에 저장된 특정한 타입의 모든 변수 객체의 이름을 문자열 배열로 얻는다.

○ 인수

데이터 형	설명
String	필터링할 타입 이름

○ Return

조건	데이터 형	설명
	Array	특정한 타입을 가진 변수 이름 배열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
```

```
var names = inputdeckForm.getVariableNamesByType(InputdeckConstants.VARIABLE_TYPE_NUMERIC);
```

□ setData

○ 설명

InputdeckForm 객체에 JSON 객체를 데이터로 추가한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var jsonData = getDataFromService();
var result = inputdeckForm.setData(jsonData);
```

5. VectorForm

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
```

□ JSON Format

```
{
    "brace-char" : "#{brace-char}",
    "space" : "#{one of SQUARE, SQUARE_SPACE, ROUND, ROUND_SPACE}",//deprecated
    "delimiter" : "#{delimiter-char}"
    "sample" : "#{sample}"
},
```

□ setValues

○ 설명

VectorForm 객체의 프로퍼티 값을 설정한다.

○ 인수

데이터 형	설명
String	괄호 문자
String	벡터값을 구분하는 구분문자
String	샘플 문자열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
vectorForm.setValues("{", ",", "{1,2,3}");
```

□ getBraceChar

○ 설명

벡터 표현에 사용할 괄호 문자를 얻는다.

○ Return

조건	데이터 형	설명
괄호 문자가 정의되어 있을 때	String	괄호 문자
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
var braceChar = vectorForm.getBraceChar();
```

□ setBraceChar

○ 설명

벡터 표현에 사용될 괄호 문자를 지정한다.

○ 인수

데이터 형	설명
String	필터링할 타입 이름

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
vectorForm.setBraceChar("{");
```

□ getDelimiter

○ 설명

벡터 표현에서 값을 구분할 구분 문자를 얻는다.

○ Return

조건	데이터 형	설명
구분 문자가 정의되어 있을 때	String	구분 문자
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
var delimiter = vectorForm.getDelimiter();
```

□ setDelimiter

○ 설명

벡터 표현에서 값을 구분할 구분문자를 지정한다.

○ 인수

데이터 형	설명
String	구분문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
vectorForm.setDelimiter(",");
```

□ getSpace

○ 설명

벡터 표현에서 괄호 다음에 공백 문자를 사용할지 여부를 얻는다.

○ Return

조건	데이터 형	설명
스페이스 사용여부가 정의되어 있을 때	String	SQUARE SQUARE_SPACE ROUND ROUND_SPACE 중 하나
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
var space = vectorForm.getSpace();

if( space == InputdeckConstants.BRACE_TYPE_SQUARE){
}
else if( space == InputdeckConstants.BRACE_TYPE_ROUND){
}
else if( space == InputdeckConstants.BRACE_TYPE_SQUARE_SPACE ){
}
else if( space == InputdeckConstants.BRACE_TYPE_ROUND_SPACE ){
}
else{
}
```

□ setSpace

○ 설명

벡터 표현에서 값을 공백문자를 사용할 지 여부를 지정한다.

○ 인수

데이터 형	설명
boolean	공백문자 사용여부

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
vectorForm.setSpace(InputdeckConstants.BRACE_TYPE_SQUARE);
```

□ getSample

○ 설명

벡터 표현에서 샘플표현을 얻는다.

○ Return

조건	데이터 형	설명
샘플 표현이 있을 때	String	샘플 문자열
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
var sample = vectorForm.getSample();
```

□ setSample

○ 설명

벡터 표현의 샘플을 저장한다.

○ 인수

데이터 형	설명
String	샘플 벡터 표현

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
vectorForm.setSample(“{1,2,3}”);
```

□ setData

○ 설명

VecterForm 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var vectorForm = inputdeckForm.getVectorForm();
var jsonData = getDataFromService();
var result = vectorForm.setData(jsonData);
```

6. LineForm

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
```

□ JSON Format

```
{
    value-delimiter: #{delimiter-char},
    space: #{true-or-not}, //deprecated
    line-delimiter:#{delimiter-char},
    comment-char:#{comment-char}
}
```

□ setValues

○ 설명

LineForm 객체의 프로퍼티 값을 설정한다.

○ 인수

데이터 형	설명
String	변수와 값을 구분하는 구분문자
String	라인을 구분하는 구분문자
String	코멘트를 시작하는 라인의 첫문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
lineForm.setValues("{", ",", "{1,2,3}");
```

□ getValueDelimiter

○ 설명

인풋덱 파일에서 변수 명과 값 사이의 구분 문자를 얻는다.

○ Return

조건	데이터 형	설명
구분 문자가 정의되어 있을 때	String	구분 문자
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
var valueDelimiter = lineForm.getDelimiter();
```

□ setValueDelimiter

○ 설명

인풋덱에서 변수 명과 값 사이에 구분 문자로 사용할 문자를 지정한다.

○ 인수

데이터 형	설명
String	구분문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
lineForm.setValueDelimiter(“=”);
```

□ getLineDelimiter

○ 설명

인풋덱에서 라인을 구분할 구분 문자를 얻는다.

○ Return

조건	데이터 형	설명
구분 문자가 정의되어 있을 때	String	구분 문자
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
var lineDelimiter = lineForm.getLineDelimiter();
```

□ setLineDelimiter

○ 설명

인풋덱에서 라인을 구분할 때 사용할 구분문자를 지정한다.

○ 인수

데이터 형	설명
String	구분문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
lineForm.setLineDelimiter(";");
```

□ isSpace

○ 설명

인풋덱에서 변수명, 구분문자, 값 사이에에 공백 문자를 사용할지 여부를 얻는다. 공백문자를 사용한 (변수명, 값) 쌍의 표현은 "varName = val" 이며, 공백 문자를 사용하지 않은 표현은 "varName=val" 이다.

○ Return

조건	데이터 형	설명
스페이스 사용여부가 정의되어 있거나, 공백을 사용하면	boolean	true
그렇지 않으면	boolean	false

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
var space = lineForm.isSpace();

if( space == false ){
   //line form does not use space such as 'varName=123'.
}
else if( space == true ){
   //line form use space such as 'varName = 123'.
}
```

□ setSpace

○ 설명

인풋덱에서 (변수명, 값) 상을 표현하는 데 공백문자를 사용할 지 여부를 지정한다.

○ 인수

데이터 형	설명
boolean	공백문자 사용여부

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
lineForm.setSpace(true);
```

□ getCommentChar

○ 설명

인풋덱에서 사용할 코멘트 라인의 첫 시작 문자를 얻는다.

○ Return

조건	데이터 형	설명
시작문자가 있을 때	String	시작문자
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
var commentChar = lineForm.getCommentChar();
```

□ setComentChar

○ 설명

인풋덱에서 사용할 코멘트 라인의 첫 시작 문자를 지정한다.

○ 인수

데이터 형	설명
String	코멘트 시작 문자

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
lineForm.setCommentChar(“#”);
```

□ setData

○ 설명

LineForm 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var lineForm = inputdeckForm.getLineForm();
var jsonData = getDataFromService();
var result = lineForm.setData(jsonData);
```

7. VariableMap

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var map = inputdeckForm.getVariableMap();
```

□ JSON Format

```
{
    {#variable-name}: {#Variable Object},
   ......
 }
```

□ getVariableMap

○ 설명

변수를 저장한 맵 객체를 얻는다. 직접적인 호출을 권장하지 않음.

○ Return

조건	데이터 형	설명
변수맵 객체가 있을 때	Object	변수를 저장한 Map 객체
그렇지 않으면	Object	{}, 빈 객체

V반환된 Map 객체에는 변수 객체가 '변수명' : Variable 객체의 형식으로 저장된다.

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
var jsonData = variableMap.getVariableMap();
```

□ setVariableMap

○ 설명

변수를 저장한 맵 객체를 저장한다. 직접적인 호출을 권장하지 않음.

○ 인수

데이터 형	설명
Object	Object 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
var map = {};
//Do something with map.....
variableMap.setVariableMap(map);
```

□ getVariable

○ 설명

저장된 변수 객체를 얻는다.

○ 인수

데이터 형	설명
String	변수 이름

○ Return

조건	데이터 형	설명
변수 이름을 가진 객체가 있을 때	Variable	변수 객체
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
//Do something
var variable = inputdeckForm.getVariable(“var_1”);

if( variable == “undefined” )
   //there is no variable named “var_1”
else{
   //do something with variable
}
```

□ addVariable

○ 설명

변수 객체를 저장한다. 추가될 변수 객체의 이름이 명명 규칙에 맞지 않거나, 이미 동일한 이름의 변수 객체가 존재하면 변수를 추가하지 않는다.직접적인 호출을 권장하지 않음.

○ 인수

데이터 형	설명
Variable	저장할 Variable 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
var variable = new Variable();

//do something with variable

variableMap.addVariable(variable);
```

□ createVariable

○ 설명

변수 객체를 생성하여 저장한다.

○ 인수

데이터 형	설명
String	생성할 Variable 객체 변수 이름

○ Return

조건	데이터 형	설명
생성할 변수 객체 이름이 명명 규칙에 맞고 동일한 이름을 가진 변수 객체가 존재하지 않을 때	Variable	변수 객체
그렇지 않으면	null	

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
var variable = variableMap.createVariable("var_1");

if( variable == null )
   //변수 이름이 잘못 명명되었음
else{
   //do something with variable
}
```

□ removeVariable

○ 설명

저장된 변수 객체를 삭제한다.

○ 인수

데이터 형	설명
String	삭제할 Variable 객체 이름

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
variableMap.removeVariable("var_1");
```

□ getVariableNames

○ 설명

저장된 모든 변수 객체의 이름을 문자열 배열로 얻는다.

○ Return

조건	데이터 형	설명
	Array	변수 이름 배열 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
var variableNames = variableMap.getVariableNames();
```

□ getVariableNamesByType

○ 설명

저장된 특정한 타입의 모든 변수 객체의 이름을 문자열 배열로 얻는다.

○ 인수

데이터 형	설명
String	필터링할 타입 이름

○ Return

조건	데이터 형	설명
	Array	특정한 타입을 가진 변수 이름 배열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
var names = variableMap.getVariableNamesByType(InputdeckConstants.VARIABLE_TYPE_NUMERIC);
```

□ setData

○ 설명

VariableMap 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variableMap = inputdeckForm.getVariableMap();
var jsonData = getDataFromService();
var result = map.setData(jsonData);
```

8. Sweep

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
variable.setSweepable(true);
//Do something with variable
```

□ JSON Format

```
{
    sweepable:{#true-or-not},
    slice:{#slice value},
    domain:{#Domain Object}
}
```

□ getDomain

○ 설명

Sweep의 범위를 지정한 Domain 객체를 얻는다. 직접적인 호출은 반드시 필요한 경우가 아니면 하지 않는다.

○ Return

조건	데이터 형	설명
Sweep 도메인이 정의되어 있을 때	Domain	Domain 객체
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var sweep = variable.getSweep();
var domain = sweep.getDomain();
```

□ setDomain

○ 설명

Sweep의 범위를 저장한 Domain 객체를 설정한다. 직접적인 호출은 하지 않는다.

○ 인수

데이터 형	설명
Domain	Domain 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var sweep = variable.getSweep();
var domain = sweep.getDomain();
//Do something with domain.....
sweep.setDomain(domain);
```

□ getSlice

○ 설명

Sweep의 범위를 어떤 간격으로 나눌 지를 조회한다.

○ Return

조건	데이터 형	설명
나눔 값이 젖의되어 있으면	number	나눔값
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var sweep = variable.getSweep();
var slice = sweep.getSlice();
```

□ setSlice

○ 설명

Sweep의 범위를 나눌 간격을 설정한다.

○ 인수

데이터 형	설명
Number	나눔값

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var sweep = variable.getSweep();
sweep.setSlice(0.1);
```

□ isSweepable

○ 설명

변수가 Sweep 가능한 지 아닌 지를 조회한다

○ Return

조건	데이터 형	설명
Sweep 도메인이 정의되어 있을 때	boolean	true
그렇지 않으면	boolean	false

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var sweep = variable.getSweep();
var sweepable = sweep.isSweepable();
```

☐ getDomainLowerLimit

@See Domain.getLowerLimit()

☐ setDoaminLowerLimit

@See Domain.setLowerLimit()

☐ getDomainUpperLimit

@See Domain.getUpperLimit()

☐ setDomainUpperLimit

@See Domain.setUpperLimit()

☐ getDomainOperand

@See Domain.getOperand()

☐ setDomainOperand

@See Domain.setOperand()

☐ containDomainLowerLimit

@See Domain.containLowerLimit()

☐ containDomainUpperLimit

@See Domain.containUpperLimit()

☐ isDomainLowerLimitContained

@See Domain.isLowerLimitContained()

☐ isDomainUpperLimitContained

@See Domain.isUpperLimitContained()

☐ setDomainValues

@See Domain.setValues()

☐ isInDomain

@See Domain.isInDomain()

9. Variable

☐ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
//Do something with variable
```

□ JSON Format

```
{
    name:{#name},
    name-text-map:{#LocalizedTextMap Object},
    type:{#variable-type} // string, numeric, vector, list, group, file(deprecated)
    activate-condition-container:{#ActivateConditionContainer Object},
    value:{#value},
    value-domain:{#Domain Object},
    unit : {#unit},
    list-map:{#ListMap Object},
    sweep-domain:{#Domain Object},
    order:{#order}, //deprecated
    description-map: {#LocalizedTextMap Object}
    dimension:{#vector-dimension} //default = 1
}
```

□ getName

○ 설명

변수 명을 얻는다.

○ Return

조건	데이터 형	설명
변수 명이 있을 때	String	변수 명
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var name = variable.getName();
```

□ setName

○ 설명

변수의 이름을 설정한다.

○ 인수

데이터 형	설명
String	변수 이름

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
variable.setName(“var_1”);
```

□ getNameTextMap

○ 설명

변수 이름에 대 한 별칭을 (언어 아이디 : 텍스트) 쌍의 맵으로 얻는다.

○ Return

조건	데이터 형	설명
데이터가 정의되어 있을 경우	LocalizedTextMap	데이터를 포함한 객체
그렇지 않으면	LocalizedTextmap	빈 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var map = variable,getNameTextMap();
```

□ setNameTextMap

○ 설명

Variable 객체에 LocalizedTextMap 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
LocalizedTextMap	데이터를 저장하고 있는 LocalizedTextMap 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var map = new LocalizedTextMap();
variable.setNameTextMap(map);
```

□ getNameLocalizedText

@See LocalizedTextMap.getLocalizedText()

□ addNameLocalizedText

@See LocalizedTextMap.addLocalizedText()

□ removeNameLocalizedText

@See LocalizedTextMap.removeLocalizedText()

□ getNameLocalizedXML

@See LocalizedTextMap.getLocalizedXML()

□ cleanNameText

@See LocalizedTextMap.clean()

□ getType

○ 설명

변수 타입을 얻는다.

○ Return

조건	데이터 형	설명
변수 타입이 정의되어 있을 때	String	변수 타입 numeric, string, vector, list, group, comment, file 중 하나
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var type = variable.getType();
```

□ setType

○ 설명

변수의 타입을 설정한다.

○ 인수

데이터 형	설명
String	변수 타입. numeric, string, vector, list, group, comment, file 중 하나

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
variable.setType(InputdeckConstants.VARIABLE_TYPE_NUMERIC);
```

□ getActivateConditionContainer

○ 설명

변수의 활성화 조건인 ActivateCondition 객체들을 저장한 ActivateConditionContainer 객체를 얻는다.

○ Return

데이터 형	설명
ActivateConditionContainer	ActivateConditionContainer 객체가 형성되어 있으면 그 객체를 반환하고, 그렇지 않으면 객체를 생성하여 저장한 뒤 반환한다.

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
```

```
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var container = variable.getActivateConditionContainer();
//Do something with container
```

☐ setActivateConditionContainer

○ 설명

변수의 활성화 조건인 ActivateCondition 객체들을 저장한 ActivateConditionContainer 객체를 설정한다.

○ 인수

데이터 형	설명
ActivateConditionContainer	ActivateConditionContainer 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var container = new ActivateConditionContainer();
variable.setActivateConditionContainer(container);
```

☐ addActivateCondition

@See ActivateConditionContainer.addActivateCondition()

☐ addListItemActivateCondition

@See ActivateConditionContainer.addListItemActivateCondition

☐ addNumericActivateCondition

@See ActivateConditionContainer.addNumericActivateCondition

☐ removeActivateConditions

@See ActivateConditionContainer.removeActivateCondition

☐ removeListItemActivateCondition

@See ActivateConditionContainer.removeListItemActivateCondition

☐ removeNumericActivateCondition

@See ActivateConditionContainer.removeNumericActivateCondition

☐ getValue

○ 설명

변수의 값을 얻는다.

○ Return

조건	데이터 형	설명
변수 값이 정의되어 있을 때	?	변수 값
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var value = variable.getValue();
```

□ setValue

○ 설명

변수의 값을 설정한다.

○ 인수

데이터 형	설명
?	변수 값

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
variable.setValue(‘1.24e-4’);
```

□ getUnit

○ 설명

변수의 값의 단위를 얻는다.

○ Return

조건	데이터 형	설명
변수 값이 정의되어 있을 때	String	변수 값의 단위
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var unit = variable.getUnit();
```

□ setUnit

○ 설명

변수 값의 단위를 설정한다.

○ 인수

데이터 형	설명
String	변수 값의 단위

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
variable.setUnit('Å');
```

□ getValueDomain

○ 설명

변수의 타입이 numeric일 때, 값의 범위를 저장한 Domain 객체를 얻는다.

○ Return

조건	데이터 형	설명
변수 도메인이 정의되어 있을 때	Domain	Domain 객체
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var domain = variable.getValueDomain();
```

□ setValueDomain

○ 설명

변수 값의 범위를 저장한 Domain 객체를 설정한다.

○ 인수

데이터 형	설명
Domain	Domain 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var domain = variable.getValueDomain();
//Do something with domain....
variable.setValueDomain(domain);
```

□ getValueDomainLowerLimit

@See Domain.getLowerLimit()

☐ setValueDoaminLowerLimit

@See Domain.setLowerLimit()

☐ getValueDomainUpperLimit

@See Domain.getUpperLimit()

☐ setValueDomainUpperLimit

@See Domain.setUpperLimit()

☐ getValueDomainOperand

@See Domain.getOperand()

☐ setValueDomainOperand

@See Domain.setOperand()

☐ containValueDomainLowerLimit

@See Domain.containLowerLimit()

☐ containValueDomainUpperLimit

@See Domain.containUpperLimit()

☐ isValueDomainLowerLimitContained

@See Domain.isLowerLimitContained()

☐ isValueDomainUpperLimitContained

@See Domain.isUpperLimitContained()

☐ setValueDomainValues

@See Domain.setValues()

☐ isInValueDomain

@See Domain.isInDomain()

☐ getListMap

○ 설명

변수의 타입이 list일 때, 리스트 항목들을 저장한 ListMap 객체를 얻는다.

○ Return

조건	데이터 형	설명
ListMap 객체가 정의되어 있을 때	ListMap	ListMap 객체
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var listMap = variable.getListMap();
```

□ setListMap

○ 설명

변수의 타입이 list일 때, 리스트 항목들을 저장한 ListMap 객체를 설정한다.

○ 인수

데이터 형	설명
ListMap	ListMap 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var listMap = new ListMap();
variable.setListMap(listMap);
```

□ getListItem

@See ListMap.getListItem()

□ addListItem

@See ListMap.addListItem()

□ getLocalizedListItemText

@See ListMap.getLocalizedListItemText()

□ getLocalizedListItemXML

@See ListMap.getLocalizedListItemXML()

□ getLocalizedListMap

@See ListMap.getLocalizedListMap()

□ getListXMLMap

@See ListMap.getListXMLMap()

□ removeListItem

@See ListMap.removeListItem()

□ getSweep

○ 설명

변수의 타입이 numeric일 때, Sweep 객체를 얻는다.

○ Return

조건	데이터 형	설명
Sweep이 정의되어 있을 때	Sweep	Sweep 객체
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var sweep = variable.getSweep();
```

□ setSweep

○ 설명

Sweep 객체를 설정한다.

○ 인수

데이터 형	설명
Sweep	Sweep 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var sweep = new Sweep();
//Do something with sweep.....
variable.setSweep(sweep);
```

□ isSweepable

○ 설명

변수가 Sweep 가능한 지 아닌 지를 조회한다

○ Return

조건	데이터 형	설명
Sweep 도메인이 정의되어 있을 때	boolean	true
그렇지 않으면	boolean	false

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var sweepable = variable.isSweepable();
```

□ getSweepDomainLowerLimit

@See Sweep.getDomainLowerLimit(), Domain.getLowerLimit()

□ setSweepDoaminLowerLimit

@See Sweep.setDomainLowerLimit(), Domain.setLowerLimit()

□ getSweepDomainUpperLimit

@See Sweep.getDomainUpperLimit(), Domain.getUpperLimit()

☐ setSweepDomainUpperLimit

@See Sweep.setDomainUpperLimit(), Domain.setUpperLimit()

☐ getSweepDomainOperand

@See Sweep.getDomainOperand(), Domain.getOperand()

☐ setSweepDomainOperand

@See Sweep.setDomainOperand(), Domain.setOperand()

☐ containSweepDomainLowerLimit

@See Sweep.containDomainLowerLimit(), Domain.containLowerLimit()

☐ containSweepDomainUpperLimit

@See Sweep.containDomainUpperLimit(), Domain.containUpperLimit()

☐ isSweepDomainLowerLimitContained

@See Sweep.isDomainLowerLimitContained(), Domain.isLowerLimitContained()

☐ isSweepDomainUpperLimitContained

@See Sweep.isDomainUpperLimitContained(), Domain.isUpperLimitContained()

☐ setSweepDomainValues

@See Sweep.setDomainValues(), Domain.setValues()

☐ isInSweepDomain

@See Sweep.isInDomain(), Domain.isInDomain()

☐ getOrder

○ 설명

Deprecated. 변수의 지정된 순서를 조회한다.

○ Return

조건	데이터 형	설명
순서가 정의되어 있을 때	number	변수의 순서
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var order = variable.getOrder();
```

☐ setOrder

○ 설명

Deprecated. 변수의 순서를 설정한다.

○ 인수

데이터 형	설명
Number	변수의 순서

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
variable.setOrder(1);
```

□ getDescriptionMap

○ 설명

변수에 대한 설명을 (언어 아이디 : 텍스트) 쌍의 맵으로 얻는다.

○ Return

조건	데이터 형	설명
데이터가 정의되어 있을 경우	LocalizedTextMap	데이터를 포함한 객체
그렇지 않으면	LocalizedTextmap	빈 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var map = variable,getDescriptionMap();
```

□ setDescriptionMap

○ 설명

Variable 객체에 LocalizedTextMap 객체를 설명 속성으로 설정한다.

○ 인수

데이터 형	설명
LocalizedTextMap	LocalizedTextMap 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var map = new LocalizedTextMap();
variable.setDescriptionMap(map);
```

□ getLocalizedDescription

@See LocalizedTextMap.getLocalizedText()

□ addLocalizedDescription

@See LocalizedTextMap.addLocalizedText()

□ removeLocalizedDescription

@See LocalizedTextMap.removeLocalizedText()

□ getLocalizedDescriptionXML

@See LocalizedTextMap.getLocalizedXML()

□ cleanDescription

@See LocalizedTextMap.clean()

□ getDimension

○ 설명

변수가 Vector 타입일 때 정의된 차원을 조회한다.

○ Return

조건	데이터 형	설명
차원이 정의되어 있을 때	number	변수의 차원
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
var dimension = variable.getDimension();
```

□ setDimension

○ 설명

변수의 타입이 Vector일 때 차원을 정의한다.

○ 인수

데이터 형	설명
Number	변수의 차원

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable(“var_1”);
variable.setDimension(3);
```

□ verifyName

○ 설명

변수 이름이 명명 규칙에 맞는 지를 검사한다. 변수의 이름은 반드시 알파벳 대소문자로 시작되어야 하며,

알파벳 대소문자, “_”와 숫자만 허용한다.

○ 인수

데이터 형	설명
String	변수 이름

○ Return

조건	데이터 형	설명
명명 규칙에 맞으면	boolean	true
그렇지 않으면	boolean	false

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var variable = new Variable();
var result = variable.verifyName(“var_1”);
```

□ setData

○ 설명

Variable 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var variable = new Variable();
var jsonData = getDataFromService();
var result = variable.setData(jsonData);
```

10. Domain

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable(“var_1”);
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel(“var_1”);
}
var domain = variable.getValueDomain();
//Do something with domain
```

□ JSON Format

```
{
    lower-limit: "value",
    upper-limit: "value",
    operand: "<>"
}
```

□ getLowerLimit

○ 설명

도메인의 하한값을 얻는다.

○ Return

조건	데이터 형	설명
하한값이 정의되어 있을 때	number	도메인의 하한 값
그렇지 않으면	String	“undefined”

□ setLowerLimit

○ 설명

도메인의 하한값을 설정한다.

○ 인수

데이터 형	설명
numeric	도메인의 하한값

○ Return

조건	데이터 형	설명
하한값이 상한값보다 크거나 같으면	boolean	false
그렇지 않으면	boolean	true

□ getUpperLimit

○ 설명

도메인의 상한값을 얻는다.

○ Return

조건	데이터 형	설명
상한값이 정의되어 있을 때	number	도메인의 상한 값
그렇지 않으면	String	“undefined”

□ setUpperLimit

○ 설명

도메인의 상한값을 설정한다.

○ 인수

데이터 형	설명
numeric	도메인의 상한값

○ Return

조건	데이터 형	설명
상한값이 하한값보다 작거나 같으면	boolean	false
그렇지 않으면	boolean	true

□ getOperand

○ 설명

도메인의 상하한 값 포함 여부를 얻는다.

○ Return

조건	데이터 형	설명
상한하한 값 포함 여부가 정의되어 있을 때	String	"==", "<=", "=>", "<>" 중 하나. • "==": min <= val <= max • "<=": min < val <= max • "=>": min <= val < max • "<>": min < val < max
그렇지 않으면	String	"undefined"

□ setOperand

○ 설명

도메인의 상하한값 포함 여부를 설정한다.

○ 인수

데이터 형	설명
String	도메인의 상하한 값 포함 여부

○ Return

조건	데이터 형	설명
Operand가 유효하고 저장에 성공하면	boolean	true
그렇지 않으면	boolean	false

□ containLowerLimit

○ 설명

도메인의 하한값을 포함하도록 설정한다.

○ Return

조건	데이터 형	설명
Operand가 정의되어 있고 저장에 성공하면	boolean	true
그렇지 않으면	boolean	false

□ containUpperLimit

○ 설명

도메인의 상한값을 포함하도록 설정한다.

○ Return

조건	데이터 형	설명
Operand가 정의되어 있고 저장에 성공하면	boolean	true
그렇지 않으면	boolean	false

□ isLowerLimitContained

○ 설명

도메인의 하한값이 포함되도록 설정 되었는 지 여부를 테스트한다.

○ Return

조건	데이터 형	설명
하한값이 포함되어 있으면	boolean	true
하한값이 포함되어 있지 않으면	boolean	false
하한값이 정의되어 있지 않으면	String	“undefined”

□ isUpperLimitContained

○ 설명

도메인의 상한값이 포함되도록 설정 되었는 지 여부를 테스트한다.

○ Return

조건	데이터 형	설명
상한값이 포함되어 있으면	boolean	true
상한값이 포함되어 있지 않으면	boolean	false
상한값이 정의되어 있지 않으면	String	“undefined”

□ setValues

○ 설명

도메인의 상하한 값과 상하한값 포함 여부를 저장한다.

○ 인수

데이터 형	설명
number	하한값
number	상한값
String	경계값 포함여부

○ Return

조건	데이터 형	설명
모든 인수가 검증되고 저장에 성공하면	boolean	true
그렇지 않으면	boolean	false

□ isInDomain

○ 설명

특정한 값이 도메인 내에 있는 지 조사한다.

○ 인수

데이터 형	설명
number	조사 대상 값

○ Return

조건	데이터 형	설명
특정 값이 도메인 내에 존재하면	boolean	true
그렇지 않으면	boolean	false

□ setData

○ 설명

Domain 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

11. LocalizedTextMap

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
var localizedTextMap = variable.getNameTextMap();
//Do something with localizedTextMap
```

□ JSON Format

```
{
   map:{
        {#language-id} : {#text},
        ......
   }
}
```

□ getMap

○ 설명

LocalizedTextMap 객체에서 (언어 아이디:텍스트) 프로퍼티를 관리하는 Object 객체. 직접적인 호출을 금한다.

○ Return

조건	데이터 형	설명
객체가 정의되어 있으면 저장된 객체를 반환하고, 정의되어 있지 않으면 객체를 생성하여 반환한다.	Object	Object 객체

□ setMap

○ 설명

LocalizedTextMap 객체에서 (언어 아이디:텍스트) 프로퍼티를 관리하는 Object 객체를 저장한다. 직접적인 호출을 금한다.

○ 인수

데이터 형	설명
Object	저장할 Object 객체

○ 사용 예제

□ getLocalizedText

○ 설명

특정한 언어로 된 텍스트를 얻는다.

○ 인수

데이터 형	설명
String	언어 아이디 문자열

○ Return

조건	데이터 형	설명
언어 아이디로 정의된 텍스트가 있으면	String	특정 언어로된 텍스트 문자열
그렇지 않으면	String	“undefined”

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var map = new LocalizedTextMap();
var text = map.getLocalizedText(“ko_KR”);
```

□ addLocalizedText

○ 설명

특정한 언어로 된 텍스트를 추가한다.

○ 인수

데이터 형	설명
String	언어 아이디 문자열
String	텍스트 문자열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var map = new LocalizedTextMap();
map.addLocalizedText(“ko_KR”, “한국어”);
```

□ removeLocalizedText

○ 설명

특정한 언어로 된 텍스트를 삭제한다.

○ 인수

데이터 형	설명
String	언어 아이디 문자열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var map = new LocalizedTextMap();
map.removeLocalizedText("ko_KR");
```

□ getLocalizedXML

○ 설명

Liferay의 서비스 빌더에서 localized 필드로 지정된 xml 형식 문자열을 생성한다.

○ 인수

데이터 형	설명
String[]	사용가능한 언어 아이디 문자열 배열
String	기본 언어 아이디

○ Return

조건	데이터 형	설명
	String	xml 형식의 문자열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
<%
   Locale[] locales = LanguageUtil.getAvailableLocales();

   JSONArray jsonLocales = JSONFactoryUtil.createJSONArray();
   for(Locale al : locales){
        jsonLocales.put(al.toString());
   }

   var defaultLocale =  themeDisplay.getSiteDefaultLocale();
%>

var map = new LocalizedTextMap();
var availableLanguageIds = <%= jsonLocales.toString() %>;
var defaultLanguageId = '<%= defaultLocale.toString() %>';

map.getLocalizedXML(lavailableLanguageIds, defaultLocale);
```

□ setData

○ 설명

LocalizedTextMap 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var map = new LocalizedTextMap();
var jsonData = getDataFromService();
var result = map.setData(jsonData);
```

12. ActivateCondition

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
var conditionContainer = variable.getActivateConditionContainer();
var conditions = conditionContainer.getContainer();
var condition = conditions[0];
//Do something with condition
```

□ JSON Format

```
{
    variable-name : {#variable-name},
    list-item-value : {#list-item-value},
    domain : {#Domain Object}
}
```

□ getVariableName

○ 설명

활성화 조건의 대상인 변수 이름을 얻는다.

○ Return

조건	데이터 형	설명
변수 이름이 정의 되어 있을 때	String	변수 명
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
```

```
var name = condition.getVariableName();
```

□ setVariableName

○ 설명

변수 이름을 설정한다.

○ 인수

데이터 형	설명
String	변수 이름

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
condition.setVariableName("var_1");
```

□ getListItemValue

○ 설명

활성화 조건의 리스트 아이템 값을 얻는다.

○ Return

조건	데이터 형	설명
리스트 아이템 값이 정의되어 있으면	String	리스트 아이템 값
그렇지 않으면	String	"undefined"

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
var value = condition.getListItemValue();
```

□ setListItemValue

○ 설명

활성화 조건의 리스트 아이템 값을 설정한다.

○ 인수

데이터 형	설명
String	아이템 값

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
condition.setListItemValue("itemValue");
```

□ getDomain

○ 설명

활성화 조건의 수치 조건을 얻는다.

○ Return

조건	데이터 형	설명
수치 도메인이 정의되어 있으면	Domain	값이 설정된 수치 도메인 객체
그렇지 않으면	Domain	빈 도메인 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
var domain = condition.getDomain();
```

☐ setDomain

○ 설명

활성화 조건의 수치 도메인을 설정한다.

○ 인수

데이터 형	설명
Domain	도메인 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
var domain = new Domain();
condition.setDomain(domain);
```

☐ getDomainLowerLimit

@See Domain.getLowerLimit

☐ setDomainLowerLimit

@See Domain.setLowerLimit

☐ getDomainUpperLimit

@See Domain.getUpperLimit

☐ setDomainUpperLimit

@See Domain.setUpperLimit

☐ getDomainOperand

@See Domain.getOperand

☐ setDomainOperand

@See Domain.setOperand

☐ containDomainLowerLimit

@See Domain.containLowerLimit

☐ containDomainUpperLimit

@See Domain.containUpperLimit

☐ isDomainLowerLimitContained

@See Domain.isLowerLimitContained

□ isDomainUpperLimitContained

@See Domain.isUpperLimitContained

□ isInDomain

@See Domain.isInDomain

□ setNumericCondition

○ 설명

활성화 조건의 변수 명과 수치 도메인의 값을 설정한다.

○ 인수

데이터 형	설명
String	변수 명
numeric	하한값
numeric	상한값
String	상하한 경계값 포함 여부

○ Return

조건	데이터 형	설명
상하한값과 operand가 검증되면	boolean	true
그렇지 않으면	boolean	false

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
var result = condition.setNumericCondition("var_1", 0.0, 1.0, "<>");
```

□ setListItemCondition

○ 설명

활성화 조건의 변수 명과 리스트 아이템 값을 설정한다.

○ 인수

데이터 형	설명
String	변수 명
String	리스트 아이템 값

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
condition.setListItemCondition("var_1", "itemValue");
```

□ setData

○ 설명

ActivateCondition 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var condition = new ActivateCondition();
var jsonData = getDataFromService();
var result = condition.setData(jsonData);
```

13. ActivateConditionContainer

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
var conditionContainer = variable.getActivateConditionContainer();

//Do something with conditionContainer
```

□ JSON Format

```
{
    container : [
        {#ActivateCondition Object}, ......
    ]
}
```

□ getContainer

○ 설명

ActivateCondition 객체들을 저장하는 배열을 얻는다. 만약 배열이 존재하지 않으면 새로운 배열을 생성하여 저장한 뒤 반환한다. 특별한 상황이 아니면 직접적인 사용을 권장하지 않는다.

○ Return

데이터 형	설명
Array	ActivateCondition 객체를 저장하는 배열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var activateConditionContainer = new ActivateConditionContainer();
var container = activateConditionContainer.getContainer();
```

□ setContainer

○ 설명

ActivateCondition 객체들을 저장하는 배열을 저장한다. 특별한 상황이 아니면 직접적인 사용을 권장하지 않는다.

○ 인수

데이터 형	설명
Array	배열

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var activateConditionContainer = new ActivateConditionContainer();
var container = [];

activateConditionContainer.setContainer(container);
```

□ addActivateCondition

○ 설명

ActivateCondition 객체를 컨테이너에 추가한다.

○ 인수

데이터 형	설명
ActivateCondition	ActivateCondition 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var conditionContainer = variable.getActivateConditionContainer();
var condition = new ActivateCondition();
//Do something with condition.....
conditionContainer.addActivateCondition(condition);
```

□ addListItemActivateCondition

○ 설명

리스트 항목을 활성화 조건에 추가한다.

○ 인수

데이터 형	설명
String	변수 이름
String	리스트 항목 값

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var conditionContainer = variable.getActivateConditionContainer();
```

```
conditionContainer.addListItemActivateCondition("var_1", "itemValue");
```

□ addNumericActivateCondition

○ 설명

Numeric 타입의 변수를 활성화 조건으로 추가한다.

○ 인수

데이터 형	설명
String	변수 이름
Numeric	활성화 조건의 하한 값
Numeric	활성화 조건의 상한 값
String	활성화 조건의 상하한 값 포함 여부

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var conditionContainer = variable.getActivateConditionContainer();
conditionContainer.addNumericActivateCondition("var_1", 0, 1, "<>");
```

□ removeActivateConditions

○ 설명

특정한 변수의 활성화 조건들을 삭제한다.

○ 인수

데이터 형	설명
String	변수 이름
String	리스트 항목 값

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var conditionContainer = variable.getActivateConditionContainer();
conditionContainer.removeActivateConditions("var_1");
```

□ removeListItemActivateCondition

○ 설명

특정한 리스트 항목의 활성화 조건을 삭제한다.

○ 인수

데이터 형	설명
String	변수 이름

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var conditionContainer = variable.getActivateConditionContainer();
conditionContainer.removeListItemActivateConditions("var_1", "itemValue");
```

□ removeNumericActivateCondition

○ 설명

특정한 Numeric 타입 변수의 활성화 조건을 삭제한다.

○ 인수

데이터 형	설명
String	변수 이름
Numeric	활성화 조건의 하한 값
Numeric	활성화 조건의 상한 값
String	활성화 조건의 상하한 값 포함 여부

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var conditionContainer = variable.getActivateConditionContainer();
conditionContainer.removeNumericActivateConditions("var_1", 0, 1, "<>");
```

□ setData

○ 설명

ActivateConditionContainer 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var container = new ActivateConditionContainer();
var jsonData = getDataFromService();
var result = container.setData(jsonData);
```

14. ListItem

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
var listMap = variable.getListMap();
var listItem = listMap.getListItem("itemValue");
//Do something with listItem
```

□ JSON Format

```
{
    {
        value : {#list-item-value},
        localized-text-map : {#LocalizedTextMap Object},
        activate-condition-container:{#ActivateConditionContainer Object}
}
```

□ getLocalizedTextMap

○ 설명

리스트 아이템에 대한 다중언어 텍스트를 (언어 아이디 : 텍스트) 쌍의 맵으로 얻는다.

○ Return

조건	데이터 형	설명
데이터가 정의되어 있을 경우	LocalizedTextMap	데이터를 포함한 객체
그렇지 않으면	LocalizedTextmap	빈 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var listMap = variable,getListMap();
var listItem = listMap.getListItem("itemValue");
var textMap = listItem.getLocalizedTextMap();
```

□ setNameTextMap

○ 설명

ListItem 객체에 LocalizedTextMap 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
LocalizedTextMap	데이터를 저장하고 있는 LocalizedTextMap 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = InputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED )
   variable = InputdeckForm.createVariable("var_1");
var listMap = variable,getListMap();
var listItem = listMap.getListItem("itemValue");
var textMap = new LocalizedTextMap();
listItem.setLocalizedTextMap(textMap);
```

☐ getLocalizedText

@See LocalizedTextMap.getLocalizedText()

☐ addLocalizedText

@See LocalizedTextMap.addLocalizedText()

☐ removeLocalizedText

@See LocalizedTextMap.removeLocalizedText()

☐ getLocalizedXML

@See LocalizedTextMap.getLocalizedXML()

☐ cleanLocalizedTextMap

@See LocalizedTextMap.clean()

☐ getItemValue

○ 설명

리스트 아이템 값을 조회한다.

○ Return

조건	데이터 형	설명
아이템 값이 정의되어 있을 때	String	리스트 아이템 값
그렇지 않으면	String	"undefined"

☐ setItemValue

○ 설명

리스트 아이템 값을 설정한다.

○ 인수

데이터 형	설명
String	리스트 아이템 값

☐ getActivateConditionContainer

○ 설명

리스트 아이템의 활성화 조건인 ActivateCondition 객체들을 저장한 ActivateConditionContainer 객체를 얻는다.

○ Return

데이터 형	설명
ActivateConditionContainer	ActivateConditionContainer 객체가 형성되어 있으면 그 객체를 반환하고, 그렇지 않으면 객체를 생성하여 저장한 뒤 반환한다.

☐ setActivateConditionContainer

○ 설명

리스트 아이템의 활성화 조건인 ActivateCondition 객체들을 저장한 ActivateConditionContainer 객체를 설정한다.

○ 인수

데이터 형	설명
ActivateConditionContainer	ActivateConditionContainer 객체

☐ addActivateCondition

@See ActivateConditionContainer.addActivateCondition()

☐ addListItemActivateCondition

@See ActivateConditionContainer.addListItemActivateCondition

☐ addNumericActivateCondition

@See ActivateConditionContainer.addNumericActivateCondition

☐ addGroupActivateCondition

@See ActivateConditionContainer.addGroupActivateCondition

☐ removeActivateConditions

@See ActivateConditionContainer.removeActivateCondition

☐ removeListItemActivateCondition

@See ActivateConditionContainer.removeListItemActivateCondition

☐ removeNumericActivateCondition

@See ActivateConditionContainer.removeNumericActivateCondition

☐ setData

○ 설명

ListItem 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

15. ListMap

□ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
var listMap = variable.getListMap();
//Do something with listMap
```

□ JSON Format

```
{
    map : {#ListItem Object},
    ......
}
```

□ getListMap

○ 설명

ListItem 객체들을 (리스트 아이템 값, ListItem 객체)의 쌍으로 수용하고 있는 객체를 얻는다. 직접적인 호출을 금한다.

○ Return

데이터 형	설명
Object	ListItem 객체를 수용하는 Object 객체

□ setListMap

○ 설명

ListItem 객체들을 (리스트 아이템 값, ListItem 객체)의 쌍으로 수용하는 객체를 설정한다. 직접적인 호출을 금한다.

○ 인수

데이터 형	설명
Object	ListItem 객체를 수용하는 Object 객체

□ getLocalizedListItemText

○ 설명

특정한 리스트 아이템에 대하여 특정한 언어의 텍스트를 얻는다.

○ 인수

데이터 형	설명
String	리스트 아이템 값
String	언어 아이디

○ Return

@See ListItem.getLocalizedText()

□ getLocalizedListItemXML

○ 설명

특정한 리스트 아이템에 대하여 Liferay의 서비스 빌더에서 localized 필드로 지정된 xml 형식 문자열을 생성한다.

○ 인수

데이터 형	설명
String	리스트 아이템 값
String[]	사용가능한 언어 아이디 문자열 배열
String	기본 언어 아이디

○ Return

@See ListItem.getLocalizedText()

□ getLocalizedListMap

○ 설명

특정한 언어에 대하여 (리스트 아이템 값, 텍스트) 쌍으로 된 Object 객체를 얻어온다.

○ 인수

데이터 형	설명
String	언어 아이디

○ Return

조건	데이터 형	설명
지역화 텍스트가 있을 때	Object	(리스트 아이템 값, 지역화 텍스트) 쌍으로 모든 리스트 아이템을 객체에 저장하여 리턴
그렇지 않으면	Object	빈 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
var listMap = variable.getListMap();
var textMap = listMap.getLocalizedListMap("ko_KR");
```

□ getListXMLMap

○ 설명

저장된 모든 리스트 아이템에 대하여 (리스트 아이템 값, Liferay 서비스 빌더의 다국어 지원 XML 문자열) 형식의 Object 객체를 얻는다.

○ 인수

데이터 형	설명
String[]	사용가능한 언어 아이디 문자열 배열
String	기본 언어 아이디

○ Return

조건	데이터 형	설명
지역화 텍스트가 있을 때	Object	(리스트 아이템 값, XML) 쌍으로 모든 리스트 아이템을 객체에 저장하여 리턴
그렇지 않으면	Object	빈 객체

○ 사용 예제

```
<script src='path-to-js'/inputdeck_form.js></script>
<script src='path-to-js'/inputdeck_form.js></script>
<%
   Locale[] locales = LanguageUtil.getAvailableLocales();

   JSONArray jsonLocales = JSONFactoryUtil.createJSONArray();
   for(Locale al : locales){
         jsonLocales.put(al.toString());
   }

   String defaultLanguageId = themeDisplay.getLanguageId();
%>
var inputdeckForm = new InputdeckForm();
var variable = inputdeckForm.getVariable("var_1");
if( typeof variable == InputdeckConstants.UNDEFINED ){
   variable = inputdeckForm.createVariabel("var_1");
}
var listMap = variable.getListMap();

var availableLanguageIds = <%= jsonLocales.toString() %>;
var defaultLnaguageId = "<%= defaultLanguageId %>";
var xmlMap = listMap.getListXMLMap(availableLanguageIds, defaultLnaguageId);
```

□ removeListItem

○ 설명

특정한 값을 가진 리스트 아이템을 삭제한다.

○ 인수

데이터 형	설명
String	삭제할 리스트 아이템 값

□ setData

○ 설명

ListMap 객체에 JSON 객체를 데이터로 설정한다.

○ 인수

데이터 형	설명
Object	데이터를 저장하고 있는 JSON 객체

○ Return

조건	데이터 형	설명
오류가 발생했을 경우	String	오류 메세지
그렇지 않으면	boolean	true

16. InputdeckConstants

```
var InputdeckConstants = {
    VARIABLE_TYPE_STRING : 'string',
    VARIABLE_TYPE_NUMERIC : 'numeric',
    VARIABLE_TYPE_GROUP : 'group',
    VARIABLE_TYPE_COMMENT : 'comment',
    VARIABLE_TYPE_LIST : 'list',
    VARIABLE_TYPE_VECTOR: 'vector',
    VARIABLE_TYPE_FILE: 'file', //deprecated
    BRACE_TYPE_SQUARE:"SQUARE",
    BRACE_TYPE_SQUARE_SPACE:"SQUARE_SPACE",
    BRACE_TYPE_ROUND:"ROUND",
    BRACE_TYPE_ROUND_SPACE:"ROUND_SPACE",
    LOWER_LIMIT : 'lower-limit',
    UPPER_LIMIT : 'upper-limit',
    UNDEFINED : 'undefined',
    OPERAND : 'operand',
    MAP : 'map',
    VARIABLE_NAME: 'variable-name',
    LIST_ITEM_VALUE : 'list-item-value',
    DOMAIN : 'domain',
    CONTAINER : 'container',
    LOCALIZED_TEXT_MAP : 'localized-text-map',
    VALUE : 'value',
    ACTIVATE_CONDITION_CONTAINER : 'activate-condition-container',
    NAME : 'name',
    NAME_TEXT_MAP : 'name-text-map',
    TYPE : 'type',
    VALUE_DOMAIN : 'value-domain',
    LIST_MAP : 'list-map',
    SWEEP_DOMAIN : 'sweep-domain',
    ORDER : 'order',
    DESCRIPTION_MAP : 'description-map',
    DIMENSION : 'dimension',
    BRACE_CHAR : 'brace-char',
    DELIMITER : 'delimiter',
    SAMPLE : 'sample',
    SPACE : 'space',
    VALUE_DELIMITER : 'value-delimiter',
    LINE_DELIMITER : 'line-delimiter',
    COMMENT_CHAR : 'comment-char',
    VECTOR_FORM : 'vector-form',
    LINE_FORM : 'line-form',
    AVAILABLE_LANGUAGE_IDS : 'available-language-ids',
```

```
    DEFAULT_LANGUAGE_ID : 'default-language-id',
    VARIABLE_MAP : 'variable-map',
    UNIT : 'unit',
    SWEEP : 'sweep',
    SWEEPABLE : 'sweepable',
    SWEEP_SLICE : 'sweep-slice',
    TARGET_LANGUAGE_ID : 'target-language-id',
    getBraceTypes : function (){
        var types = [];
        types.push(this.BRACE_TYPE_SQUARE);
        types.push(this.BRACE_TYPE_SQUARE_SPACE);
        types.push(this.BRACE_TYPE_ROUND);
        types.push(this.BRACE_TYPE_ROUND_SPACE);
    },
    getVariableTypes : function (){
        var types = [];
        types.push(this.VARIABLE_TYPE_NUMERIC);
        types.push(this.VARIABLE_TYPE_VECTOR);
        types.push(this.VARIABLE_TYPE_STRING);
        types.push(this.VARIABLE_TYPE_GROUP);
        types.push(this.VARIABLE_TYPE_COMMENT);
    }
};
```

17. InputdeckErrors

```
var InputdeckErrors = {
    LOWER_LIMIT_INVALID : "lower limit invalid",
    UPPER_LIMIT_INVALID : "upper limit invalid",
    OPERAND_INVALID : "operand invalid",
};
```

18. PortConstants

```
var PortConstants = {
    NAME : 'name',
    MANDATORY : 'mandatory',
    PORT_TYPE_ID : 'port-type-id',
    DEFAULT_EDITOR_ID : 'default-editor-id',
    DEFAULT_ANALYZER_ID : 'default-analyzer-id',
    ORDER : 'order',
    FILE_NAME : 'file-name',
    CONTAINER : 'container',
    UNDEFINED : 'undefined'
};
```

19. PortErrors

```
var PortErrors = {
    INVALID_PORT_NAME : 'invalid-port-name',
    DUPLICATED_PORT_NAME : 'duplicated-port-name',
```

```
};
```