SMS Service

Summary

The SMS Service provides the interface for using the M-Gov's SMS service, and you can access it by request.

This service layer provides easier access and usability to the **SMS feature** that utilizes the existing eGovFrame common components, by excluding the framework itself and the other common modules.

In case of using the eGovFrame, you can simply use the existing SMS messaging service.

Prerequisites

In order to use the SMS Service, you ("the requester" from here) need to submit a request to the National Computing and Information Agency under the Ministry of Public Administration and Security ("the Agency" from here) for approval.

Following is the overall procedure:

① Request for usage: depending on the service, fill out the Annex No.1 or 3 form (refer to the website's official guideline) and submit it to the Agency.
② Approval: within 15 days the review and approval of the request will be sent to the requester.
③ Development: The requester can use the M-Gov API provided by the Agency to connect and implement.
④ Going live: if the requester does not go live with the service within 60 days of approval notice, the Agency reserves the right to withdraw permission and access to use the service.

For further details and questions, check out the M-gov website (http://www.mgov.go.kr).

Description

The SMS service provides access to the SMS feature via the M-Gov connection API, as well as SMS history management feature. (There exists a separate method for the scenario which does not require history management, but this lacks the ability to confirm send results.)

Package Dependency

The package has direct functional dependency to the common package (cmm) of element technology and authorization management package. However, for executing without errors in component distribution, the distribution package should include department authorization management, authorization group management, group management, spring security, user integration certification, element technology (utility) system, format/date/calculation, web-editor, and mail connection interface pacakge.

- Package dependency: Collaboration bulletin, communities, clubs Package Dependency

Related Source

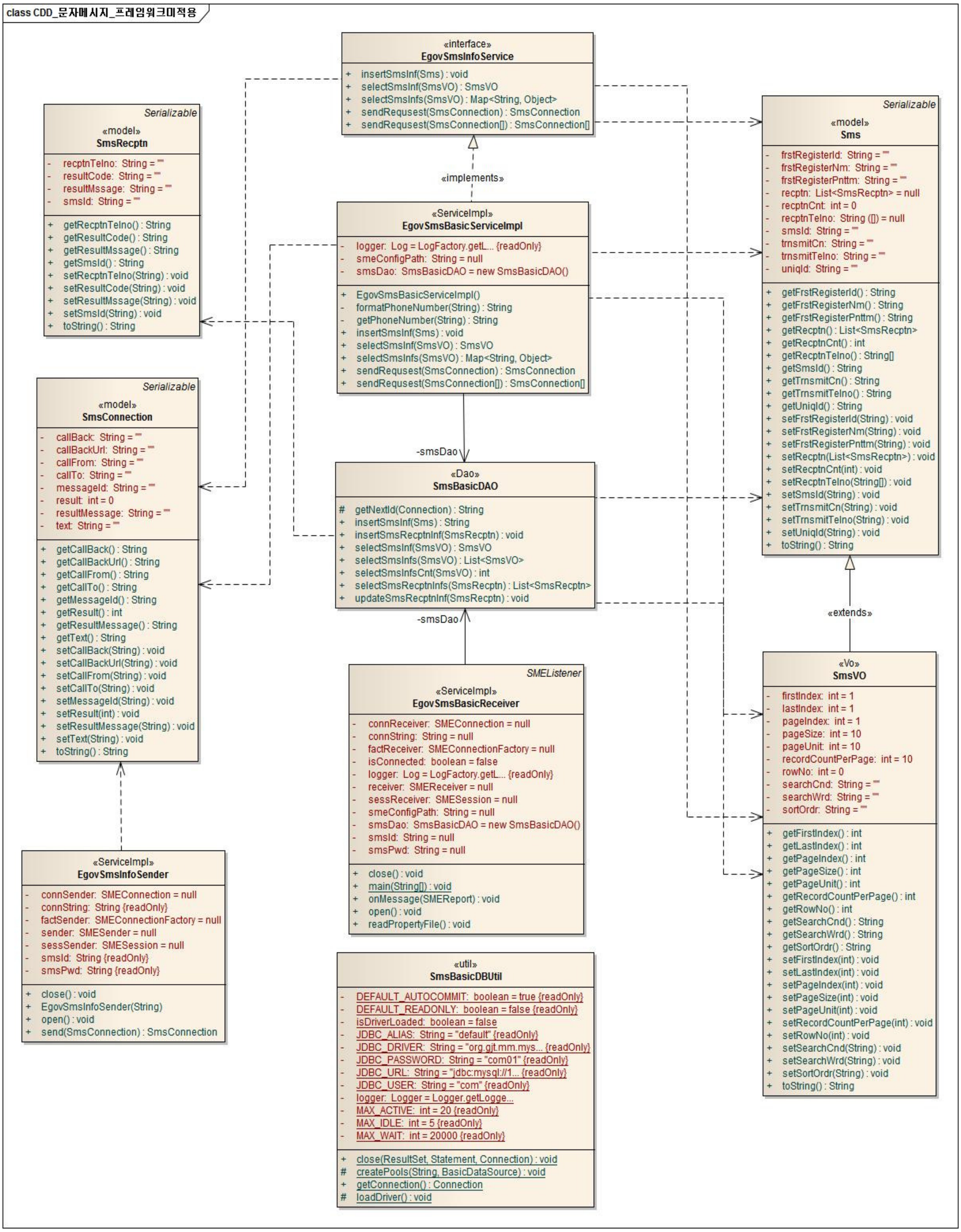
Type	Source	Remarks	Common?
Service	egovframework.com.cop.sms.service.EgovSmsInfoService.java	Service interface for the SMS Service	Y
ServiceImpl	egovframework.com.cop.sms.service.impl.EgovSmsBasicServiceImpl.java	Service implementation class for SMS service	
ServiceImpl	egovframework.com.cop.sms.service.impl.EgovSmsInfoSender.java	Class for handling SMS connections	Y
ServiceImpl	egovframework.com.cop.sms.service.impl.EgovSmsBasicReceiver.java	Class for handling SMS connection results	
Model	egovframework.com.cop.sms.service.Sms.java	Model class for SMS Service	Y
Model	egovframework.com.cop.sms.service.SmsRecptn.java	Model class for SMS Service (message reception info)	Y

VO	egovframework.com.cop.sms.service.SmsVO.java	VO class for SMS Service	Y
VO	egovframework.com.cop.sms.service.SmsConnection.java	Model class for SMS Service (connection info)	Y
DAO	egovframework.com.cop.sms.service.impl.SmsBasicDAO.java	Data processing class for SMS Service	
Util	egovframework.com.cop.sms.service.impl.SmsBasicDBUtil.java	Data processing utility (Pool creation etc.)	

※ Common column indicates whether the class is a shared one with the eGovFrame's SMS service.

Related Classes Diagrams

Class Diagram


class CDD_문자메시지_프레임워크미적용
«interface»
EgovSmsInfoService
+ insertSmsInf(Sms) : void
+ selectSmsInf(SmsVO) : SmsVO
+ selectSmsInfs(SmsVO) : Map<String, Object>
+ sendRequsest(SmsConnection) : SmsConnection
+ sendRequsest(SmsConnection[]) : SmsConnection[]
«implements»
«ServiceImpl»
EgovSmsBasicServiceImpl
- logger: Log = LogFactory.getL... {readOnly}
- smeConfigPath: String = null
- smsDao: SmsBasicDAO = new SmsBasicDAO()
+ EgovSmsBasicServiceImpl()
- formatPhoneNumber(String) : String
- getPhoneNumber(String) : String
+ insertSmsInf(Sms) : void
+ selectSmsInf(SmsVO) : SmsVO
+ selectSmsInfs(SmsVO) : Map<String, Object>
+ sendRequsest(SmsConnection) : SmsConnection
+ sendRequsest(SmsConnection[]) : SmsConnection[]
-smsDao
«Dao»
SmsBasicDAO
# getNextId(Connection) : String
+ insertSmsInf(Sms) : String
+ insertSmsRecptnInf(SmsRecptn) : void
+ selectSmsInf(SmsVO) : SmsVO
+ selectSmsInfs(SmsVO) : List<SmsVO>
+ selectSmsInfsCnt(SmsVO) : int
+ selectSmsRecptnInfs(SmsRecptn) : List<SmsRecptn>
+ updateSmsRecptnInf(SmsRecptn) : void
-smsDao
SMEListener
«ServiceImpl»
EgovSmsBasicReceiver
- connReceiver: SMEConnection = null
- connString: String = null
- factReceiver: SMEConnectionFactory = null
- isConnected: boolean = false
- logger: Log = LogFactory.getL... {readOnly}
- receiver: SMEReceiver = null
- sessReceiver: SMESession = null
- smeConfigPath: String = null
- smsDao: SmsBasicDAO = new SmsBasicDAO()
- smsId: String = null
- smsPwd: String = null
+ close() : void
+ main(String[]) : void
+ onMessage(SMEReport) : void
+ open() : void
+ readPropertyFile() : void
«util»
SmsBasicDBUtil
- DEFAULT_AUTOCOMMIT: boolean = true {readOnly}
- DEFAULT_READONLY: boolean = false {readOnly}
- isDriverLoaded: boolean = false
- JDBC_ALIAS: String = "default" {readOnly}
- JDBC_DRIVER: String = "org.gjt.mm.mys... {readOnly}
- JDBC_PASSWORD: String = "com01" {readOnly}
- JDBC_URL: String = "jdbc:mysql://1... {readOnly}
- JDBC_USER: String = "com" {readOnly}
- logger: Logger = Logger.getLogge...
- MAX_ACTIVE: int = 20 {readOnly}
- MAX_IDLE: int = 5 {readOnly}
- MAX_WAIT: int = 20000 {readOnly}
+ close(ResultSet, Statement, Connection) : void
# createPools(String, BasicDataSource) : void
+ getConnection() : Connection
# loadDriver() : void
Serializable
«model»
SmsRecptn
- recptnTelno: String = ""
- resultCode: String = ""
- resultMssage: String = ""
- smsId: String = ""
+ getRecptnTelno() : String
+ getResultCode() : String
+ getResultMssage() : String
+ getSmsId() : String
+ setRecptnTelno(String) : void
+ setResultCode(String) : void
+ setResultMssage(String) : void
+ setSmsId(String) : void
+ toString() : String
Serializable
«model»
SmsConnection
- callBack: String = ""
- callBackUrl: String = ""
- callFrom: String = ""
- callTo: String = ""
- messageId: String = ""
- result: int = 0
- resultMessage: String = ""
- text: String = ""
+ getCallBack() : String
+ getCallBackUrl() : String
+ getCallFrom() : String
+ getCallTo() : String
+ getMessageId() : String
+ getResult() : int
+ getResultMessage() : String
+ getText() : String
+ setCallBack(String) : void
+ setCallBackUrl(String) : void
+ setCallFrom(String) : void
+ setCallTo(String) : void
+ setMessageId(String) : void
+ setResult(int) : void
+ setResultMessage(String) : void
+ setText(String) : void
+ toString() : String
«ServiceImpl»
EgovSmsInfoSender
- connSender: SMEConnection = null
- connString: String {readOnly}
- factSender: SMEConnectionFactory = null
- sender: SMESender = null
- sessSender: SMESession = null
- smsId: String {readOnly}
- smsPwd: String {readOnly}
+ close() : void
+ EgovSmsInfoSender(String)
+ open() : void
+ send(SmsConnection) : SmsConnection
Serializable
«model»
Sms
- frstRegisterId: String = ""
- frstRegisterNm: String = ""
- frstRegisterPnttm: String = ""
- recptn: List<SmsRecptn> = null
- recptnCnt: int = 0
- recptnTelno: String ([]) = null
- smsId: String = ""
- trnsmitCn: String = ""
- trnsmitTelno: String = ""
- uniqId: String = ""
+ getFrstRegisterId() : String
+ getFrstRegisterNm() : String
+ getFrstRegisterPnttm() : String
+ getRecptn() : List<SmsRecptn>
+ getRecptnCnt() : int
+ getRecptnTelno() : String[]
+ getSmsId() : String
+ getTrnsmitCn() : String
+ getTrnsmitTelno() : String
+ getUniqId() : String
+ setFrstRegisterId(String) : void
+ setFrstRegisterNm(String) : void
+ setFrstRegisterPnttm(String) : void
+ setRecptn(List<SmsRecptn>) : void
+ setRecptnCnt(int) : void
+ setRecptnTelno(String[]) : void
+ setSmsId(String) : void
+ setTrnsmitCn(String) : void
+ setTrnsmitTelno(String) : void
+ setUniqId(String) : void
+ toString() : String
«extends»
«Vo»
SmsVO
- firstIndex: int = 1
- lastIndex: int = 1
- pageIndex: int = 1
- pageSize: int = 10
- pageUnit: int = 10
- recordCountPerPage: int = 10
- rowNo: int = 0
- searchCnd: String = ""
- searchWrd: String = ""
- sortOrdr: String = ""
+ getFirstIndex() : int
+ getLastIndex() : int
+ getPageIndex() : int
+ getPageSize() : int
+ getPageUnit() : int
+ getRecordCountPerPage() : int
+ getRowNo() : int
+ getSearchCnd() : String
+ getSearchWrd() : String
+ getSortOrdr() : String
+ setFirstIndex(int) : void
+ setLastIndex(int) : void
+ setPageIndex(int) : void
+ setPageSize(int) : void
+ setPageUnit(int) : void
+ setRecordCountPerPage(int) : void
+ setRowNo(int) : void
+ setSearchCnd(String) : void
+ setSearchWrd(String) : void
+ setSortOrdr(String) : void
+ toString() : String

Tables

Table	Name (English)	Remarks
SMS	COMTNSMS	Manages SMS sending
SMS Reception	COMTNSMSRECPTN	Manages SMS receiving

Configuration

- SMEConfig.properties file

In order to use the Service, the SMEConfig.properties file provided by the M-Gov connection API needs to be specified.

To do so, check out the **egovframework.com.cop.sms.service.impl.EgovSmsBasicServiceImpl** - specifically the "public EgovSmsBasicServiceImpl()" constructor.

```
public class EgovSmsBasicServiceImpl implements EgovSmsInfoService {
...
  public EgovSmsBasicServiceImpl() {
          //-------------------------------
          // Get attributes.
          // Use the file path of the M-Gov-provided SMEConfig.conf file.
          //-------------------------------
          //smeConfigPath = EgovProperties.getProperty("Globals.SMEConfigPath");

      // ...

          String globalsPropertiesFile = System.getProperty("user.home")
                      + System.getProperty("file.separator") + "egovProps"
                      + System.getProperty("file.separator") + "conf"
                      + System.getProperty("file.separator") + "SMEConfig.properties";

          smeConfigPath = globalsPropertiesFile;

  }
...
```

For the globalsPropertiesFile variable, feed in the path for the SMEConfig.properties file.

- Tweaking how the SMEConfig.properties is handled

A less elegant but more simple option is to hard-code the path as a String value for the smeConfigPath variable. This will require modifying the source code again in case the file location changes later on.

(If project common module properties can be handled altogether via other means, it is recommended to choose such method over what is just mentioned above.)

```
public class EgovSmsBasicServiceImpl implements EgovSmsInfoService {
...
  public EgovSmsBasicServiceImpl() {
          //-------------------------------
          // Get attribute.
          // Use the file path of the M-Gov-provided SMEConfig.conf file.
          //-------------------------------
          smeConfigPath = "/product/jeus/egovProps/conf/SMEConfig.properties";
  }
...
```

※ This method just requires setting the smeConfigPath value with the file path.

For your information, the SMEConfig.properties file contains the M-Gov center information and account information, and is provided by the M-Gov.

Changing DBMS Information

To avoid framework dependency, there is a method provided to handle DBMS on its own. Currently the **egovframework.com.cop.sms.impl.SmsBasicDBUtil** handles creating connection pool via DBCP (http://commons.apache.org/dbcp/).

Following is an example of how you can set the DBMS information.

(Also, if there are common modules related to DB Connection, switch to the module as well.)

```
public class SmsBasicDBUtil {
...
   /** Connection Pool Alias */
   private static final String JDBC_ALIAS = "default";
   /** JDBC Driver name */
```

```
    private static final String JDBC_DRIVER = "org.gjt.mm.mysql.Driver";
    /** JDBC connection URL */
    private static final String JDBC_URL = "jdbc:mysql://192.168.200.24:1621/com";
    /** JDBC login user ID */
    private static final String JDBC_USER = "user";
    /** JDBC login password */
    private static final String JDBC_PASSWORD = "password";
    /** Maximum number of active connections a pool can maintain */
    private static final int MAX_ACTIVE = 20;
    /** Maximum number of idle connections a pool can maintain */
    private static final int MAX_IDLE = 5;
    /** Connection timeout */
    private static final int MAX_WAIT = 20000;
    /** Auto commit */
    private static final boolean DEFAULT_AUTOCOMMIT = true;
    /** Read-only */
    private static final boolean DEFAULT_READONLY = false;

...
```

Additionally, **egovframework.com.cop.sms.service.impl.SmsBasicDAO** has a query predefined for DBMS, and it uses MySQL format by default.

For Oracle databases, you can uncomment the commented “for Oracle” section and use it. (Altibase, Cubrid, Tibero queries also use the same format as Oracle’s.)

```
    public String insertSmsInf(Sms sms) throws Exception {
            String smsId = null;

            //variables
        Connection conn = null;
            PreparedStatement pstmt = null;

            StringBuffer buffer = new StringBuffer();

            try {
                // for mySql
                buffer.append("INSERT INTO COMTNSMS\n");
                buffer.append("  (SMS_ID, TRNSMIT_TELNO, TRNSMIT_CN,\n");
                buffer.append("   FRST_REGISTER_ID, FRST_REGISTER_PNTTM )\n");
                buffer.append("VALUES\n");
                buffer.append("(?, ?, ?, ?, SYSDATE())");


                // for Oracle
                /*
                buffer.append("INSERT INTO COMTNSMS\n");
                buffer.append("  (SMS_ID, TRNSMIT_TELNO, TRNSMIT_CN,\n");
                buffer.append("   FRST_REGISTER_ID, FRST_REGISTER_PNTTM )\n");
                buffer.append("VALUES\n");
                buffer.append("(?, ?, ?, ?, SYSDATE)");
                */

                conn = SmsBasicDBUtil.getConnection();

                conn.setAutoCommit(false);

                smsId = getNextId(conn);          // Generate SMS_ID...

                pstmt = conn.prepareStatement(buffer.toString());

                int index = 0;
                pstmt.setString(++index, smsId);
                pstmt.setString(++index, sms.getTrnsmitTelno());
                pstmt.setString(++index, sms.getTrnsmitCn());
                pstmt.setString(++index, sms.getFrstRegisterId());

                pstmt.executeUpdate();

                conn.commit();

                return smsId;
            } catch (Exception ex) {
                if (conn != null) {
                        conn.rollback();
                }
                throw ex;

            } finally {
                SmsBasicDBUtil.close(null, pstmt, conn);
            }
    }
```

SMS Receiver Daemon

Results of sent SMS is handled by a separate program. (M-Gov-provided API handles them asynchronously.)

You can enable it by running the **egovframework.com.cop.sms.service.impl.EgovSmsBasicReceiver** as below.

```
java [JVM Options] egovframework.com.cop.sms.service.impl.EgovSmsBasicReceiver
/home/egovframe/conf/SMEConfig.conf
```

JVM Options need classpath to be set, and the absolute path of the SMEConfig.properties (or SMEConfig.conf) file needs to be used as the last parameter.

In Unix environment, add the & at the end to run it in the background.

Additional Features

SMS Service comprises of SMS Sending, SMS History, and SMS Details.

SMS Sending

Business Rule

SMS Sending calls the following interface methods.

Related Codes

N/A

Screen and execution manual

(The Implement object uses EgovSmsBasicServiceImpl)

Interface	Return	Method	Parameters	Remarks
EgovSmsInfoService	void	insertSmsInf	Sms	

- Usage

```
EgovSmsInfoService service = new EgovSmsBasicServiceImpl();
...
Sms sms = new Sms();

sms.setTrnsmitTelno("010-1234-5678");
sms.setTrnsmitCn("CONTENTS");
sms.setFrstRegisterId("USRCNFRM_00000000001");
sms.setRecptnTelno(
  new String[] {
    "010-1234-1234",
    "010-1234-1235"
  }
);

service.insertSmsInf(sms);
...
```

recptnTelno is a String[] type that lets the user send SMS to multiple recipients.

SMS History

Business Rule

SMS History calls the following interface methods.

Related Codes

N/A

Screen and execution manual

(The Implement object uses EgovSmsBasicServiceImpl)

Interface	Return	Method	Parameters	Remarks
EgovSmsInfoService	Map<String, Object>	selectSmsInfs	SmsVO	

- Usage

```
EgovSmsInfoService service = new EgovSmsBasicServiceImpl();
...
searchVO.setUniqId("USRCNFRM_00000000001");

searchVO.setRecordCountPerPage(10);        // records per page
searchVO.setFirstIndex(0);            // set the row for beginning of page

// Calculation based on current page number
// currentPageNo = current page number (one-based)
// recordCountPerPage = records per page
// firstRecordIndex = (currentPageNo - 1) * recordCountPerPage

Map<String, Object> map = service.selectSmsInfs(searchVO);

System.out.println("List action is successed...");
System.out.println("Total result list count: " + (String)map.get("resultCnt"));
System.out.println("List count: " +  ((List<SmsVO>)map.get("resultList")).size());
...
```

Additionally, you can use “sender phone number” and “contents”.

- Setting “sender phone number”

```
searchVO.setSearchCnd(0);
searchVO.setSearchWrd("1234");
```

- Setting “contents”

```
searchVO.setSearchCnd(1);
searchVO.setSearchWrd("확인");
```

SMS Details

Business Rule

SMS Details calls the following interface methods.

Related Codes

N/A

Screen and execution manual

(The Implement object EgovSmsBasicServiceImpl)

Interface	Return	Methods	Parameters	Remarks
EgovSmsInfoService	SmsVO	selectSmsInf	SmsVO	

- Usage

```
SmsVO vo = new SmsVO();

vo.setSmsId("SMSID_00000000000001");

SmsVO detail = service.selectSmsInf(vo);

System.out.println("Detail action is successed...");
System.out.println("Transmit tel.: " + detail.getTrnsmitTelno());
System.out.println("Transmit contents: " + detail.getTrnsmitCn());
System.out.println("First Reg. ID: " + detail.getFrstRegisterId());
System.out.println("First Reg. Name: " + detail.getFrstRegisterNm());
System.out.println("First Reg. Time: " + detail.getFrstRegisterPnttm());
System.out.println("Receive: ");
List<SmsRecptn> list = detail.getRecptn();
for (SmsRecptn recptn: list) {
   System.out.println("\tTel.: " + recptn.getRecptnTelno());
   System.out.println("\tResult Code: " + recptn.getResultCode());
   System.out.println("\tResult msg.: " + recptn.getResultMssage());
}
...
```

The receiving number can be retrieved via List<SmsRecptn> as above.