Excel Service

1. Summary

The Excel Service is to support the uploading of mass amounts of Excel data to the system or downloading of data in the Excel file format by providing java library that can handle Excel file format. Excel service was implemented using **Apache POI** open source and has functions such as **Excel download**, **Excel file upload in addition to major Excel access** function.

2. Description

Main Function

Create Excel File

Provide the function to create Excel file and save in the designated location.
Cab additionally create Excel sheet by creating HSSFWorkbook instance.

Sample Source

```
HSSFWorkbook wb = new HSSFWorkbook();

HSSFSheet sheet1 = wb.createSheet(sheetName1);
HSSFSheet sheet2 = wb.createSheet(sheetName2);
HSSFSheet sheet = wb.createSheet();

// Create excel file
HSSFWorkbook tmp = excelService.createWorkbook(wb, filename);

// Check existence of file
assertTrue(EgovFileUtil.isExistsFile(sb.toString()));

// Examine agreement of saved sheet name
assertEquals(sheetName1, tmp.getSheetName(0));
assertEquals(sheetName2, tmp.getSheetName(1));
```

Update Excel File

Change and save the contents in the cell of the Excel file.
Saved Excel file can be loaded and text can be saved or updated by creating row and cell objects in the designated sheet.

Sample Source

```
// Excel file load
HSSFWorkbook wb = excelService.loadWorkbook(filename);

HSSFSheet sheet = wb.getSheetAt(0);
HSSFRow row = sheet.createRow( rownum );
row.setHeight( (short) 0x349 );
HSSFCell cell = row.createCell( cellnum );
cell.setCellType( HSSFCell.CELL_TYPE_STRING );
cell.setCellValue( new HSSFRichTextString(content) );
cell.setCellStyle( cs );

sheet.setColumnWidth( (short) 20, (short) ( ( 50 * 8 ) / ( (double) 1 / 20 ) ) );

FileOutputStream out = new FileOutputStream(filename);
wb.write(out);
out.close();
```

Update Excel File Property

Update the property of excel file(size of cell, property of Border, color of cell and alignment, etc.).
Can set font, color and arrangement applied to cell using HSSFFont, HSSFCellStyle class.

Sample Source

```
HSSFWorkbook wb = new HSSFWorkbook();

HSSFSheet sheet1 = wb.createSheet("new sheet");
HSSFSheet sheet2 = wb.createSheet("second sheet");

// Size of cell
sheet1.setDefaultRowHeight(rowheight);
sheet1.setDefaultColumnWidth(columnwidth);

HSSFFont f2 = wb.createFont();
HSSFCellStyle cs = wb.createCellStyle();
cs = wb.createCellStyle();

cs.setFont( f2 );
cs.setWrapText( true );

// Arrangement
cs.setAlignment(HSSFCellStyle.ALIGN_RIGHT);
cs.setFillPattern(HSSFCellStyle.DIAMONDS); // pattern style

// Color of cell
cs.setFillForegroundColor(new HSSFColor.BLUE().getIndex()); // pattern color
cs.setFillBackgroundColor(new HSSFColor.RED().getIndex());   // background color

sheet1.setDefaultColumnStyle((short) 0, cs);
```

Update Excel Document Property

Update the property of Excel file document(Header, Footer).
As HSSFHeader and HSSFFooter class, it can set the property and value of Header and Footer of excel file.

Sample Source

```
// Load Excel file
HSSFWorkbook wb = excelService.loadWorkbook(filename);
HSSFSheet sheet = wb.createSheet("doc test sheet");

HSSFRow row = sheet.createRow(1);
HSSFCell cell = row.createCell((short) 1);
cell.setCellValue(new HSSFRichTextString("Header/Footer Test"));

// Header
HSSFHeader header = sheet.getHeader();
header.setCenter("Center Header");
header.setLeft("Left Header");
header.setRight(HSSFHeader.font("Stencil-Normal", "Italic") + HSSFHeader.fontSize((short) 16) +
"Right Stencil-Normal Italic font and size 16");

// Footer
HSSFFooter footer = sheet.getFooter();
footer.setCenter(HSSFHeader.font("Fixedsys", "Normal") + HSSFHeader.fontSize((short) 12) + "- 1 -
");
footer.setLeft("Left Footer");
footer.setRight("Right Footer");
```

```
// Save Excel file
FileOutputStream out = new FileOutputStream(sb.toString());
wb.write(out);
out.close();
```

Extract Cell Contents

Read the Excel file and get the value of specific cell.
Can extract cell contents of various types including getRichStringCellValue, getNumericCellValue, getStringCellValue of HSSFCell class.

Sample Source

```
HSSFWorkbook wbT = excelService.loadWorkbook(filename);
HSSFSheet sheetT = wbT.getSheet("cell test sheet");

for (int i = 0; i < 100; i++) {
        HSSFRow row1 = sheetT.getRow(i);

        for (int j = 0; j < 5; j++) {
                HSSFCell cell1 = row1.getCell((short) j);
                assertEquals("row " + i + ", cell " + j, cell1.getRichStringCellValue().toString());
        }
}
```

Extract Cell Property

Update property of specific cell (font, size).
Can update properties of cell including font and size of cell using classes such as HSSFFont, HSSFCellStyle.

Sample Source

```
// Load Excel file
HSSFWorkbook wb = excelService.loadWorkbook(sb.toString());

HSSFSheet sheet = wb.createSheet("cell test sheet2");
sheet.setColumnWidth((short) 3, (short) 200);          // column Width

HSSFCellStyle cs = wb.createCellStyle();
HSSFFont font = wb.createFont();
font.setFontHeight((short) 16);
font.setBoldweight((short) 3);
font.setFontName("fixedsys");

cs.setFont(font);
cs.setAlignment(HSSFCellStyle.ALIGN_RIGHT);          // arrange cell
cs.setWrapText( true );

for (int i = 0; i < 100; i++) {
        HSSFRow row = sheet.createRow(i);
        row.setHeight((short)300); // height setting of row

        for (int j = 0; j < 5; j++) {
                HSSFCell cell = row.createCell((short) j);
                cell.setCellValue(new HSSFRichTextString("row " + i + ", cell " + j));
                cell.setCellStyle( cs );
        }
}

// Save Excel file
```

```
FileOutputStream out = new FileOutputStream(sb.toString());
wb.write(out);
out.close();
```

Use Common Template

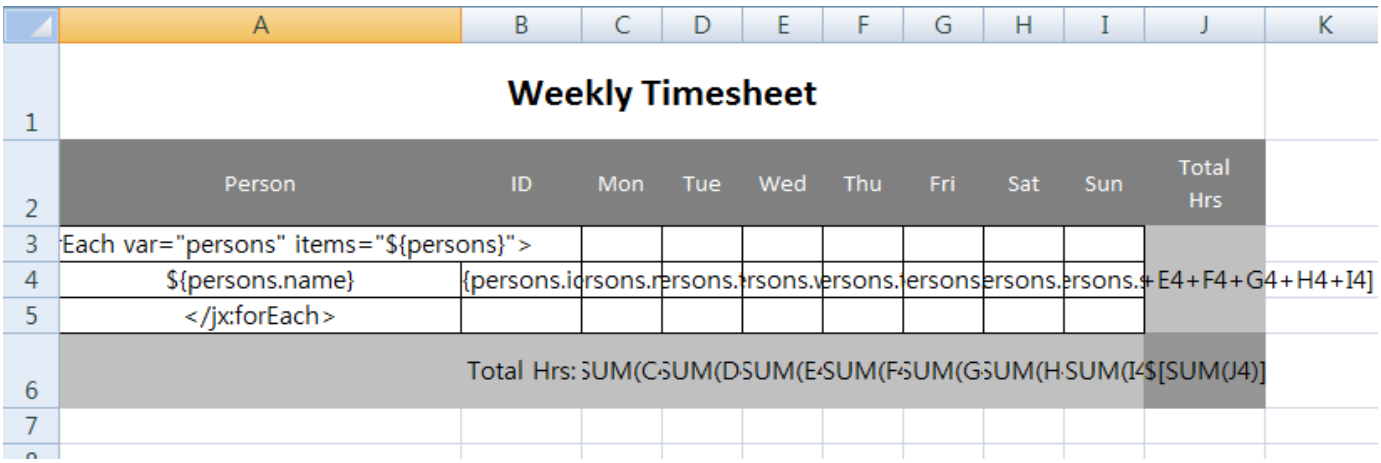
Keep the consistency using common template. Save the value designated in the template using jXLS open source.

Sample Source

```
List<PersonHourVO> persons = new ArrayList<PersonHourVO>();
PersonHourVO person = new PersonHourVO();
person.setName("Yegor Kozlov");
person.setId("YK");
person.setMon(5.0);
person.setTue(8.0);
person.setWed(10.0);
person.setThu(5.0);
person.setFri(5.0);
person.setSat(7.0);
person.setSun(6.0);

persons.add(person);

PersonHourVO person1 = new PersonHourVO();
person1.setName("Gisella Bronzetti");
person1.setId("GB");
person1.setMon(4.0);
person1.setTue(3.0);
person1.setWed(1.0);
person1.setThu(3.5);
person1.setSun(4.0);

persons.add(person1);

Map<String, Object> beans = new HashMap<String, Object>();
beans.put("persons", persons);
XLSTransformer transformer = new XLSTransformer();

transformer.transformXLS(filename, beans, sbResult.toString());
```

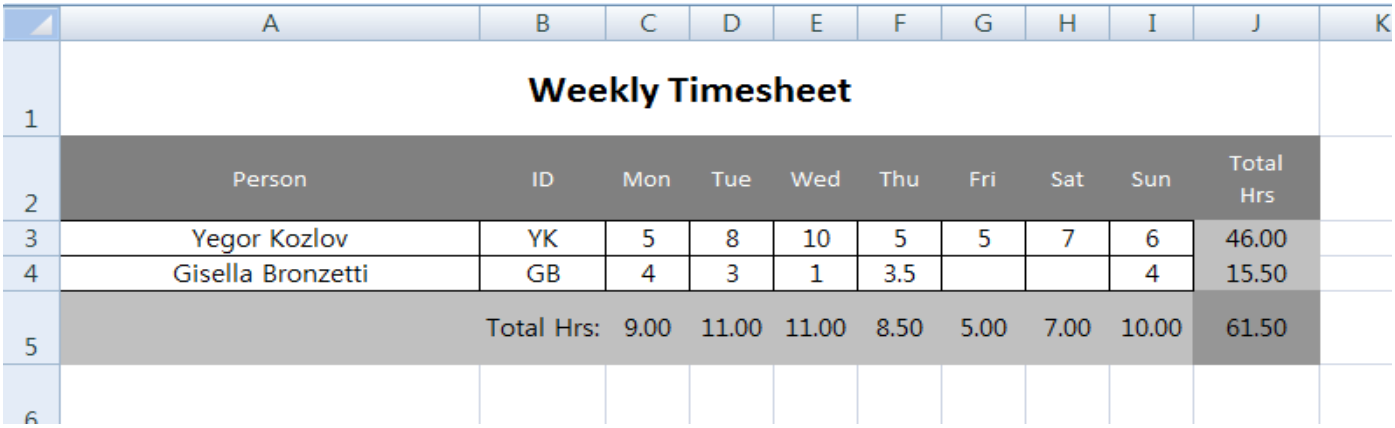
Excel Template

```
<jx:forEach var="persons" items="${persons}">
${persons.name}${persons.id}     ${persons.mon}  ${persons.tue}    ${persons.wed}
         ${persons.thu}    ${persons.fri}     ${persons.sat}    ${persons.sun}
         $[C4+D4+E4+F4+G4+H4+I4]
</jx:forEach>
                  Total Hrs:          $[SUM(C4)]       $[SUM(D4)]       $[SUM(E4)]       $[SUM(F4)]
$[SUM(G4)]       $[SUM(H4)]       $[SUM(I4)]       $[SUM(J4)]
```



	A	B	C	D	E	F	G	H	I	J	K
1	Weekly Timesheet										
2	Person	ID	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total Hrs	
3	Each var="persons" items="${persons}">										
4	${persons.name}	{persons.id	rsons.m	ersons.t	rsons.w	ersons.t	ersons	ersons.s	ersons.s	E4+F4+G4+H4+I4]	
5	</jx:forEach>										
6		Total Hrs:	SUM(C	SUM(D	SUM(E	SUM(F	SUM(G	SUM(H	SUM(I	$[SUM(J4)]	
7											

Results of Template Application



	A	B	C	D	E	F	G	H	I	J	K
1	Weekly Timesheet										
2	Person	ID	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Total Hrs	
3	Yegor Kozlov	YK	5	8	10	5	5	7	6	46.00	
4	Gisella Bronzetti	GB	4	3	1	3.5			4	15.50	
5		Total Hrs:	9.00	11.00	11.00	8.50	5.00	7.00	10.00	61.50	
6											

Download Excel

Configuration

```
<bean id="categoryExcelView" class="egovframework.rte.fdl.excel.download.CategoryExcelView" />

<bean class="org.springframework.web.servlet.view.BeanNameViewResolver">
        <property name="order" value="0" />
</bean>
```

Sample Source

Use Controller Class Creation Map

```
@RequestMapping("/sale/listExcelCategory.do")
public ModelAndView selectCategoryList() throws Exception {

        List<Map> lists = new ArrayList<Map>();

        Map<String, String> mapCategory = new HashMap<String, String>();
        mapCategory.put("id", "0000000001");
        mapCategory.put("name", "Sample Test");
        mapCategory.put("description", "This is initial test data.");
        mapCategory.put("useyn", "Y");
        mapCategory.put("reguser", "test");

        lists.add(mapCategory);

        mapCategory.put("id", "0000000002");
        mapCategory.put("name", "test Name");
        mapCategory.put("description", "test Deso1111");
```

```
		mapCategory.put("useyn", "Y");
		mapCategory.put("reguser", "test");

		lists.add(mapCategory);

		Map<String, Object> map = new HashMap<String, Object>();
		map.put("category", lists);

		return new ModelAndView("categoryExcelView", "categoryMap", map);
}
```

Use VO

```
@RequestMapping("/sale/listExcelVOCategory.do")
public ModelAndView selectCategoryVOList() throws Exception {

		List<UsersVO> lists = new ArrayList<UsersVO>();

		UsersVO users = new UsersVO();

		//Map<String, String> mapCategory = new HashMap<String, String>();
		users.setId("0000000001");
		users.setName("Sample Test");
		users.setDescription("This is initial test data.");
		users.setUseYn("Y");
		users.setRegUser("test");

		lists.add(users);

		users.setId("0000000002");
		users.setName("test Name");
		users.setDescription("test Deso1111");
		users.setUseYn("Y");
		users.setRegUser("test");

		lists.add(users);

		Map<String, Object> map = new HashMap<String, Object>();
		map.put("category", lists);

		return new ModelAndView("categoryExcelView", "categoryMap", map);
}
```

Create View Class

```
public class CategoryExcelView extends AbstractExcelView {
		@Override
		protected void buildExcelDocument(Map model, HSSFWorkbook wb,
				HttpServletRequest req, HttpServletResponse resp) throws Exception {
			HSSFCell cell = null;

			HSSFSheet sheet = wb.createSheet("User List");
			sheet.setDefaultColumnWidth((short) 12);

			// put text in first cell
			cell = getCell(sheet, 0, 0);
			setText(cell, "User List");

			// set header information
			setText(getCell(sheet, 2, 0), "id");
			setText(getCell(sheet, 2, 1), "name");
			setText(getCell(sheet, 2, 2), "description");
```

```
                setText(getCell(sheet, 2, 3), "use_yn");
                setText(getCell(sheet, 2, 4), "reg_user");

                Map<String, Object> map= (Map<String, Object>) model.get("categoryMap");
                List<Object> categories = (List<Object>) map.get("category");

                boolean isVO = false;

                if (categories.size() > 0) {
                        Object obj = categories.get(0);
                        isVO = obj instanceof UsersVO;
                }

                for (int i = 0; i < categories.size(); i++) {

                        if (isVO) {          // VO

                                Map<String, String> category = (Map<String, String>)
categories.get(i);

                                cell = getCell(sheet, 3 + i, 0);
                                setText(cell, category.get("id"));

                                cell = getCell(sheet, 3 + i, 1);
                                setText(cell, category.get("name"));

                                cell = getCell(sheet, 3 + i, 2);
                                setText(cell, category.get("description"));

                                cell = getCell(sheet, 3 + i, 3);
                                setText(cell, category.get("useyn"));

                                cell = getCell(sheet, 3 + i, 4);
                                setText(cell, category.get("reguser"));
                        } else {  // Map

                                Map<String, String> category = (Map<String, String>)
categories.get(i);

                                cell = getCell(sheet, 3 + i, 0);
                                setText(cell, category.get("id"));

                                cell = getCell(sheet, 3 + i, 1);
                                setText(cell, category.get("name"));

                                cell = getCell(sheet, 3 + i, 2);
                                setText(cell, category.get("description"));

                                cell = getCell(sheet, 3 + i, 3);
                                setText(cell, category.get("useyn"));

                                cell = getCell(sheet, 3 + i, 4);
                                setText(cell, category.get("reguser"));

                        }
                }
        }
}
```

Upload Excel

Configuration

```
<bean id="excelService" class="egovframework.rte.fdl.excel.impl.EgovExcelServiceImpl">
        <property name="propertyPath" value="excelInfo.xml" />
        <property name="mapClass"
value="egovframework.rte.cvpl.service.impl.EgovExcelTestMapping" />
        <property name="sqlMapClient" ref="sqlMapClient" />
</bean>
```

- class: egovframework.rte.fdl.excel.impl.EgovExcelServiceImpl
- propertyPath: location of Excel format information in xml format
- mapClass: class for mapping of Query and VO created by developer
- sqlMapClient: sqlMapClient of ibatis

Sample Source

Create VO Class

```
public class EmpVO implements Serializable {

        private BigDecimal empNo;
        private String empName;
        private String job;

        public BigDecimal getEmpNo() {
                return empNo;
        }

  public void setEmpNo(BigDecimal empNo) {
        this.empNo = empNo;
  }

        public String getEmpName() {
                return empName;
        }

        public void setEmpName(String empName) {
                this.empName = empName;
        }

        public String getJob() {
                return job;
        }

        public void setJob(String job) {
                this.job = job;
        }
}
```

- VO Class for Mapping of Query and Excel

Create Mapping Class

```
public class EgovExcelTestMapping extends EgovExcelMapping {

        @Override
        public EmpVO mappingColumn(HSSFRow row) {
                HSSFCell cell0 = row.getCell((short) 0);
                HSSFCell cell1 = row.getCell((short) 1);
                HSSFCell cell2 = row.getCell((short) 2);

                EmpVO vo = new EmpVO();

                vo.setEmpNo(new BigDecimal(cell0.getNumericCellValue()));
```

```
                vo.setEmpName(cell1.getRichStringCellValue().toString());
                vo.setJob(cell2.getRichStringCellValue().toString());

                return vo;
        }
}
```

- Mapping class for mapping of VO and Excel
- Implement by inheriting **EgovExcelMapping** class and overriding **mappingColumn** method
- VO and mapping for executing Query by extracting excel value in **HSSFCell** class

Query

```
<sqlMap namespace="EmpBatchInsert">
        <typeAlias alias="empVO" type="egovframework.rte.fdl.excel.vo.EmpVO" />
        <insert id="insertEmpUsingBatch" parameterClass="empVO">
                <![CDATA[
                        insert into EMP (
                                EMP_NO,
                                EMP_NAME,
                                JOB
                        ) values (
                                #empNo#,
                                #empName#,
                                #job#
                        )
                ]]>
        </insert>
</sqlMap>
```

N. Reference