

전자정부 표준프레임워크 [별첨]실행환경 (공통기반)



1. 별첨 공통기반 레이어

1. Cache
2. Compress/Decompress
3. Encryption/Decryption
4. Excel
5. File Handling
6. File Upload/Download
7. FTP
8. Mail
9. Marshalling/Unmarshalling
10. Object Pooling
11. Property
12. Resource
13. Scheduling
14. Server Security
15. String Util
16. Xml Manipulation



□ 서비스 개요

- 빈번히 사용되는 콘텐츠에 대해서 빠른 접근을 가능하게 하여 잦은 접근을 통한 오버헤드나 시간을 절약하기 위한 서비스

□ 주요 기능

- Cache를 위한 속성 설정
- Cache 매체(메모리, 디스크 등)를 지정
- Cache 사이즈(Disk 용량, Object 수 등)를 지정
- Cache 알고리즘(LRU, FIFO 등)을 지정
- Cache에 대상을 입력
- Cache로부터 대상을 얻는 기능
- 클러스터링을 지원하는 기능

□ Cache 서비스

- Cache Service는 **EhCache**를 선정.
- Spring 3.1부터 Cache추상화를 제공하여 Java메소드에 Cache를 적용할 수 있다.
- 변경이 자주 일어나지 않지만 사용빈도가 높고 생성하는데 비용이 많이 드는 객체일 경우, Cache를 이용하면 다음과 같은 장점을 얻을 수 있음.
 - 자주 접근하는 데이터를 매번 데이터베이스로부터 fetch할 필요가 없으므로 오버헤드가 줄어듦.
 - 객체를 매번 생성하지 않기 때문에 메모리를 효율적으로 사용할 수 있음.

□ 기본 사용법

- Cache를 사용하기 위한 Cache Manager 생성 샘플

//클래스 패스를 이용하여 설정파일 읽어서 Cache Manager 생성하기.

```
URL url = getClass().getResource("/ehcache-default.xml");  
manager = new CacheManager(url);
```

□ 기본 사용법

– Configuration

```
<ehcache>
<diskStore path="user.dir/second"/>
  <cache name="sampleMem"
    maxElementsInMemory="3"
    eternal="false"
    timeToIdleSeconds="360"
    timeToLiveSeconds="1000"
    overflowToDisk="false"
    diskPersistent="false"
    memoryStoreEvictionPolicy="LRU">
  </cache>
</ehcache>
```

– Basic Usage Source

- 위에서 정의한 Cache Manager를 사용하여 Cache에 데이터를 입력/조회/삭제하는 기본 샘플 코드

```
// cache Name을 가지고 cache 얻기
Cache cache = manager.getCache("sampleMem");

// 1.Cache에 데이터 입력
cache.put(new Element("key1", "value1"));

// 2.Cache로부터 입력한 데이터 읽기
Element value = cache.get("key1");

// 3. Cache에서 데이터 삭제
cache.remove("key1");
```

❑ Cache 추상화(1/2)

– Configuration

- Cache abstraction을 쓰기 위한 <cache:annotation-driven/>과 cacheManager설정

```
<cache:annotation-driven cache-manager="cacheManager" />

<!-- EhCache를 저장소로 사용하는 Cache Manager -->
<bean id="cacheManager" class="org.springframework.cache.ehcache.EhCacheCacheManager">
  <property name="cacheManager" ref="ehcache"/>
</bean>

<!-- Ehcache library setup -->
<bean id="ehcache" class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean"
  p:config-location="classpath:springframework/cache/ehcache/ehcache.xml" />
```

– Basic Usage Source

- cacheManager를 사용하는 방법

```
import org.springframework.cache.CacheManager;

...
@Resource(name="cacheManager")
CacheManager cacheManager;

...
Cache cache = cacheManager.getCache("ehcache");
```

❑ Cache 추상화(2/2)

– @Cacheable

- 별다른 조건없이 호출되는 모든 인자를 caching하고 할 때는 @Cacheable(“cache명”)을 써 주면 된다.
- 특정 key로 caching하거나 특정조건을 줄 수도 있다.

```
@Cacheable("books")
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)

@Cacheable(value="books", key="#isbn")
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)

@Cacheable(value="book", condition="#name.length < 32")
public Book findBook(String name)
```

– @CacheEvict

- @CacheEvict는 @Cacheable과 반대로 cache저장소의 데이터를 제거할 수 있다.

```
@CacheEvict(value = "books", allEntries=true)
public void loadBooks(InputStream batch)
```

❑ Cache Algorithm

- LRU Algorithm
- FIFO Algorithm
- LFU Algorithm

❑ Cache Size & Device

- Cache에서 관리해야 하는 정보의 사이즈 설정 및 저장 디바이스 관련 설정
 - **Cache Device** : 디바이스 관련 세팅은 메모리 관리 캐쉬의 디스크 관리로의 이동관련 설정으로 메모리 관리 오브젝트 수가 넘었을 때 Disk 로 이동여부, flush 호출시 파일로 저장 여부에 대한 세팅이 있음
 - **Cache Size** : 사이즈 관련 세팅은 메모리에서 관리해야 할 최대 오브젝트 수, 디스크에서 관리하는 최대 오브젝트 수에 대한 세팅

❑ Distributed Cache

- Ehcache는 Distributed Cache를 지원하는 방법으로 RMI,JGROUP,JMS등을 지원함
 - **Using Jgroups** : JGroups는 multicast 기반의 커뮤니케이션 툴킷으로 그룹을 생성하고 그룹멤버간에 메시지를 주고 받을 수 있도록 지원함
 - **Using ActiveMQ** : ActiveMQ는 JMS 메시징 서비스를 제공하는 오픈 소스

☐ EhCache

- <http://ehcache.org/>

□ 서비스 개요

- 데이터를 압축하고 복원하는 기능을 제공하는 서비스로서, 데이터를 효율적으로 저장하고 전송하기 위해서 원본 데이터를 압축하거나, 압축된 데이터를 복원하여 원본 데이터를 얻기 위한 서비스

□ 주요 기능

- 데이터 압축(Compress)
- 데이터의 복원을 위하여 비손실 압축 방식을 이용하여 압축
- 데이터 복원(Decompress)
- 압축된 데이터 복원

❑ Compress/Decompress 서비스

- 다양한 압축방식과 개발자들에게 편리한 API를 제공하는 Jakarta Commons의 Compress 를 사용
- Jakarta Commons의 Compress에서 지원하는 tar, zip and bzip2 파일등을 지원함

❑ Commons Compress에서 제공하고 있는 API

(<http://commons.apache.org/compress/apidocs/index.html>)

- org.apache.commons.compress.archivers
- org.apache.commons.compress.archivers.ar
- org.apache.commons.compress.archivers.cpio
- org.apache.commons.compress.archivers.jar
- org.apache.commons.compress.archivers.tar
- org.apache.commons.compress.archivers.zip
- org.apache.commons.compress.changes
- org.apache.commons.compress.compressors
- org.apache.commons.compress.compressors.bzip2
- org.apache.commons.compress.compressors.gzip
- org.apache.commons.compress.utils

❑ Commons Compress를 사용한 간단한 예제(Archive)

- org.apache.commons.compress.archivers 패키지에 속한 ArchiveInputStream 클래스를 사용하여 compress/decompress로 구성됨

```
.  
.  
// Archive  
final File output = new File(dir, "bla.zip");  
final File file1 = getFile("test1.xml");  
final File file2 = getFile("test2.xml");  
  
{  
    final OutputStream out = new FileOutputStream(output);  
    final ArchiveOutputStream os = new  
        ArchiveStreamFactory().createArchiveOutputStream("zip", out);  
  
    os.putArchiveEntry(new ZipArchiveEntry("testdata/test1.xml"));  
    IOUtils.copy(new FileInputStream(file1), os);  
    os.closeArchiveEntry();  
  
    os.putArchiveEntry(new ZipArchiveEntry("testdata/test2.xml"));  
    IOUtils.copy(new FileInputStream(file2), os);  
    os.closeArchiveEntry();  
    os.close();  
}  
.  
.  
.
```

❑ Commons Compress를 사용한 간단한 예제(Unarchive)

- org.apache.commons.compress.archivers 패키지에 속한 ArchiveInputStream 클래스를 사용하여 compress/decompress로 구성됨

```
//Unarchive
List results = new ArrayList();

{
    final InputStream is = new FileInputStream(output);
    final ArchiveInputStream in = new ArchiveStreamFactory().
        createArchiveInputStream("zip", is);

    File result = File.createTempFile("dir-result", "");
    result.delete();
    result.mkdir();

    ZipArchiveEntry entry = null;
    while((entry = (ZipArchiveEntry)in.getNextEntry()) != null) {
        File outfile = new File(result.getCanonicalPath() + "/result/" + entry.getName());
        outfile.getParentFile().mkdirs();
        OutputStream out = new FileOutputStream(outfile);
        IOUtils.copy(in, out);
        out.close();
        results.add(outfile);
    }
    in.close();
}
```

❑ Apache Commons Compress

- <https://commons.apache.org/proper/commons-compress/index.html>

❑ Commons Compress 1.21.0 API

- <https://javadoc.io/doc/org.apache.commons/commons-compress/1.21/index.html>

□ 서비스 개요

- 암호화(Encryption,Ciphering)는 메시지의 내용이 불명확하도록 평문을 재구성하여 암호문을 만드는 것인데, 이 때 사용되는 메시지의 재구성 방법을 암호화 알고리즘(Encryption Algorithm)이라고 부른다. 암호화 알고리즘에서는 암호화의 비밀성을 높이기 위해 키(Key)를 사용하기도 한다. 복호화(Decryption,deciphering)란 암호화의 역과정으로, 불명확한 메시지로부터 본래의 메시지를 환원하는 과정이다.
- Jasypt는 오픈소스 Java library로 개발자는 암호화 관련 깊은 지식이 없어도 암호복호화 프로그램을 개발할 수 있도록 지원한다. 여기서 설명하는 부분은 암호복호화 모듈로 사용한 API 중심으로 설명하겠다.

□ 주요 기능

- Environment Cryptography Service
 - 외부환경설정파일(properties)의 키와 대칭되는 값에 대해 인코딩 · 디코딩 기능을 지원한다
- Password Encoder
 - Jasypt의 ConfigurablePasswordEncryptor를 통해 패스워드 인코딩을 지원한다. (기본 알고리즘: SHA-256)
- General Cryptography Service
 - setAlgorithm(...), setBlockSize(...), setPasswordEncoder(...) 메소드를 통해 암호복호화에 대한 알고리즘 등을 설정한다.
 - 기본 알고리즘은 PBESWithSHA1AndDESede, 기본 블록사이즈는 1024로 처리된다.
 - byte[], BigDecimal, File에 대하여 암호화 및 복호화 서비스를 제공한다.

- ARIA Algorithm Cryptography Service
 - 국정원의 ARIA 알고리즘을 통한 암호화 서비스를 제공한다.
 - setBlockSize(...), setPasswordEncoder(...) 메소드를 통해 암호화에 대한 정보를 설정한다.
 - 기본 블록사이즈는 1024로 처리된다.
 - byte[], File에 대하여 암호화 및 복호화 서비스를 제공한다
- Digest Service
 - Hash Function을 통한 Message Digest 서비스를 제공한다.
 - setAlgorithm(...), setPlainDigest(...) 메소드를 통해 Digest 방식에 대한 정보를 설정한다.
 - 기본 알고리즘은 SHA-256이며, 기본 PlainDigest 설정은 false로 처리된다. (PainDigest가 true이면 1000 iterations, 8 salt byte size가 적용된다.)
 - byte[]에 대한 digest 및 matches 서비스가 제공된다.

– Property 설정

```
# Message digest algorithm using EgovPasswordEncoder..  
crypto.password.algorithm=SHA-256  
  
# hashed password (ex: egovframe (SHA-256) => gdyYs/IZqY86VcWhT8emCYfqY1ahw2vtLG+/FzNqtrQ=)  
crypto.hashed.password=gdyYs/IZqY86VcWhT8emCYfqY1ahw2vtLG+/FzNqtrQ=
```

- crypto.password.algorithm : 비밀번호 인코더에 사용될 hash function 알고리즘
- crypto.hashed.password : 비밀번호에 대한 hash value (org.egovframe.rte.fdl.cryptography.EgovPasswordEncoder의 main 메소드에 의해 해당 값을 얻어 기록한다.)

– Properties 파일 지정

```
<context:property-placeholder  
location="classpath*/META-INF/spring/crypto_config.properties,classpath*/META-INF/spring/password.properties" />  
<!-- recommended location method is using file prefix.. ex) "file:/home/properties/crypto_config.properties" -->
```

– XML 설정

```
<bean id="passwordEncoder" class="org.egovframe.rte.fdl.cryptography.EgovPasswordEncoder">
<property name="algorithm" value="${crypto.password.algorithm}" /><!-- default : SHA-256 -->
<property name="hashedPassword" value="${crypto.hashed.password}" />
</bean>

<bean id="ARIACryptoService" class="org.egovframe.rte.fdl.cryptography.impl.EgovARIACryptoServiceImpl">
<property name="passwordEncoder" ref="passwordEncoder" />
<property name="blockSize" value="1025" /><!-- default : 1024 -->
</bean>

<bean id="digestService" class="org.egovframe.rte.fdl.cryptography.impl.EgovDigestServiceImpl">
<property name="algorithm" value="SHA-256" /><!-- default : SHA-256 -->
<property name="plainDigest" value="false" /><!-- default : false -->
</bean>

<bean id="generalCryptoService" class="org.egovframe.rte.fdl.cryptography.impl.EgovGeneralCryptoServiceImpl">
<property name="passwordEncoder" ref="passwordEncoder" />
<property name="algorithm" value="PBEWithSHA1AndDESede" /><!-- default : PBEWithSHA1AndDESede -->
<property name="blockSize" value="1024" /><!-- default : 1024 -->
</bean>
```

- Encryption texts Guide Program

```
@Test
public void testString() {
    String[] testString = { "This is a testing...\nHello!",
        "한글 테스트입니다...",
        "!@#$$%^&*()_+|~{}:\\">?-=\\\'[];\'./" };

    try {
        For (String str : testString) {
            byte[] encrypted = cryptoService.encrypt(str.getBytes("UTF-8"), password);

            byte[] decrypted = cryptoService.decrypt(encrypted, password);

            assertEquals(str, new String(decrypted, "UTF-8"));
        }
    } catch (UnsupportedEncodingException uue) {
        uue.printStackTrace();
        fail();
    }
}
```

– Encryption file Guide Program

```
@Test
public void testFile() {
    String filePath = "/META-INF/spring/file/test.hwp";
    File srcFile = new File(this.getClass().getResource(filePath).getFile());

    File trgtFile;
    File decryptedFile;
    try {
        trgtFile = File.createTempFile("tmp", "encrypted");
        trgtFile.deleteOnExit();

        cryptoService.encrypt(srcFile, password, trgtFile);

        decryptedFile = File.createTempFile("tmp", "decrypted");
        decryptedFile.deleteOnExit();

        cryptoService.decrypt(trgtFile, password, decryptedFile);

        assertTrue("Decrypted file not same!!",
            checkFileWithHashFunction(srcFile, decryptedFile));
    } catch (Exception ex) {
        ex.printStackTrace();
        fail(ex.getMessage());
    }
}
```

– Message Digest Guide Program

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "classpath:/META-INF/spring/*.xml" })
public class EgovDigestServiceTest {
    @Resource(name="digestService")
    EgovDigestService digestService;

    @Test
    public void testDigest() {
        String data = "egovframe";

        byte[] digested = digestService.digest(data.getBytes());

        assertTrue(digestService.matches(data.getBytes(), digested));
    }
}
```

❑ JASYPT class API

- <http://www.jasypt.org/api/jasypt/1.9.3/index.html>

□ 서비스 개요

- Excel 파일 포맷을 다룰 수 있는 자바 라이브러리를 제공하여 Java 어플리케이션 에서도 Excel 파일을 다룰 수 있는 서비스
- Apache POI 오픈소스를 사용하여 Excel파일접근 및 Excel업/다운로드 기능을 서비스함

□ 주요 기능

- 엑셀 파일을 읽어 특정 셀의 값 조회
- 엑셀 파일 내 특정 셀의 내용을 변경
- 엑셀 파일 내 특정 셀의 속성(폰트, 사이즈 등) 변경
- 엑셀 파일 생성
- 엑셀 파일 다운로드
- 엑셀 파일의 속성(셀의 크기, Border의 속성, 셀의 색상, 정렬 등) 변경
- 엑셀 파일 문서의 속성(Header, Footer)을 변경
- 공통 템플릿 사용을 통한 일관성 제공

□ 주요기능

- Excel 파일 생성
 - HSSFWorkbook 인스턴스를 생성하여 Excel sheet를 추가 생성

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

// 엑셀 파일 생성
HSSFWorkbook tmp = excelService.createWorkbook(wb, filename);

// 파일 존재 확인
assertTrue(EgovFileUtil.existsFile(sb.toString()));

// 저장된 Sheet명 일치 점검
assertEquals(sheetName1, tmp.getSheetName(0));
assertEquals(sheetName2, tmp.getSheetName(1));
```


□ 주요기능

- Excel 파일 수정
 - 저장된 엑셀파일을 로드할 수 있음
 - 지정한 sheet에 row와 cell 객체를 생성하여 텍스트를 저장하고 수정할 수 있음

// 엑셀 파일 로드

```
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();
```

□ 주요기능

– Excel 파일 속성수정

- 엑셀 파일의 속성(셀의 크기, Border의 속성, 셀의 색상, 정렬 등)을 수정
- HSSFWorkbook, HSSFCellStyle 클래스를 이용하여 셀에 적용된 폰트, 색상 및 정렬 등을 설정할 수 있음

```
HSSFWorkbook wb = new HSSFWorkbook();

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

// 셀의 크기
sheet1.setDefaultRowHeight(rowheight);
sheet1.setDefaultColumnWidth(columnwidth);

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

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

// 정렬
cs.setAlignment(HSSFCellStyle.ALIGN_RIGHT);
cs.setFillPattern(HSSFCellStyle.DIAMONDS); // 무늬 스타일

// 셀의 색상
cs.setFillForegroundColor( new HSSFColor.BLUE().getIndex()); // 무늬 색
cs.setFillBackgroundColor( new HSSFColor.RED().getIndex()); // 배경색

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

□ 주요기능

– Excel 문서 속성수정

- 엑셀 파일 문서의 속성(Header/Footer)을 수정
- HSSFWorkbook, HSSFHeader 클래스로 엑셀문서의 Header와 Footer의 값과 속성을 설정

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

// 엑셀 파일 저장
FileOutputStream out = new FileOutputStream(sb.toString());
wb.write(out);
out.close();
```

□ 주요기능

– 셀 내용 추출

- 엑셀 파일을 읽어 특정 셀의 값을 얻어옴
- HSSFCell 클래스의 getRichStringCellValue, getNumericCellValue, getStringCellValue 등 다양한 type의 Cell 내용을 추출

```
// 엑셀 파일 로드
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);
// cell 정렬
cs.setWrapText(true);
for (int i = 0; i < 100; i++) {
    HSSFRow row = sheet.createRow(i);
    row.setHeight((short) 300);
    // row의 height 설정
    for (int j = 0; j < 5; j++) {
        HSSFCell cell = row.createCell((short) j);
        cell.setCellValue(new HSSFRichTextString("row " + i + ", cell " + j));
        cell.setCellStyle(cs);
    }
}
// 엑셀 파일 저장
FileOutputStream out = new FileOutputStream(sb.toString());
wb.write(out);
out.close();
```

□ 주요기능

– 셀 속성 추출

- 특정 셀의 속성(폰트, 사이즈 등)을 수정
- HSSFWorkbook, HSSFCellStyle 등의 클래스를 이용하여 셀의 폰트, 사이즈 등의 셀 속성을 수정

```
// 엑셀 파일 로드
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();
HSSFWorkbook font = wb.createFont();
font.setFontHeight((short) 16);
font.setBoldweight((short) 3);
font.setFontName("fixedsys");

cs.setFont(font);
cs.setAlignment(HSSFCellStyle.ALIGN_RIGHT);
// cell 정렬
cs.setWrapText( true );
for (int i = 0; i < 100; i++) {
    HSSFRow row = sheet.createRow(i);
    row.setHeight((short)300); // row의 height 설정

    for (int j = 0; j < 5; j++) {
        HSSFCell cell = row.createCell((short) j);
        cell.setCellValue(new HSSFRichTextString("row " + i + ", cell " + j));
        cell.setCellStyle( cs );
    }
}
// 엑셀 파일 저장
FileOutputStream out = new FileOutputStream(sb.toString());
wb.write(out);
out.close();
```

□ 주요기능

– 공통템플릿 사용

- 공통 템플릿을 사용하여 일관성을 유지한다. jXLS 오픈소스를 사용하여 작성된 템플릿에 지정된 값을 저장

```
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());
```

□ 주요기능

– 공통템플릿 사용

- 공통 템플릿을 사용하여 일관성을 유지한다. jXLS 오픈소스를 사용하여 작성된 템플릿에 지정된 값을 저장

| | 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} | {persons.mon} | {persons.tue} | {persons.wed} | {persons.thu} | {persons.fri} | {persons.sat} | {persons.sun} | E4+F4+G4+H4+I4] | |
| 5 | </jx:forEach> | | | | | | | | | | |
| 6 | Total Hrs: SUM(C4:SUM(D4:SUM(E4:SUM(F4:SUM(G4:SUM(H4:SUM(I4:SUM(J4)) | | | | | | | | | | |
| 7 | | | | | | | | | | | |
| 8 | | | | | | | | | | | |
| | 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 | | | | | | | | | | | |

□ 엑셀다운로드

– Configuration

```
<bean id="categoryExcelView" class="org.egovframe.rte.fdl.excel.download.CategoryExcelView" />
    <bean class="org.springframework.web.servlet.view.BeanNameViewResolver">
        <property name="order" value="0" />
    </bean>
```

– Controller 클래스(Map을 사용할 경우)

```
@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);

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

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


□ 엑셀다운로드

- Controller 클래스(VO를 사용한 경우)

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

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

    UsersVO users = new UsersVO();
    users.setId("0000000001");
    users.setName("Sample Test");
    users.setDescription("This is initial test data.");
    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);
}
```

□ 엑셀다운로드

– ExcelView 클래스

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

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

        HSSFCell cell = getCell(sheet, 0, 0);
        setText(cell, "User List");

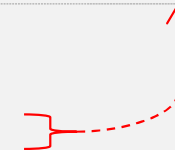
        setText(getCell(sheet, 2, 0), "id");
        setText(getCell(sheet, 2, 1), "name");
        .
        .

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

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

        for (int i = 0; i < categories.size(); i++) {
            Map<String, String> category = (Map<String, String>) categories.get(i);
            cell = getCell(sheet, 3 + i, 0);
            setText(cell, category.get("id"));
        }
    }
}
```

Controller의 return한 객체의
파라미터



□ 엑셀업로드

– Configuration

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

– VO 작성

```
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; }
}
```

□ 엑셀업로드

– Mapping 클래스

```
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;
    }
}
```

– Query

```
<sqlMap namespace="EmpBatchInsert">
  <typeAlias alias="empVO" type="org.egovframe.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>
```

```
excelService.uploadExcel("insertEmpUsingBatch", fileInputStream);
```

❑ The Apache POI Project

- <http://poi.apache.org>

❑ jXLS

- <http://jxls.sourceforge.net/>

□ 서비스 개요

- File에 대한 생성 및 접근, 변경 등과 같이 File을 Access하는 기능을 제공하는 서비스
- 시스템을 개발할 때 필요한 문자열 데이터를 다루기 위해 다양한 기능을 사용하도록 서비스함

□ 주요 기능

- File 생성
 - 특정 위치에 File 생성
- File Access
 - File에 대한 접근 및 수정
 - 파일 위치는 절대 경로, 상대 경로 등 다양한 형식 지원
 - File을 읽고 수정할 때 File Caching 기능 제공

□ Jakarta Commons Virtual File System(VFS)란

- 파일처리를 추상화 하여 간편하고 쉽게 사용할 수 있는 아파치의 Jakarta Commons Virtual File System (VFS)를 File Handling 기능의 기반 오픈 소스로 채택함
- Jakarta Commons Virtual File System (VFS)는 파일 시스템에 접근하기 위한 API를 제공하며, Local disk, HTTP server, ZIP 파일 내부 등 다양한 파일시스템에 대한 단일한 view를 제공
 - 다양한 파일시스템에 대한 일관된 접근 API를 제공한다.
 - 다양한 파일시스템을 지원한다.
 - 다양한 파일시스템을 하나의 로컬 파일 시스템처럼 접근 가능하도록 한다.
 - 파일 접근 시 파일 캐싱을 통하여 성능을 향상 시킨다.

❑ FileSystemManager

- Commons Virtual File System (VFS)을 사용하기 위해서는 **FileSystemManager** 인터페이스가 필요
- **FileSystemManager** 인스턴스를 얻는 가장 간단한 방법은 `VFS.getManager()` 스태틱 메소드를 사용

```
FileSystemManager fsManager = VFS.getManager();
FileObject jarFile = fsManager.resolveFile( "jar:lib/aJarFile.jar" );
```

❑ FileObject

- 파일 및 폴더를 **FileObject** 인스턴스로 표현하여 파일을 생성/삭제하거나 파일 리스트 검색 및 파일 읽기/쓰기 등 가능

```
// Locate the Jar file
FileSystemManager fsManager = VFS.getManager();
FileObject jarFile = fsManager.resolveFile( "jar:lib/aJarFile.jar" );
// List the children of the Jar file
FileObject[] children = jarFile.getChildren();
System.out.println( "Children of " + jarFile.getName().getURI() );
for ( int i = 0; i < children.length; i++ ) {
    System.out.println( children[ i ].getName().getBaseName() );
}
```


□ 파일생성

- FileObject의 createFile() 메소드 사용

```
public void testCeateFile() throws Exception {  
    FileSystemManager manager = VFS.getManager();  
    FileObject baseDir = manager.resolveFile(System.getProperty("user.dir"));  
    final FileObject file = manager.resolveFile(baseDir, "testfolder/file1.txt");  
    // 모든 파일 삭제  
    file.delete(Selectors.SELECT_FILES);  
    assertFalse(file.exists());  
    // 파일 생성  
    file.createFile();  
    assertTrue(file.exists());  
}
```

□ 파일접근

– 파일쓰기

```
FileObject file = manager.resolveFile(baseDir, "testfolder/file1.txt");
FileContent fileContent = file.getContent();
// 파일 쓰기
OutputStream os = fileContent.getOutputStream();
String string = "test입니다.";
os.write(string.getBytes());
os.flush();
os.close();

assertNotSame(0, fileContent.getSize());
```

– 파일읽기

```
FileObject writtenFile = manager.resolveFile(baseDir, "testfolder/file1.txt");
FileContent writtenContents = writtenFile.getContent();
InputStream is = writtenContents.getInputStream();
BufferedReader reader = new BufferedReader(new InputStreamReader(is));
StringBuffer sb = new StringBuffer();

for (String line = ""; (line = reader.readLine()) != null; sb.append(line));
is.close();
assertEquals(sb.toString(), string);
```

□ 파일캐싱

– 파일쓰기

```
// 캐싱 Manager 생성
DefaultFileSystemManager fs = new DefaultFileSystemManager(); fs.setFilesCache(manager.getFilesCache());
// zip, jar, tgz, tar, tbz2, file
if (!fs.hasProvider("file")) {
    fs.addProvider("file", new DefaultLocalFileProvider());
}

fs.setCacheStrategy(CacheStrategy.ON_RESOLVE);
fs.init();
// 캐싱 객체 생성
FileObject foBase2 = fs.resolveFile(testFolder);
FileObject cachedFolder = foBase2.resolveFile(scratchFolder.getName().getPath());
// 파일이 존재하지 않음
FileObject[] fos = cachedFolder.getChildren(); assertFalse(contains(fos, "file1.txt"));
// 파일생성
scratchFolder.resolveFile("file1.txt").createFile();
// 파일 존재함
// BUT cachedFolder 에는 파일이 존재하지 않음
fos = cachedFolder.getChildren();
assertFalse(contains(fos, "file1.txt"));
```

❑ Apache Commons VFS

- <https://commons.apache.org/proper/commons-vfs/>

❑ Apache Commons IO

- <https://commons.apache.org/proper/commons-io/>

□ 서비스 개요

- 파일을 클라이언트로부터 서버로 업로드 하거나 서버로부터 클라이언트로 다운로드 할 때 반복적으로 일어나는 Overhead를 피하기 위해 단순한 패턴 및 인터페이스 제공

□ 주요 기능

- HTTP 기반 파일 업로드
- HTTP request(Multipart Request)를 파싱
- 업로드 크기 및 디렉토리 위치에 대한 설정
- 복수개의 파일을 동시에 업로드

□ File Upload/Download 서비스

- 다양한 파일 업로드 API를 제공하는 Commons FileUpload를 채택
- Spring에서는 Commons FileUpload를 사용하여 싱글 파일 업로드에 대하여 가이드함
- 멀티플 파일업로드를 지원하도록 CommonsMultipartResolver를 사용함.
(Spring mvc Multipart Multi file upload 지원부분은 Wiki를 참조)

□ 데이터 전송방식

- GET방식 :
 - URL에 폼데이터가 노출되기 때문에 입력내용의 길이 제한이 있고 256byte~4096byte 까지의 데이터를 전송할 수 있음.
- POST방식 :
 - URL에 노출되지않고 데이터를 전송하기 때문에 입력내용의 길이에 제한을 받지 않음.
- ENCTYPE 속성의 "multipart/form-data" : 파일이나 대용량 데이터를 전송 시 폼 데이터 전송방식 사용

□ File Upload 구현 예제

– Step 1

- 빈 설정 파일에 다음과 같이 CommonsMultipartResolver를 정의함.

```
<!-- MULTIPART RESOLVERS -->
<bean id="spring.RegularCommonsMultipartResolver"
      class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
    <property name="maxUploadSize" value="100000000" />
    <property name="maxInMemorySize" value="100000000" />
</bean>
```

– Step 2

- 컨트롤러의 property로 파일의 업로드 위치를 지정해주고 컨트롤러에서 setter 메소드를 통해 지정된 파일 업로드 위치를 불러올 수 있음

```
# PATH 위치는 설정시 변경

# windows NT일 경우 file.upload.path=C:\\temp
# Unix일 경우 file.upload.path=/usr/file/upload
```

□ File Upload 구현 예제

–Setp 3

- 파일 업로드를 위해 JSP파일의 입력 폼 타입을 file로 지정하고 form의 enctype을 multipart/form-data로 지정함

```
<form method="post" action="<c:url value='/upload/genericMulti.do'/>" enctype="multipart/form-data"> <p>Type: <input type="text" name="type" value="genericFileMulti" size="60" /> </p> <p>File1: <input type="file" name="file[]" size="60" /> </p> <p>File2: <input type="file" name="file[]" size="60" /> </p> <p> <input type="submit" value="Upload" /> </p> </form>
```

–Setp 4

- Controller를 구현

```
@Controller("genericFileUploadController")
public class GenericFileUploadController {

    @Resource(name = "multipartResolver")
    CommonsMultipartResolver multipartResolver;

    @Resource(name = "fileUploadProperties")
    Properties fileUploadProperties;

    @SuppressWarnings("unchecked")
    @RequestMapping(value = "/upload/genericMulti.do")
    public String multipartProcess(final HttpServletRequest request, Model model)
        throws Exception {

        final long startTime = System.nanoTime();

        ... 이하 생략(자세한 설명은 wiki File Upload/Download 참조)
```


□ File Download 구현

–DownloadController 클래스 예시

```
.
.
@Controller("downloadController")
public class DownloadController {

    @Resource(name = "fileUploadProperties")
    Properties fileUploadProperties;

    @RequestMapping(value = "/download/downloadFile.do")
    public void downloadFile(
        @RequestParam(value = "requestedFile") String requestedFile,
        HttpServletResponse response) throws Exception {

        String uploadPath = fileUploadProperties
            .getProperty("file.upload.path");

        File uFile = new File(uploadPath, requestedFile);
        int fSize = (int) uFile.length();

        if (fSize > 0) {

            BufferedInputStream in = new BufferedInputStream(
                new FileInputStream(uFile));
            // String mimetype = servletContext.getMimeType(requestedFile);
            String mimetype = "text/html";

            response.setBufferSize(fSize);
            response.setContentType(mimetype);

        }
    }
}
```

❑ Spring's multipart (fileupload) support

- <https://docs.spring.io/spring-framework/docs/5.3.20/reference/html/web.html#mvc-multipart>

❑ commons.apache.org {fileupload}

- <http://commons.apache.org/fileupload/>

❑ commons.apache.org{empty-parse}

- <http://commons.apache.org/fileupload/faq.html#empty-parse>

□ 서비스 개요

- 네트워크 상에 존재하는 타 어플리케이션과 데이터(파일)을 주고 받기 위해 FTP(File Transfer Protocol) Client 기능을 제공하는 서비스

□ 주요 기능

- FTP Server 연결
 - 파일 주고 받을 FTP Server와 연결
- 파일 송신
 - FTP Server에 파일 송신
- 파일 수신
 - FTP Server에서 파일을 수신

□ FTP 서비스

- Jakarta Commons의 Net에서 지원하는 것은 단순 클라이언트측의 기본적인 Internet Protocol로 구현한 FTP기능을 편리하게 제공
- Jakarta Commons의 Net은 network utility collection

□ Jakarta Commons Net

- Jakarta Commons Net 프로젝트에서 지원하는 프로토콜
 - FTP/FTPS
 - NNTP
 - SMTP
 - POP3
 - Telnet
 - TFTP
 - Finger
 - Whois
 - rexec/rcmd/rlogin
 - Time (rdate) and Daytime
 - Echo
 - Discard
 - NTP/SNTP

□ org.apache.commons.net.ftp의 논리적 흐름

– 논리적 흐름

- ① FTP Client를 생성한다.
- ② FTP Server에 연결한다.
- ③ 응답이 정상적인지 확인한다.
- ④ FTP Server에 로그인한다.
- ⑤ 접속하여 여러가지 작업(list, get, put 등) 을 한다.
- ⑥ FTP Server에서 로그아웃한다.
- ⑦ FTP Server 와의 연결을 끊는다.

□ FTP접속 사용예제 리스트

- 파일리스트 보기
- 파일 다운로드 (get)
- 파일 업로드 (put)
- 파일 업로드 (append)
- 파일 이름변경 (rename)
- 파일 삭제 (delete)
- Directory생성
- OS 커맨드 입력하기
- 파일 및 전송상태 설정

□ FTP접속 사용예제

– 파일리스트 보기

```
FTPFile[] ftpfiles = client.listFiles("/");

if(ftpfiles != null ){
    for (int i = 0; i < ftpfiles.length; i++) {
        FTPFile file = ftpfiles[i];
        logger.info(file.toString()); // 파일정보
        logger.info(file.getName()); // 파일명
        logger.info(file.getSize()); // 파일사이즈
    }
}
```

– 파일 다운로드(GET)

```
File get_file = new File("c:\\temp\\test.jpg");
FileOutputStream outputstream = new FileOutputStream(get_file);
boolean result = client.retrieveFile("/public/test.jpg", outputstream);

outputstream.close();
```

□ FTP접속 사용예제

– 파일 업로드(PUT)

```
File put_file = new File("c:\\temp\\test.jpg");  
InputStream inputStream = new FileInputStream(put_file);  
boolean result = client.storeFile("/public/test.jpg", inputStream);  
  
inputStream.close();
```

– 파일 업로드 (APPEND)

```
File append_file = new File("c:\\temp\\test.jpg");  
InputStream inputStream = new FileInputStream(append_file);  
boolean result = client.appendFile("/public/test.jpg", inputStream);  
  
inputStream.close();
```

– 파일 이름변경(RENAME)

```
boolean result = client.rename("/public/바꾸기전.jpg", "/public/바꾼후.jpg");
```


□ FTP접속 사용예제

– 파일 삭제 (DELETE)

```
boolean result = client.deleteFile("/public/삭제할.jpg");
```

– Directory 생성

```
boolean result = client.makeDirectory("/public/test");
```

□ FTP접속 사용예제

– OS 커맨드 입력하기

```
client.sendCommand(FTPCommand.MAKE_DIRECTORY, "/public/test");
```

– 파일 및 전송상태 설정

```
/* 파일 타입 */  
client.setFileType(FTP.BINARY_FILE_TYPE);  
  
/* 파일 전송 형태 */  
client.setFileTransferMode(FTP.COMPRESSED_TRANSFER_MODE);
```

❑ Jakarta Commons Net

- <http://commons.apache.org/net/>

❑ Commons Net 3.8.0 API

- <https://javadoc.io/doc/commons-net/commons-net/3.8.0/index.html>

□ 서비스 개요

- E-mail을 송신하는 기능을 제공하는 서비스

□ 주요 기능

- E-mail 송신
 - SMTP 표준을 준수하여 E-mail 송신
 - E-Mail 송신시 간단히 텍스트만 보내기
 - E-Mail 송신시 파일 첨부하기
 - E-Mail 송신시 URL을 통해 첨부하기
 - E-Mail 송신시 HTML 이메일 보내기
 - E-Mail 송신시 인증 처리하기

□ Mail 서비스

- 전자정부 프레임워크에서는 Mail 발송을 쉽게 처리하기 위해 Jakarta Commons Email API를 사용
- Commons Email은 내부적으로 Java Mail API와 JavaBeans Activation API 를 제공

□ Mail 서비스의 종류

- 간단히 텍스트만 보내기
- 파일 첨부하기
- URL을 통해 첨부하기
- HTML 이메일 보내기
- 인증 처리하기

□ 간단히 텍스트만 보내기

- org.apache.commons.mail.SimpleEmail 은 가장 중심이 되는 org.apache.commons.mail.Email을 상속받아 setMsg(java.lang.String msg)만을 구현한 가장 기본적인 클래스
- SMTP서버 지정 :
 - setHostName(java.lang.String aHostName)
- 받는 사람의 메일주소 :
 - addTo(java.lang.String email) or addTo(java.lang.String email, java.lang.String name)
- 보내는 사람의 메일주소 :
 - setFrom(java.lang.String email) or setFrom(java.lang.String email, java.lang.String name)
- 여러 사람에게 메일을 보낼 경우 :
 - addTo 함수의 추가

□ 간단히 텍스트만 보내기 (Source)

```
public static void main(String args[]) throws MailException {
    SimpleEmail email = new SimpleEmail();
    // setHostName에 실제 메일서버정보

    email.setCharset("euc-kr"); // 한글 인코딩

    email.setHostName("mail.myserver.com"); //SMTP서버 설정
    try {
        email.addTo("egov@somewhere.org", "egov man"); // 수신자 추가
    } catch (EmailException e) {
        e.printStackTrace();
    }

    try {
        email.setFrom("me@apache.org", "Me"); // 보내는 사람
    } catch (EmailException e) {
        e.printStackTrace();
    }

    email.setSubject("Test message"); // 메일 제목
    email.setContent("simple 메일 Test입니다", "text/plain; charset=euc-kr");
    try {
        email.send();
    } catch (EmailException e) {
        ..
    }
}
```

□ Mail에 파일 첨부하기 (특징)

- org.apache.commons.mail.SimpleEmail 은 가장 중심이 되는 org.apache.commons.mail.Email을 상속받아 setMsg(java.lang.String msg)만을 구현한 가장 기본적인 클래스
- 첨부파일을 보내려면 org.apache.commons.mail.EmailAttachment 클래스와 org.apache.commons.mail.MultiPartEmail 클래스를 사용하면 됨
- 파일 경로와 파일 설명 등을 추가하여 setName(java.lang.String name)을 통해 첨부되는 파일명을 설정한다.
그후 MultiPartEmail 을 통해 SimpleEmail 처럼 기본 메일정보를 설정
- 그리고 MultiPartEmail의 attach() 함수를 통해 첨부 파일을 추가하여 전송
- 만약 첨부파일이 두 개 이상이라면 EmailAttachment 를 여러 개 생성하여 파일 정보를 설정 한 후 attach()를 통해 추가
- EmailAttachment 객체를 생성한 뒤 email.attach()를 사용해서 첨부할 파일을 추가해주기만 하면 된다.
실제 파일명은 한글이 포함되더라도, EmailAttachment.setName() 메소드를 사용해서 파일명을 변경해서 전송할 수도 있다. 주의할 점은 1.0 버전의 Commons Email은 파일명을 한글로 전달할 경우, 파일명이 올바르게 전달되지 않고 깨진다는 점이다.
(파일 자체는 올바르게 전송된다.)
따라서 1.0 버전의 Common Email을 사용하여 파일을 전송할 때에는 알파벳과 숫자로만 구성된 이름의 파일을 전송한다.

□ Mail에 파일 첨부하기(Source)

```
// 첨부할 attachment 정보를 생성합니다
EmailAttachment attachment = new EmailAttachment();
attachment.setPath("C:\\xxxx.jpg");
attachment.setDisposition(EmailAttachment.ATTACHMENT);
attachment.setDescription("첨부 관련 TEST입니다");
attachment.setName("xxxx.jpg"); //

// 기본 메일 정보를 생성합니다
MultiPartEmail email = new MultiPartEmail();
email.setCharset("euc-kr");// 한글 인코딩
email.setHostName("mail.myserver.com");
email.addTo("egov@egov.org", "전자정부");
email.setFrom("egovto@egov.org", "Me");
email.setSubject("전자 정부 첨부 파일 TEST입니다");
email.setMsg("여기는 첨부관련 내용을 입력합니다");

// 생성한 attachment를 추가합니다
email.attach(attachment);

// 메일을 전송합니다
email.send();
```

❑ Apache Commons–Email UserGuide

- <http://commons.apache.org/email/userguide.html>

❑ Commons–Email

- <https://commons.apache.org/proper/commons-email/index.html>

❑ Commons–Email API

- <https://commons.apache.org/proper/commons-email/javadocs/api-release/index.html>

□ 서비스 개요

- 객체를 특정 데이터 형식으로 변환하고, 반대로 특정 데이터 형식으로 작성된 데이터를 객체로 변환하는 기능을 제공하는 서비스
- 메모리 상에 존재하는 객체를 물리적 장치에 저장하거나 네트워크를 통해 전송하기 위해 사용
- Spring Web Service OXM
 - WS는 Server와 Client 두 대상간의 데이터를 주고 받는 기술중에 하나이다. 정보를 요청하는쪽이 Client이다.(Client는 Server가 될수도 있고 일반 사용자가 될수도있다.) 요청한 정보를 받아서 알맞게 처리후 결과값을 리턴하는 쪽이 Server이다.
Client <----- XML -----> Server
 - WS는 XML(WSDL)형식으로 데이터를 주고 받는다.따라서 이 XML를 객체화 하거나 객체를 XML화 해야 한다. 그것이 Marshalling,Unmarshalling이다. OXM Util은 JAXB,Castor,XMLBeans,jiBX,XStream..등 여러 가지가 있다.
Client(OXM) <----- XML(WSDL) -----> (OXM)Server

□ 주요 기능

- Marshalling
 - 객체를 특정 데이터 형식으로 변환
 - 변환된 데이터는 Unmarshalling 기능을 이용하여 원본 객체로 생성 가능해야 함
- Unmarshalling
 - 특정 데이터 형식으로 작성된 정보를 이용하여, 해당하는 객체 생성
 - 생성된 객체는 Marshalling 기능을 이용하여 원본 데이터로 변환 가능해야 함
- Castor와 XMLBeans의 Marshaller, Unmarshaller 사용하기
 - Spring's OXM은 다양한 Java-XML Binding 오픈소스를 지원한다. 여기서는 오픈소스 **XMLBeans**를 사용하여 구현한 가이드프로그램을 제시한다.
 - XMLBeans Configuration

```
<bean id="divertxmlbeans" class="org.egovframe.rte.fdl.divert.DivertXMLBeans">
  <property name="marshaller" ref="xmlBeansMarshaller" />
  <property name="unmarshaller" ref="xmlBeansMarshaller" />
</bean>
<bean id="xmlBeansMarshaller" class="org.springframework.oxm.xmlbeans.XmlBeansMarshaller" />
```

- XMLBeans Guide Program(Java Object의 데이터를 XML문서로 DataBinding Sample Source)

```
@Resource(name = "xmlBeansMarshaller")
private Marshaller marshaller;
@Test
public void testMarshalling()
{
    FileOutputStream os = null;
    userDoc = UserinfoDocument.Factory.newInstance(); userElement = userDoc.addNewUserinfo();
    userElement.setName("홍길동"); userElement.setAge(31);
    userElement.setPhone(022770918); xmlOptions = new XmlOptions();
    xmlOptions.setSavePrettyPrint(); xmlOptions.setSavePrettyPrintIndent(4);
    xmlOptions.setCharacterEncoding("euc-kr");
    try {
        os = new FileOutputStream("XMLBeanGen.xml");
        marshaller.marshal(userDoc, new StreamResult(os));
    } catch (Exception ee)
    {
        ...
    }
}
```

- XMLBeans Guide Program(XML문서를 JavaObject로 DataBinding Sample Source)

```
@Resource(name = "xmlBeansMarshaller")
private Unmarshaller unmarshaller;
@Test
public void testUnmarshalling() {
    FileInputStream is = null;
    try {
        is = new FileInputStream("XMLBeanGen.xml");
        userDoc = (UserinfoDocument) unmarshaller.unmarshal(new StreamSource(is));
        userElement = userDoc.getUserinfo();
    } catch (FileNotFoundException fnfe) {
        fnfe.printStackTrace();
        fail("testUnmarshalling failed!");
    }
    catch (IOException ioe) {
        ioe.printStackTrace();
    }
    ...
}
```

❑ Spring Framework OXM class API

- <https://docs.spring.io/spring-framework/docs/5.3.20/javadoc-api/org/springframework/oxm/package-summary.html>

❑ XMLBeans

- <http://xmlbeans.apache.org/>

□ 서비스 개요

- 객체에 대한 Pooling기능을 제공하는 서비스이다. 객체의 생성 비용이 크고, 생성 횟수가 많으며, 평균적으로 사용되는 객체의 수가 적은 경우, 성능을 향상시키기 위해서 사용한다.

Object Pool은 소프트웨어 디자인 패턴으로서, 객체를 필요에 따라 생성하고 파괴하는 방식이 아닌, 적절한 개수의 객체를 미리 사용 가능한 상태로 생성하여 이를 이용하는 방식이다. Client는 Pool에 객체를 요청하여 객체를 얻은 후, 업무를 수행한다. 얻어온 객체를 이용하여 업무 수행을 끝마친 후, 객체를 파괴하는 것이 아니라 Pool에게 돌려주어 다른 Client가 사용할 수 있도록 한다. Object Pooling은 객체 생성 비용이 크고, 객체 생성 횟수가 많으며, 평균적으로 사용되는 객체의 수가 적은 경우, 높은 성능의 향상을 가져다 준다.

□ 주요 기능

- Pool 초기화(Initialization)
 - 특정 객체에 대한 Pool 생성
 - Pool 초기값 및 최대값을 설정하면, 초기값 개수만큼 객체를 생성하여 Pool에 추가함
- 객체 할당(Allocate)
 - Pool에 사용 가능한 객체가 있을 경우, 객체를 할당하고 사용 가능한 객체가 없을 경우, 만약 Pool 크기가 최대값 보다 작으면 새로운 객체를 생성하여 할당하고, Pool의 크기가 최대값인 경우, 예외 처리를 수행함

- 객체 반환(Release)
 - 사용이 완료된 객체를 Pool로 반환하고, Pool로 반환된 객체는 재사용 할 수 있도록 초기화함
- ObjectPool Guide Program
 - Configuration

```
<bean id="dbcpObjectpool" class="org.egovframe.rte.fdl.objectpool.dbcp.dbcpObjectpoolset">  
  <property name="pool" ref="dbcpObjectPooler" />  
</bean>  
<bean id="dbcpObjectPooler" class="org.egovframe.rte.fdl.objectpool.dbcp.DbcpObjectpool" />
```

– ObjectPool 설정 필드

| 설정 | 설명 |
|-------------------------------|---|
| maxActive | 커넥션 풀이 제공할 최대 커넥션 개수 |
| whenExhaustedAction | 커넥션 풀에서 가져올 수 있는 커넥션이 없을 때 어떻게 동작할지를 지정한다. |
| maxWait | 사용되지 않고 풀에 저장될 수 있는 최대 커넥션 개수. 음수 일 경우 제한이 없다. |
| maxIdle | 사용되지 않고 풀에 저장될 수 있는 최소 커넥션 개수. |
| testOnBorrow | True일 경우 커넥션 풀에서 커넥션을 가져올 때 커넥션이 유효한지의 여부를 검사한다. |
| testOnReturn | True일 경우 커넥션 풀에서 커넥션을 반환할 때 커넥션이 유효한지의 여부를 검사한다. |
| timeBetweenEvictionRunsMillis | 사용되지 않는 커넥션을 추출하는 쓰레드의 실행 주기를 지정한다. |
| numTestsPerEvictionRun | 사용되지 않는 커넥션을 몇 개 검사할지 지정한다. |
| minEvictableIdleTimeMillis | 사용되지 않는 커넥션을 추출할 때 이 속성에서 지정한 시간 이상 비활성화 상태인 커넥션만 추출한다. |
| testWhileIdle | True일 경우 비활성화 커넥션을 추출할 때 커넥션이 유효한지의 여부를 검사해서 유효하지 않은 커넥션을 풀에서 제거한다. |

– Guide Program

- jdbc_driver,username,password를 jdbc.properties로 부터 읽어온다.

```
private void loadProperties() throws IOException
{
    String fileName_ = Thread.currentThread().getContextClassLoader().getResource("jdbc.properties").getFile();
    Properties props = null;
    FileInputStream fis = null;
    props = new Properties();
    fis = new FileInputStream(fileName_);
    props.load(new BufferedInputStream(fis));
    fis.close();
    g_Driver = props.getProperty("driver");
    g_URL = props.getProperty("dburl");
    g_User = props.getProperty("username");
    g_Password = props.getProperty("password");
}
```

– Guide Program

- jdbc.properties를 통해서 데이터베이스 접속정보를 얻어온다.

```
private void setupDriver(String driver,String url,String user,String password)throws Exception
{
    Class.forName(driver);
    GenericObjectPool connectionPool = new GenericObjectPool(null);
    connectionPool.setMaxActive(maxActive);
    .....
    connectionPool.setTestWhileIdle(testWhileIdle);
    connectionPool.setNumTestsPerEvictionRun(numTestsPerEvictionRun);
    ConnectionFactory connectionFactory = new DriverManagerConnectionFactory(url,user,password);
    PoolableConnectionFactory poolableConnectionFactory =
        new PoolableConnectionFactory(connectionFactory, connectionPool, null, null, false, true);
    PoolingDriver poolingDriver = new PoolingDriver();
    poolingDriver.registerPool("database", connectionPool);
}
```

ObjectPool 설정 세팅

실제 DB와의 커넥션을
연결해주는 팩토리 생성

Connection Pool이 PoolableConnection 객체를 생성할
때 사용할 PoolableConnectionFactory 생성

– Guide Program

- DbcpObjectpoolTest.java은 가이드프로그램을 이용한 테스트 프로그램이다.

```
@Resource(name = "dbcpObjectPooler")
private DbcpObjectpool pool;
@Test
public void runModule() {
    Connection con = null;
    PreparedStatement pstmt = null;
    ResultSet rs = null;
    String sql = "select * from emp";
    try {
        con = pool.getConnection(); pstmt = con.prepareStatement(sql);
        rs = pstmt.executeQuery(); logger.debug("module1");
        while(rs.next()){
            logger.debug(rs.getString("ename")); }
    }
    ...
}
```

❑ Apache Commons Pool class API

- <http://commons.apache.org/proper/commons-pool/>

□ 서비스 개요

- 외부 파일이나 환경 정보에 구성되어 있는 key, value의 쌍을 내부적으로 가지고 있으며, 어플리케이션이 이 특정 key에 대한 value에 접근할 수 있도록 해주는 서비스
- 주로 시스템의 설치 환경에 관련된 정보나, 잦은 정보의 변경이 요구되는 경우 외부에서 그 정보를 관리하게 함으로써 시스템의 가변성을 향상시킴

□ 주요 기능

- 외부 별도의 파일에 설정 정보를 저장
- 외부 파일은 절대/상대 경로나 클래스 패스로 접근하여 사용
- 다양한 Character Set을 지원
- 서버의 재기동 없이 프로퍼티를 적용
- 문자열 key에 모든 객체를 value에 저장
- Key 문자열을 가지고 value 객체 조회
- 전체 key-value 쌍을 가져오는 기능 제공

□ Property

- 시스템의 설치 환경에 관련된 정보나, 잦은 정보의 변경이 요구되는 경우 외부에서 그 정보를 관리하게 함
- 시스템의 유연성을 높일 수 있음
- Spring Bean 설정 파일에 관리하고자 하는 정보를 입력(Beans 설정 파일 사용) 하거나 외부 파일에 정보 입력 후에 Beans 설정 파일에서 그 파일 위치를 입력하여 이용

□ Bean 설정 파일 사용(1/2)

– Configuration

```
<bean name="propertyService"
      class="org.egovframe.rte.fdl.property.impl.EgovPropertyServiceImpl"
      destroy-method="destroy">
  <property name="properties">
    <map>
      <entry key="AAAA" value="1234"/>
    </map>
  </property>
</bean>
```

– Sample Source

```
@Resource(name="propertyService")
protected EgovPropertyService propertyService ;

@Test
public void testPropertiesService() throws Exception {
    assertEquals("1234",propertyService.getString("AAAA"));
}
```

□ Bean 설정 파일 사용(2/2)

– 제공 유형별 설정/사용방법

| 제공유형 | 설정 방법 | 사용 방법 |
|---------|-------------------------|---------------------------------|
| String | key="A" value="ABC" | propertyService.getString("A") |
| boolean | key="B" value="true" | propertyService.getBoolean("B") |
| int | key="C" value="123" | propertyService.getInt("C") |
| long | key="D" value="123" | propertyService.getLong("D") |
| short | key="E" value="123" | propertyService.getShort("E") |
| float | key="F" value="123" | propertyService.getFloat("F") |
| Vector | key="G" value="123,456" | propertyService.getVector("G") |

□ 외부 설정 파일 사용(1/2)

– Configuration

```
<bean name="propertyService"          class="org.egovframe.rte.fdl.property.impl.EgovPropertyServiceImpl"
    destroy-method="destroy">
    <property name="extFileName">
        <set>
            <map>
                <entry key="encoding" value="UTF-8"/>
                <entry key="filename" value="file:./src/**/*.refresh-resource.properties"/>
            </map>
            <value>classpath*:properties/resource.properties</value>
        </set>
    </property>
</bean>
```

– properties

```
AAAA=1234
```

□ 외부 설정 파일 사용(2/2)

– Sample Source

```
@Resource(name="propertyService")
protected EgovPropertyService propertyService ;

@Test
public void testPropertiesService() throws Exception {
    assertEquals("1234",propertyService.getString("AAAA"));
}
```

– 실시간 갱신방법

- 외부파일에 기재된 property 내용 수정
- propertyService.refreshPropertyFiles() 호출

❑ Property-placeholder

– Configuration

```
<context:property-placeholder location="com/bank/config/datasource.properties"/>

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
  <property name="driverClass" value="${database.driver}"/>
  <property name="jdbcUrl" value="${database.url}"/>
  <property name="username" value="${database.username}"/>
  <property name="password" value="${database.password}"/>
</bean>
```

□ DB PropertySource(1/2)

– DB 설정

```
CREATE TABLE PROPERTY (  
  PKEY VARCHAR(20) NOT NULL PRIMARY KEY ,  
  PVALUE VARCHAR(20) NOT NULL  
);  
  
INSERT INTO PROPERTY (PKEY, PVALUE) VALUES ('egov.test.sample01', 'db-property-sample01');  
INSERT INTO PROPERTY (PKEY, PVALUE) VALUES ('egov.test.sample02', 'db-property-sample02');  
  
...  
  
commit;
```

– DB 관련 XML 설정

```
<jdbc:embedded-database id="dataSource" type="HSQL">  
  <jdbc:script location="classpath:db/ddl.sql" />  
  <jdbc:script location="classpath:db/dml.sql" />  
</jdbc:embedded-database>  
  
<bean id="dbPropertySource" class="org.egovframe.rte.fdl.property.db.DbPropertySource">  
  <constructor-arg value="dbPropertySource"/>  
  <constructor-arg ref="dataSource"/>  
  <constructor-arg value="SELECT PKEY, PVALUE FROM PROPERTY"/>  
</bean>
```

❑ DB PropertySource(2/2)

– web.xml 설정

```
<context-param>
  <param-name>contextInitializerClasses</param-name>
  <param-value>org.egovframe.rte.fdl.property.db.initializer.DBPropertySourceInitializer</param-value>
</context-param>
<context-param>
  <param-name>propertySourceConfigLocation</param-name>
  <param-value>classpath:/initial/property-source-context.xml</param-value>
</context-param>
```

– 사용 예

```
...
<context:property-placeholder/>

<!-- 메시지소스빈 설정 -->
<bean id="propertyTest" class="egov.sample.property.PropertyTest">
  <property name="sample01" value="${egov.test.sample01}"/>
  <property name="sample02" value="${egov.test.sample02}"/>
</bean>
...
```

□ 서비스 개요

- 국제화(Internationalization) 및 현지화(Localization) 등을 지원하기 위하여, key값을 이용하여, 각 국가 및 언어에 해당하는 메시지를 읽어오는 서비스

□ 주요 기능

- 메시지 참조
 - 국가 및 언어에 따라 미리 저장되어 있는 메시지 정보에서, 특정 key 값에 해당하는 메시지를 조회함

❑ Spring Message Source

- Spring은 MessageSource 인터페이스를 확장하여 어플리케이션에서 사용하는 메시지에 대한 일관된 처리 및 국제화를 지원함

❑ Message 맛보기 예제(1/2)

- 환경설정
 - MessageSource를 구현한 bean을 설정
 - 아래 설정에서 "egovframework-message" 로 지정한 파일은 실제로는 egovframework-message.properties 로 정의되어 있음

```
<bean name="messageSource"
  class="org.springframework.context.support.ResourceBundleMessageSource">
  <property name="useCodeAsDefaultMessage">
    <value>true</value>
  </property>
  <property name="basenames">
    <list>
      <value>egovframework-message</value>
    </list>
  </property>
</bean>
```

❑ Message 맛보기 예제 (2/2)

- 메시지 사용 예제
 - messageSource.getMessage() 메소드를 이용하여 Message를 얻음

```
//egovframework-message.properties에 정의된 메시지 내용.  
resource.basic.msg1=message1  
  
@Resource(name="messageSource")  
MessageSource messageSource ;  
  
String getMsg = messageSource.getMessage("resource.basic.msg1" , null , Locale.getDefault() );  
assertEquals("Get Message Success!", getMsg , "message1");
```

❑ Message Locale 사용(1/2)

- 동일한 메시지 키를 가지고 언어별로 별도로 설정 관리하여 사용자에게 따라서 사용자에게 맞는 언어로 메시지를 제공할 수 있음
- 환경 설정
 - "egovframework-message-locale" 로 지정한 파일을 egovframework-message-locale_ko.properties, egovframework-message-locale_en.properties로 정의하고 동일한 메시지 키에 해당하는 메시지를 달리 지정함

```
<bean name="messageSource"
  class="org.springframework.context.support.ResourceBundleMessageSource">
  <property name="useCodeAsDefaultMessage">
    <value>true</value>
  </property>
  <property name="basenames">
    <list>
      <value>egovframework-message-locale
      </value>
    </list>
  </property>
</bean>
```

❑ Message Locale 사용(2/2)

– Properties File 설정

```
//egovframework-message-locale_ko.properties 파일 내용  
resource.locale.msg1=메시지1
```

```
//egovframework-message-locale_en.properties 파일 내용  
resource.locale.msg1=en_message1
```

– 사용 예제

- Locale정보에 따라서 추출되는 메시지의 내용이 달라짐

```
//egovframework-message.properties에 정의된 메시지 내용.  
resource.basic.msg1=message1
```

```
String getMsg = messageSource.getMessage("resource.locale.msg1", null, Locale.KOREAN );  
assertEquals("Get Message Success!", getMsg, "메시지1");
```

```
String getMsg = messageSource.getMessage("resource.locale.msg1", null, Locale.ENGLISH );  
assertEquals("Get Message Success!", getMsg, "en_message1");
```

❑ Message Parameter

- 프로그램 수행 중에 발생하는 메시지에 파라미터를 추가하여 제공
 - 메시지 정의 시 {0},{1} 등으로 파라미터 선언

```
resource.basic.msg3=message {0} {1}
```

- 사용 예제
 - parameter에 1과 2를 지정하여 getMessage의 두 번째 인자에 넣고 호출하면 리턴 메시지로 “message 1 2”를 얻음

```
Object[] parameter = { new String("1"), new Integer(2) };
```

```
String getMsg = messageSource.getMessage("resource.basic.msg3", parameter, Locale.getDefault());  
assertEquals("Get Message Success!", getMsg, "message 1 2");
```

❑ The Spring Framework – Reference Documentation

- <https://docs.spring.io/spring-framework/docs/5.3.20/reference/html/core.html#context-functionality-messagesource>

□ 서비스 개요

- 어플리케이션 서버 내에서 주기적이거나 반복적인 작업을 지원하는 서비스

□ 주요 기능

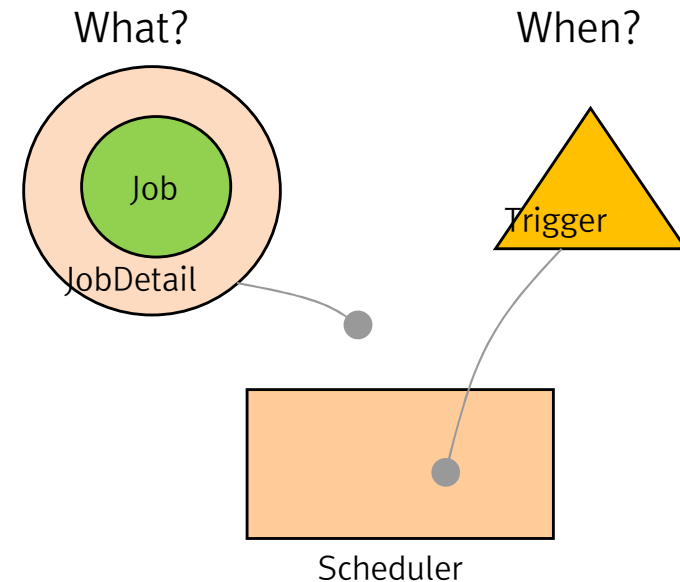
- 속성(반복 주기, 반복 회수 등)을 지정 및 변경하는 기능을 제공
- 주기적이거나 반복적으로 수행될 작업을 선언하여 관리
- 주기적이거나 반복적으로 수행될 작업들의 주기, 반복 회수 등을 선언하여 관리
- Unix의 Cron command와 유사한 형태의 표기법 지원

□ Quartz 스케줄러

- 오픈 소스 작업 스케줄링 프레임워크
- 자바로 구현되어 있으며 독립 어플리케이션 혹은 J2EE환경에서 통합되어 실행됨
- 다양한 작업 스케줄링 지원
 - Unix 시스템의 cron과 유사한 표현식 지원
- 확장 가능한 리스너 제공
- 오류 시 작업 복구
- POJO 혹은 EJB 작업 지원
- JTA 트랜잭션 지원
- 클러스터링 지원
- 멀티쓰레딩 지원
- RMI를 통한 리모트 사용지원

□ Quartz 스케줄러 주요 요소

- Scheduler
 - Quartz 실행 환경을 관리하는 핵심 개체
- Job
 - 사용자가 수행할 작업을 정의하는 인터페이스로서 Trigger 개체를 이용하여 스케줄링 됨.
- JobDetail
 - 작업명과 작업그룹과 같은 수행할 Job에 대한 상세 정보를 정의하는 개체
- Trigger
 - 정의한 Job 개체의 실행 스케줄을 정의하는 개체로서 Scheduler 개체에게 Job 수행시점을 알려주는 개체



□ Quartz 스케줄러 특징

- 수행 작업을 정의하는 Job과 실행 스케줄을 정의하는 Trigger를 분리함으로써 유연성을 제공함
 - Job 과 실행 스케줄을 정의한 경우, Job은 그대로 두고 실행 스케줄만을 변경할 수 있음
 - 하나의 Job에 여러 개의 실행 스케줄을 정의할 수 있음

□ Quartz 스케줄러 사용 예제 (1/2)

– 사용자 정의 Job

- 사용자는 업무처리를 수행 할 Job 개체를 생성하기 위해 org.quartz.Job 인터페이스를 구현하고 심각한 오류가 발생한 경우 JobExecutionException 예외를 던질 수 있음
- Job 인터페이스는 단일 메소드로 execute()가 정의되어 있음

```
public class DumbJob implements Job {  
    public void execute(JobExecutionContext context)  
        throws JobExecutionException {  
        System.out.println("DumbJob is executing.");  
    }  
}
```

□ Quartz 스케줄러 사용 예제 (2/2)

– Quartz 스케줄 등록

- ① 우선 Job 설정을 위해 JobDetail 클래스를 정의한다.
- ② TriggerUtils를 이용하여 매일 8시30분 실행하는 Trigger를 생성한다.
- ③ 마지막으로, Scheduler에 JobDetail과 Trigger를 등록한다.

```
JobDetail jobDetail =  
    new JobDetail("myJob", // Job 명  
        sched.DEFAULT_GROUP, // Job 그룹명('null' 값인 경우 DEFAULT_GROUP 으로 정의됨)  
        DumbJob.class); // 실행할 Job 클래스  
  
Trigger trigger = TriggerUtils.makeDailyTrigger(8, 30); // 매일 08시 30분 실행  
trigger.setStartTime(new Date()); // 즉시 시작  
trigger.setName("myTrigger");  
  
sched.scheduleJob(jobDetail, trigger);
```

□ Spring 과 Quartz 통합

- Spring은 Scheduling 지원을 위한 통합 클래스를 제공함

□ Spring의 Quartz 작업 생성 방법

- JobDetailBean을 이용한 방법으로, QuartzJobBean을 상속받아 Job 클래스 생성
- MethodInvokingJobDetailFactoryBean을 이용하여 Bean 객체의 메소드 직접 호출

□ JobDetailBean을 이용한 작업 생성(1/2)

– Job 작성

- QuartzJobBean은 Quartz Job 인터페이스 구현체
- 아래 예제의 SayHelloJob 클래스는 작업 생성을 위해 QuartzJobBean의 executeInternal(..) 함수를 오버라이드함

```
package org.egovframe.rte.fdl.scheduling.sample;

public class SayHelloJob extends QuartzJobBean {

    private String name;
    public void setName(String name) {
        this.name = name;
    }
    @Override
    protected void executeInternal(JobExecutionContext ctx)
        throws JobExecutionException {
        System.out.println("Hello, " + name);
    }
}
```

□ JobDetailBean을 이용한 작업 생성(2/2)

– JobDetailBean 설정

- JobDetail 작업 실행에 필요한 정보를 담고 있는 객체
- Spring은 JobDetail 빈 생성을 위해 JobDetailBean을 제공함
- jobDataAsMap 개체를 이용하여 JobDetail 개체에 Job 설정에 필요한 속성 정보를 전달함

```
<bean id="jobDetailBean" class="org.springframework.scheduling.quartz.JobDetailBean">
  <property name="jobClass"
    value="org.egovframe.rte.fdl.scheduling.sample.SayHelloJob" />
  <property name="jobDataAsMap">
    <map>
      <entry key="name" value="JobDetail" />
    </map>
  </property>
</bean>
```

❑ MethodInvokingJobDetailFactoryBean을 이용한 작업 생성

- 작업 수행을 할 Bean 클래스 정의

```
package org.egovframe.rte.fdl.scheduling.sample;

public class SayHelloService {

    public void sayHello() {
        System.out.println("Hello");
    }
}
```

- 정의한 Bean 객체의 메소드를 직접 호출하는 작업을 생성하기 위해 MethodInvokingJobDetailFactoryBean 정의

```
<bean id="sayHelloService" class="org.egovframe.rte.fdl.scheduling.sample.SayHelloService">
    <property name="name" value="FactoryBean" />
</bean>

<bean id="jobDetailFactoryBean"
    class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">
    <property name="targetObject" ref="sayHelloService" />
    <property name="targetMethod" value="sayHello" />
    <property name="concurrent" value="false" />
</bean>
```

□ 작업 스케줄링(1/2)

- SimpleTrigger
 - 특정 시간, 반복 회수, 대기 시간과 같은 단순 스케줄링에 사용됨
- CronTrigger
 - 유닉스의 Cron 명령어와 유사하며, 복잡한 스케줄링에 사용됨
 - CronTrigger 는 달력을 이용하듯 특정 시간, 요일, 월에 Job 을 수행하도록 설정할 수 있음
- SimpleTriggerBean을 이용한 설정 예

```
<bean id="simpleTrigger" class="org.springframework.scheduling.quartz.SimpleTriggerBean">
  <property name="jobDetail" ref="jobDetailBean" />
  <!-- 즉시 시작 -->
  <property name="startDelay" value="0" />
  <!-- 매 10초마다 실행 -->
  <property name="repeatInterval" value="10000" />
</bean>
```


□ 작업 스케줄링(2/2)

- CronTriggerBean을 이용한 설정

```
<bean id="cronTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean">
  <property name="jobDetail" ref="jobDetailFactoryBean" />
  <!-- 매 10초마다 실행 -->
  <property name="cronExpression" value="*/10 * * * * ?" />
</bean>
```

- 크론 표현식

| | | | | | | |
|-------------|-------------|--------------|--------------|-------------|-------------|------------|
| 초 (0-59) | 분 (0-59) | 시간 (0-23) | 날짜 (1-31) | 월 (1-12) | 요일 (1-7) | 년도 (옵션) |
|-------------|-------------|--------------|--------------|-------------|-------------|------------|

- 특수문자
 1. “*” : 항상 실행의 의미
 2. “?” : 날짜와 요일은 상호 배타적 이므로 둘 중 하나를 설정하지 않음을 표시함
- 사용 예제
 1. 0 15 10 ? * * 매일 10시 15분에 실행
 2. 0 15 10 15 * ? 매월 15일 10시15분에 실행
- 크론 표현식에 대한 자세한 설명은 아래 사이트 참조
https://en.wikipedia.org/wiki/Cron#CRON_expression

□ 작업 시작하기

- 스케줄링한 작업의 시작을 위해 Spring 은 SchedulerFactoryBean을 제공함

```
<bean id="scheduler" class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
  <property name="triggers">
    <list>
      <ref bean="simpleTrigger" />
      <ref bean="cronTrigger" />
    </list>
  </property>
</bean>
```

❑ Quartz 매뉴얼

- <http://www.quartz-scheduler.org/documentation/>

❑ Spring Scheduling 매뉴얼

- <https://docs.spring.io/spring-framework/docs/5.3.20/reference/html/integration.html#scheduling>

❑ Quartz Cron 표현식

- <http://www.quartz-scheduler.org/documentation/quartz-2.4.0/tutorials/tutorial-lesson-06.html>

□ 서비스 개요

- 응용프로그램 작성시 발생할 수 있는 보안상의 문제점은 인증 및 접근제어, 계정관리, 정보통제 및 환경설정 등의 상황에서 발생할 수 있으며 이에 대한 대응을 위한 서비스 제공
- 기존의 Servlet spec에서 Security의 한계
 - container 이식성, config. Requirements, 제한된 웹 요청 security의 한계
 - 서비스 레이어 보안 및 도메인 객체 레벨 보안 미 지원

□ 주요 기능

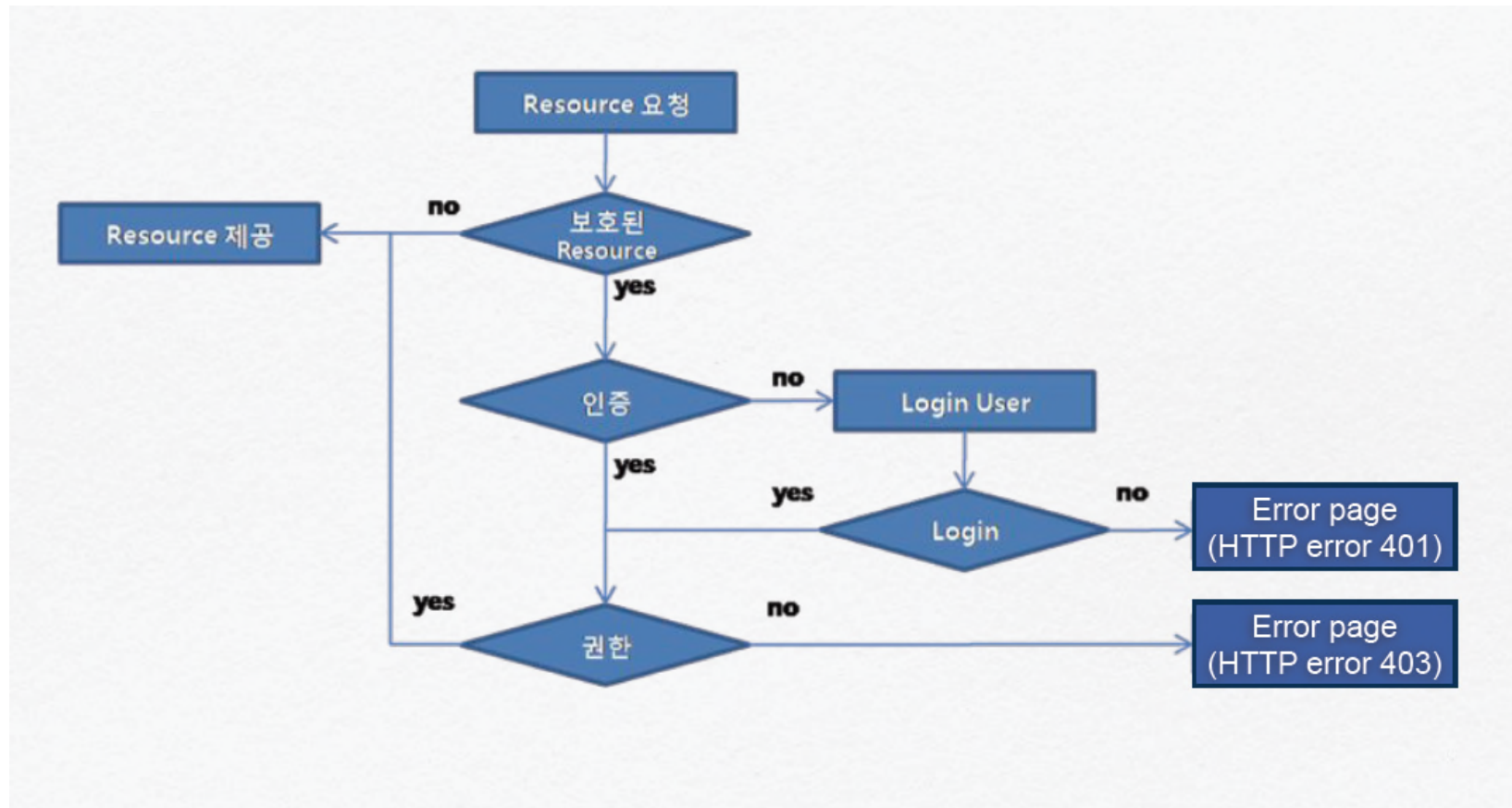
- 인증
- Web URL 페이지 인증
- Method invocation 인증
- 권한부여
- 세션관리
- 설정 간소화

□ Why Spring Security?

- 엔터프라이즈 어플리케이션을 위한 인증(Authentication), 권한 처리(Authorization) 서비스를 제공하는 강력하고 유연한 보안 솔루션
- Servlet Filter 와 Java AOP 를 통한 Interception를 사용하여 보안을 강제하며 Spring의 IoC 와 lifecycle 서비스 기반으로 동작
- authentication, Web URL authorization, Method 호출 authorization, 도메인 객체 기반의 security 처리, 채널 보안(https 강제), Human user 인식 등의 주요 기능을 제공
- Web request 보안에 더하여 Service Layer 및 인스턴스 수준의 보안 제공으로 Layering issue 해결 및 웹 클라이언트 외의 다양한 rich 클라이언트 / 웹 서비스에 대한 보안 제어를 지원
- 재사용성, 이식성, 코드 품질, 레퍼런스 (정부,은행,대학,기업 등 많은 business field), 다양한 타 프레임워크 지원, community
- Spring Security는 편의성 뿐 아니라 보안 레이어 컴포넌트의 재사용성이 높은 오픈 소스 보안 프레임워크

□ Architecture

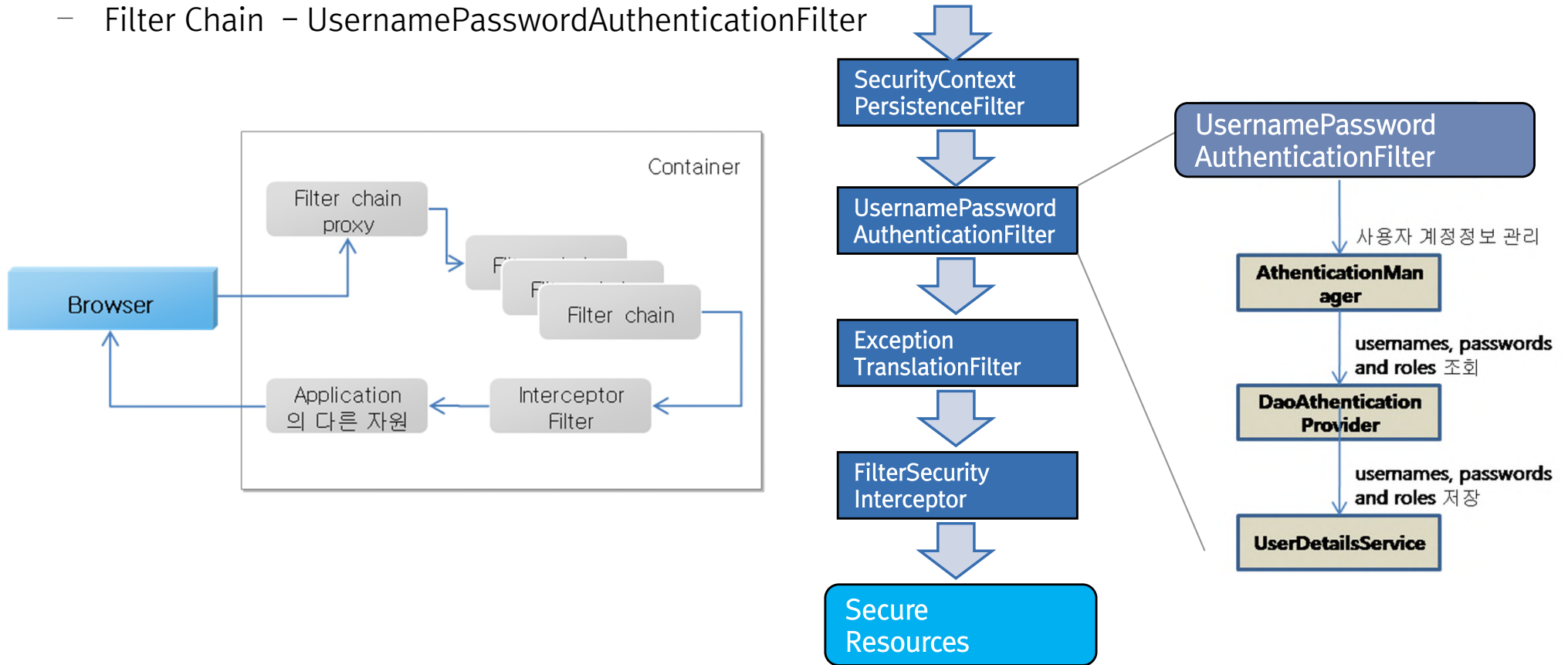
- 일반적인 웹어플리케이션의 인증절차



전형적인 웹 리소스 접근 개념도 [그림]

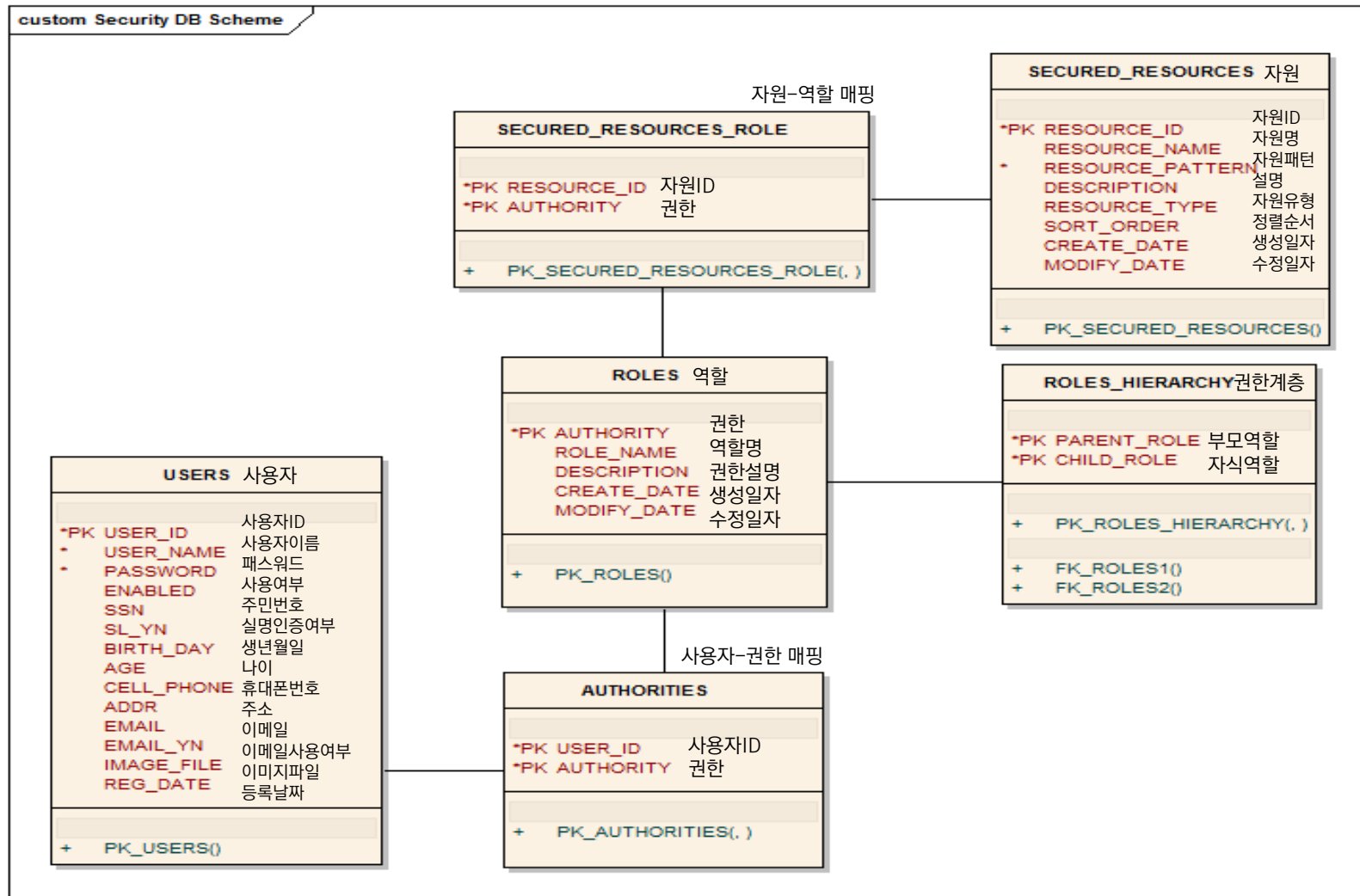
□ Architecture

- HTTP Request
- Filter Chain – UsernamePasswordAuthenticationFilter



□ Architecture

– DB Schema(1/4)



□ Architecture

– DB Schema(2/4)

- SECURED_RESOURCES(보호된 자원)

| RESOURCE_ID | RESOURCE_PATTERN |
|-------------|---|
| web-000001 | \A/sample\.do.*\Z |
| web-000002 | \A/.*\.do.*\Z |
| web-000003 | \A/.*\Z |
| web-000004 | \A/reloadAuthMapping\.do.*\Z |
| mttd-000001 | egovframework.sample.service.EgovSampleService.updateSample |
| mttd-000002 | egovframework.sample.service.EgovSampleService.deleteSample |
| mttd-000003 | execution(* egovframework.sample..service.*Service.insert*(..)) |

□ Architecture

– DB Schema(3/4)

- ROLES(역할)

AUTHORITY

IS_AUTHENTICATED_ANONYMOUSLY
 IS_AUTHENTICATED_REMEMBERED
 IS_AUTHENTICATED_FULLY
 ROLE_RESTRICTED
 ROLE_USER
 ROLE_ADMIN
 ROLE_A
 ROLE_B

DESCRIPTION

익명 사용자
 REMEMBERED 사용자
 인증된 사용자
 제한된 사용자
 일반 사용자
 관리자
 A 업무
 B 업무

□ Architecture

– DB Schema(4/4)

- ROLES_HIERARCHY(역할 계층)

CHILD_ROLE

ROLE_ADMIN
 ROLE_USER
 ROLE_RESTRICTED
 IS_AUTHENTICATED_FULLY
 IS_AUTHENTICATED_REMEMBERED
 ROLE_ADMIN
 ROLE_ADMIN
 ROLE_A
 ROLE_B

PARENT_ROLE

ROLE_USER
 ROLE_RESTRICTED
 IS_AUTHENTICATED_FULLY
 IS_AUTHENTICATED_REMEMBERED
 IS_AUTHENTICATED_ANONYMOUSLY
 ROLE_A
 ROLE_B
 ROLE_RESTRICTED
 ROLE_RESTRICTED

□ Authentication(인증) – 개요

- 허락된 사용자에게만 공개되는 콘텐츠(정보 또는 기능)에 접근하기 위하여 사용자의 아이디와 암호를 입력하여 로그인하는 과정
- 특정 사용자가 유효한 사용자인지를 판단하는 과정
- 종류
 - Common
 - DAO Authentication Provider
 - Form, BASIC, Digest, Anonymous, Remember-Me, JAAS
 - LDAP, CAS, X509, SiteMider
 - Container Adapter, EJB Integration

□ Authentication(인증) – 기본환경

– <http>

```
<http auto-config='true'>
  <intercept-url pattern="/*" access="ROLE_USER"/>
</http>
```

• auto-config ?

```
<http>
  <intercept-url pattern="/*" access="ROLE_USER" />
  <form-login />
  <anonymous />
  <http-basic />
  <logout />
  <remember-me />
</http>
```

– Authentication Provider

```
<authentication-manager>
  <authentication-provider>
    <user-service>
      <user name="jimi" password="jimispasword" authorities="ROLE_USER, ROLE_ADMIN"/>
      <user name="bob" password="bobspassword" authorities="ROLE_USER"/>
    </user-service>
  </authentication-provider>
</authentication-manager>
```

□ Authentication(인증) – 기본환경 예제(1/2)

– <http>

```
<http access-denied-page="/system/accessDenied.do" request-matcher="regex">
  <form-login
    login-processing-url="/login"
    authentication-failure-url="/cvpl/EgovCvplLogin.do?login_error=1"
    default-target-url="/index.jsp?flag=L"
    login-page="/cvpl/EgovCvplLogin.do" />
  <logout logout-success-url="/cvpl/EgovCvplLogin.do" />
</http>
```

– Authentication Provider

```
<authentication-manager>
  <authentication-provider user-service-ref="jdbcUserService">
    <password-encoder hash="md5" base64="true" />
  </authentication-provider>
</authentication-manager>
```

❑ Authentication(인증) – 기본환경 예제(2/2)

- jdbcUserService (사용자정보 조회)

```
<jdbc-user-service id="jdbcUserService" data-source-ref="dataSource"
  users-by-username-query="SELECT USER_ID,PASSWORD,ENABLED,BIRTH_DAY FROM USERS WHERE USER_ID = ?"
  authorities-by-username-query="SELECT USER_ID,AUTHORITY FROM AUTHORITIES WHERE USER_ID = ?"/>
```

- Sample Source (JSP)

```
<c:url value="/login" var="loginUrl">
<form action="${loginUrl}" method="post">
  <p>
    <label for="username">Username</label>
    <input type="text" id="username" name="username"/>
  </p>
  <p>
    <label for="password">Password</label>
    <input type="password" id="password" name="password"/>
  </p>
</form>
```

□ Authentication(인증) – 세션(1/3)

- JdbcUserDetailsManager 를 확장한 EgovJdbcUserDetailsManager 클래스를 구현
- 사용자의 기본 username, password, enabled 필드 외에 다른 정보를 추가하여 세션정보를 관리

```
<b:bean id="jdbcUserService"  
    class="org.egovframe.rte.fdl.security.userdetails.jdbc.EgovJdbcUserDetailsManager">  
    <b:property name="usersByUsernameQuery"  
        value="SELECT USER_ID,PASSWORD,ENABLED,USER_NAME,BIRTH_DAY,SSN FROM USERS WHERE USER_ID = ? ">  
    <b:property name="authoritiesByUsernameQuery"  
        value="SELECT USER_ID,AUTHORITY FROM AUTHORITIES WHERE USER_ID = ? ">  
    <b:property name="roleHierarchy" ref="roleHierarchy">  
    <b:property name="dataSource" ref="dataSource">  
    <b:property name="mapClass"  
        value="org.egovframe.rte.fdl.security.userdetails.EgovUserDetailsMapping">  
</b:bean>
```


□ Authentication(인증) – 세션(2/3)

- VO 클래스 작성 및 Mapping 클래스 작성
- EgovUsersByUsernamemapping 클래스를 상속받아서 mapRow 메소드에 VO를 메핑한다.

```
public class EgovUserDetailsMapping extends EgovUsersByUsernameMapping {  
    public EgovUserDetailsMapping(DataSource ds, String usersByUsernameQuery) {  
        super(ds, usersByUsernameQuery);  
    }  
    @Override  
    protected EgovUserDetails mapRow(ResultSet rs, int rownum) throws SQLException {  
  
        String userid = rs.getString("user_id");  
        String password = rs.getString("password");  
        boolean enabled = rs.getBoolean("enabled");  
        String username = rs.getString("user_name");  
        String birthDay = rs.getString("birth_day");  
        String ssn = rs.getString("ssn");  
  
        EgovUserDetailsVO userVO = new EgovUserDetailsVO();  
        userVO.setUserId(userid);  
        userVO.setPassWord(password);  
        userVO.setUserName(username);  
        userVO.setBirthDay(birthDay);  
        userVO.setSsn(ssn);  
  
        return new EgovUserDetails(userid, password, enabled, userVO);  
    }  
}
```

□ Authentication(인증) – 세션(3/3)

– 세션사용

```
import org.egovframe.rte.fdl.security.userdetails.util.EgovUserDetailsHelper;
...
EgovUserDetailsVO user = (EgovUserDetailsVO)EgovUserDetailsHelper.getAuthenticatedUser();

assertEquals("jimi", user.getUserId());
assertEquals("jimi test", user.getUserName());
assertEquals("19800604", user.getBirthDay());
assertEquals("1234567890123", user.getSsn());
```

– 인증여부 확인

```
Boolean isAuthenticated = EgovUserDetailsHelper.isAuthenticated(); assertFalse(isAuthenticated.booleanValue());
```

– 인증여부 확인 시 오류 발생 (null이 return되지 않음)

```
assertNull(EgovUserDetailsHelper.getAuthenticatedUser());
```

□ Authorization(권한부여) – 개요

- 웹 사이트에 존재하는 모든 사용자들은 사이트 정책에 따라 그 부류별로 컨텐츠에 대한 접근이 제한
- 특정 사용자가 웹 사이트에서 제공하는 컨텐츠(정보 또는 기능)에 접근 가능한지를 판단하는 과정
- 종류
 - Common
 - Secure Object(보안객체)
 - FilterInvocation 보호 – Filter Security Interceptor
 - Method Invocation 보호 – Method Security Interceptor – AOP Alliance, AspectJ
 - Domain Object Security
- 보안 결정 : 누가(Authentication) 어디서(MethodInvocation) 무엇을 ! (SomeDomainObject)
- 인증 결정은 메소드 호출 대상인 실제 도메인 객체 인스턴스를 고려할 필요도 있음

□ Authorization(권한부여) – 자원관리(1/5)

– url

- filterSecurityInterceptor

```
<b:bean id="filterSecurityInterceptor"
  class="org.springframework.security.web.access.intercept.FilterSecurityInterceptor">
  <b:property name="authenticationManager" ref="org.springframework.security.authenticationManager" />
  <b:property name="accessDecisionManager" ref="org.springframework.security.access.vote.AffirmativeBased#0" />
  <b:property name="securityMetadataSource" ref="databaseSecurityMetadataSource" />
</b:bean>
...
<http access-denied-page="/system/accessDenied.do" request-matcher="regex">
  ...
  <!-- for authorization -->
  <custom-filter before="FILTER_SECURITY_INTERCEPTOR" ref="filterSecurityInterceptor"/>
</http>
```

- databaseSecurityMetadataSource

```
<b:bean id="databaseSecurityMetadataSource"
  class="org.egovframe.rte.fdl.security.intercept.EgovReloadableFilterInvocationSecurityMetadataSource">
  <b:constructor-arg ref="requestMap" />
  <b:property name="securedObjectService" ref="securedObjectService"/>
</b:bean>
```

❑ Authorization(권한부여) – 자원관리(2/5)

- url (cont.)
 - requestMap

```
<b:bean id="requestMap" class="org.egovframe.rte.fdl.security.intercept.UrlResourcesMapFactoryBean"
    init-method="init">
    <b:property name="securedObjectService" ref="securedObjectService"/>
</b:bean>
```

❑ Authorization(권한부여) – 자원관리(3/5)

– Method

- methodSecurityMetadataSourceAdvisor

```
<b:bean id="methodSecurityMetadataSourceAdvisor"  
    class="org.springframework.security.access.intercept.aopalliance.MethodSecurityMetadataSourceAdvisor">  
    <b:constructor-arg value="methodSecurityInterceptor" />  
    <b:constructor-arg ref="delegatingMethodSecurityMetadataSource" />  
    <b:constructor-arg value="delegatingMethodSecurityMetadataSource" />  
</b:bean>
```

- methodSecurityInterceptor

```
<b:bean id="methodSecurityInterceptor"  
    class="org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor">  
    <b:property name="validateConfigAttributes" value="false" />  
    <b:property name="authenticationManager" ref="org.springframework.security.authenticationManager"/>  
    <b:property name="accessDecisionManager" ref="org.springframework.security.access.vote.AffirmativeBased#0"/>  
    <b:property name="securityMetadataSource" ref="delegatingMethodSecurityMetadataSource" />  
</b:bean>
```

❑ Authorization(권한부여) – 자원관리(4/5)

- methodSecurityMetatdataSources

```
<b:bean id="methodSecurityMetadatasources"
  class="org.springframework.security.access.method.MapBasedMethodSecurityMetadataSource">
  <b:constructor-arg ref="methodMap" />
</b:bean>
```

- methodMap

```
<b:bean id="methodMap" class="org.egovframe.rte.fdl.security.intercept.MethodResourcesMapFactoryBean"
  init-method="init">
  <b:property name="securedObjectService" ref="securedObjectService"/>
  <b:property name="resourceType" value="method"/>
</b:bean>
```

❑ Authorization(권한부여) – 자원관리(5/5)

– pointCut

- protectPointcutPostProcessor

```
<b:bean id="protectPointcutPostProcessor"  
    class="org.springframework.security.config.method.ProtectPointcutPostProcessor">  
    <b:constructor-arg ref="methodSecurityMetadataSources" />  
    <b:property name="pointcutMap" ref="pointcutMap"/>  
</b:bean>
```

- pointcutMap

```
<b:bean id="pointcutMap" class="org.egovframe.rte.fdl.security.intercept.MethodResourcesMapFactoryBean"  
    init-method="init">  
    <b:property name="securedObjectService" ref="securedObjectService"/>  
    <b:property name="resourceType" value="pointcut"/>  
</b:bean>
```


❑ Authorization(권한부여) – 역할관리(1/3)

- Application Context에 계층 역할을 등록하여 관리할 경우

```
<b:bean id="roleHierarchy" class="org.springframework.security.access.hierarchicalroles.RoleHierarchyImpl" >
  <b:property name="hierarchy">
    <b:value>
      ROLE_ADMIN > ROLE_USER
      ROLE_USER > ROLE_RESTRICTED
      ROLE_RESTRICTED > IS_AUTHENTICATED_FULLY
      IS_AUTHENTICATED_REMEMBERED > IS_AUTHENTICATED_ANONYMOUSLY
    </b:value>
  </b:property>
</b:bean>
```

□ Authorization(권한부여) – 역할관리(2/3)

- DB에서 계층 역할을 관리할 경우

```
<b:bean id="roleHierarchy" class="org.springframework.security.access.hierarchicalroles.RoleHierarchyImpl" >
  <b:property name="hierarchy" ref="hierarchyStrings"/>
</b:bean>

<b:bean id="hierarchyStrings"
  class="org.egovframe.rte.fdl.security.userdetails.hierarchicalroles.HierarchyStringsFactoryBean" init-method="init">
  <b:property name="securedObjectService" ref="securedObjectService"/>
</b:bean>

<b:bean id="securedObjectService"
  class="org.egovframe.rte.fdl.security.securedobject.impl.SecuredObjectServiceImpl">
  <b:property name="securedObjectDAO" ref="securedObjectDAO"/>
  <b:property name="requestMatcherType" value="regex"/><!-- default : ant -->
</b:bean>

<b:bean id="securedObjectDAO" class="org.egovframe.rte.fdl.security.securedobject.impl.SecuredObjectDAO">
  <beans:property name="dataSource" ref="dataSource"/>
</b:bean>
```

□ Authorization(권한부여) – 역할관리(3/3)

- securedObjectDAO (DBMS 또는 레이아웃 변경으로 인한 쿼리 변경 시 사용)

```
<b:bean id="securedObjectDAO" class="org.egovframe.rte.fdl.security.securedobject.impl.SecuredObjectDAO">
  <b:property name="dataSource" ref="dataSource"/>
  <b:property name="sqlHierarchicalRoles">
    <b:value>SELECT a.child_role child, a.parent_role parent
      FROM ROLES_HIERARCHY a LEFT JOIN ROLES_HIERARCHY b on (a.child_role = b.parent_role)</b:value>
  </b:property>
  <b:property name="sqlRolesAndUrl">
    <b:value>SELECT a.resource_pattern url, b.authority authority
      FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
      WHERE a.resource_id = b.resource_id AND a.resource_type = 'url' ORDER BY a.sort_order</b:value>
  </b:property>
  <b:property name="sqlRolesAndMethod">
    <b:value>SELECT a.resource_pattern method, b.authority authority
      FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
      WHERE a.resource_id = b.resource_id AND a.resource_type = 'method' ORDER BY a.sort_order</b:value>
  </b:property>
  <b:property name="sqlRolesAndPointcut">
    <b:value>SELECT a.resource_pattern pointcut, b.authority authority
      FROM SECURED_RESOURCES a, SECURED_RESOURCES_ROLE b
      WHERE a.resource_id = b.resource_id AND a.resource_type = 'pointcut' ORDER BY a.sort_order</b:value>
  </b:property>
</b:bean>
```

□ Authorization(권한부여) – 세션사용

– 역할

```
List<String> authorities = EgovUserDetailsHelper.getAuthorities();

// 사용예
// 1. authorities 에 권한이 있는지 체크 TRUE/FALSE
assertTrue(authorities.contains("ROLE_USER"));
assertTrue(authorities.contains("ROLE_RESTRICTED"));
assertTrue(authorities.contains("IS_AUTHENTICATED_ANONYMOUSLY"));
assertTrue(authorities.contains("IS_AUTHENTICATED_FULLY"));
assertTrue(authorities.contains("IS_AUTHENTICATED_REMEMBERED"));

// 2. authorities 에 ROLE 이 여러개 설정된 경우
for (Iterator<String> it = authorities.iterator(); it.hasNext();) {
    String auth = it.next();
}

// 3. authorities 에 ROLE 이 하나만 설정된 경우
String auth = (String) authorities.toArray()[0];
```

□ 설정 간소화

- 개요 : 표준프레임워크 3.0이상에서는 server security에 대하여 설정을 간소화 할 수 있는 방법을 제공
- XML namespace 및 schema 설정 (※ 표준프레임워크 실행환경 v4.0 예시)

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:egov-security="http://maven.egovframe.go.kr/schema/egov-security"
  xmlns:security="http://www.springframework.org/schema/security"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
    http://www.egovframe.go.kr/schema/egov-security
    http://www.egovframe.go.kr/schema/egov-security/egov-security-4.0.0.xsd
    http://www.springframework.org/schema/security
    http://www.springframework.org/schema/security/spring-security.xsd">
...
</beans>
```

□ 설정 간소화

– Security Config 설정

```
<egov-security:config id="securityConfig"
  loginUrl="/cvpl/EgovCvplLogin.do"
  logoutSuccessUrl="/cvpl/EgovCvplLogin.do"
  loginFailureUrl="/cvpl/EgovCvplLogin.do?login_error=1"
  accessDeniedUrl="/system/accessDenied.do"

  dataSource="dataSource"
  jdbcUsersByUsernameQuery="SELECT USER_ID,PASSWORD,ENABLED,USER_NAME,BIRTH_DAY,SSN
    FROM USERS WHERE USER_ID = ?"
  jdbcAuthoritiesByUsernameQuery="SELECT USER_ID,AUTHORITY FROM AUTHORITIES WHERE USER_ID = ?"
  jdbcMapClass="org.egovframe.rte.fdl.security.userdetails.EgovUserDetailsMapping"

  requestMatcherType="regex"
  hash="sha256"

  sniff="true"
  xframeOptions="SAMEORIGIN"
  xssProtection="true"
  csrf="false"
/>
```

□ 설정 간소화 (1/2)

– Security Config 설정 속성

| 속성 | 설명 | 필수여부 | 비고 |
|--------------------------------|---|------|---|
| loginUrl | 로그인 페이지 URL | 필수 | |
| logoutSuccessUrl | 로그아웃 처리 시 호출되는 페이지 URL | 필수 | |
| loginFailureUrl | 로그인 실패 시 호출되는 페이지 URL | 필수 | |
| accessDeniedUrl | 권한이 없는 경우 호출되는 페이지 URL | 필수 | |
| dataSource | DBMS 설정 dataSource | 선택 | 미지정시 'dataSource' bean name 사용 |
| jdbcUsersByUsernameQuery | 인증에 사용되는 query | 선택 | default : “select user_id, password, enabled, users.* from users where user_id = ?” |
| jdbcAuthoritiesByUsernameQuery | 인증된 사용자의 권한(authority) 조회 query | 선택 | default : “select user_id, authority from authorities where user_id = ?” |
| jdbcMapClass | 사용자 정보 mapping 처리 class | 선택 | default : egovframework.rte.fdl.security.userdetails.DefaultMapUserDetailsMapping |
| requestMatcherType | 패턴 매칭 방식(regex, ant, ciRegex: case-insensitive regex) | 선택 | default : regex |
| hash | 패스워드 저장 방식 (sha-256, plaintext, sha, md5, bcrypt) | 선택 | default : sha-256 |
| hashBase64 | hash값 base64 인코딩 사용 여부 | 선택 | default : true |

□ 설정 간소화 (2/2)

| 속성 | 설명 | 필수여부 | 비고 |
|---------------------------|--|------|-------------------------------------|
| concurrentMaxSessions | 동시 접속가능 연결수 | 선택 | default : 999 |
| concurrentExpiredUrl | expired된 경우 redirect되는 페이지 URL | 선택 | |
| errorIfMaximumExceeded | 중복 로그인 방지 옵션 | 필수 | default : false |
| defaultTargetUrl | 로그인 성공시 redirect되는 페이지 URL | 선택 | 미지정시 처음 접속하고자 했던 페이지 URL로 redirect됨 |
| alwaysUseDefaultTargetUrl | 로그인 이후 설정한 페이지로 이동하게 하는 옵션 | 필수 | default : true |
| sniff | 선언된 콘텐츠 유형으로부터 벗어난 응답에 대한 브라우저의 MIME 가로채기를 방지 여부 | 필수 | default : true |
| xframeOptions | sniff 옵션 이 ture 일때 X-Frame-Options 범위설정 | 선택 | DENY, SAMEORIGIN |
| xssProtection | XSS Protection 기능의 사용 여부 | 필수 | default : true |
| cacheControl | 캐쉬 비활성화 여부 옵션 | 필수 | default : false |
| csrf | spring security의 csrf 기능 사용 여부 | 필수 | default : false |
| csrfAccessDeniedUrl | 토큰 검증이 실패했을 경우 호출되는 페이지 URL | 필수 | |

□ 설정 간소화

- Security Config Initializer 설정

```
<egov-security:initializer id="initializer"  
    supportPointcut="true"  
>
```

- Security Config Initializer 설정 속성

| 속성 | 설명 | 필수여부 | 비고 |
|-----------------|-------------------|------|-----------------|
| supportPointcut | pointcut 방식 지원 여부 | 선택 | default : false |
| supportMethod | method 방식 지원 여부 | 선택 | default : true |

□ 설정 간소화

– Security Object Config 설정

```
<egov-security:secured-object-config id="securedObjectConfig" roleHierarchyString="
  ROLE_ADMIN > ROLE_USER
  ROLE_USER > ROLE_RESTRICTED
  ROLE_RESTRICTED > IS_AUTHENTICATED_FULLY
  IS_AUTHENTICATED_FULLY > IS_AUTHENTICATED_REMEMBERED
  IS_AUTHENTICATED_REMEMBERED > IS_AUTHENTICATED_ANONYMOUSLY"
sqlRolesAndUrl="
  SELECT auth.URL url, code.CODE_NM authority
  FROM RTETNAUTH auth, RTETCCODE code
  WHERE code.CODE_ID = auth.MNGR_SE"
/>
```

– Security Object Config 설정 속성

| 속성 | 설명 | 필수여부 | 비고 |
|-------------------------------|--|------|--------------------------------------|
| roleHierarchyString | 계층처리를 위한 설정 문자열 지정 | 선택 | 미지정시 DB로부터 지정된 설정정보 지정 |
| sqlRolesAndUrl | URL 방식 role 지정 query | 선택 | 미지정시 SecuredObjectDAO의 기본 query가 처리됨 |
| sqlRolesAndMethod | method 방식 role 지정 query | 선택 | " |
| sqlRolesAndPointcut | pointcut 방식 role 지정 query | 선택 | " |
| sqlRegexMatchedRequestMapping | request 마다 best matching url 보호자원 지정 query | 선택 | " |
| sqlHierarchicalRoles | 계층처리를 위한 query | 선택 | " |

❑ Spring Framework–Spring Security

- <https://spring.io/projects/spring-security>

❑ Spring Framework–Spring Security Reference Documentation

- <https://docs.spring.io/spring-security/reference/5.7.1/index.html>

□ 서비스 개요

- String 데이터를 다루기 위한 다양한 기능을 제공하는 서비스

□ 주요 기능

- Pattern Matching
 - String이 특정 Pattern (정규표현식(Regular Expression) 등)에 부합하는지 검사
- Formatting
 - 다양한 타입의 데이터를 특정 String 형식(Format)으로 변환
- Substring
 - 전체 String 중 일부를 가져옴
- Trim
 - 전체 String 중 앞뒤에 존재하는 공백 문자(white character) 제거
- Concatenate
 - 두 String을 붙여서 하나의 String을 생성
- Find
 - 전체 String 중 특정 String Pattern이 있는지 검색

□ EgovStringUtil(1/4)

- Pattern Matching
 - String이 특정 Pattern(정규표현식)에 부합하는지 검사한다.

```
// pattern match 성공
String str = "abc-def";
pattern = "*_*";
assertTrue(EgovStringUtil.isPatternMatching(str, pattern));
// pattern match 실패 str = "abc";
assertTrue(!EgovStringUtil.isPatternMatching(str, pattern));
```

□ EgovStringUtil(2/4)

– Formatting

- 다양한 타입의 데이터를 특정 String형식(Format)으로 변환한다.

```
// int => string
assertEquals("1", EgovStringUtil.integer2string(1));
// long => string
assertEquals("1000000000", EgovStringUtil.long2string(1000000000));
// float => string
assertEquals("34.5", EgovStringUtil.float2string(34.5f));
// double => string
assertEquals("34.5", EgovStringUtil.double2string(34.5));
// string => int
assertEquals(1, EgovStringUtil.string2integer("1"));
assertEquals(0, EgovStringUtil.string2integer(null, 0));
// string => float
assertEquals(Float.valueOf(34.5f), Float.valueOf(EgovStringUtil.string2float("34.5")));
assertEquals(Float.valueOf(10.5f), Float.valueOf(EgovStringUtil.string2float(null, 10.5f)));
// string => double
assertEquals(Double.valueOf(34.5), Double.valueOf(EgovStringUtil.string2double("34.5")));
assertEquals(Double.valueOf(34.5), Double.valueOf(EgovStringUtil.string2double(null, 34.5)));
// string => long
assertEquals(1000000000, EgovStringUtil.string2long("1000000000")); assertEquals(1000000000,
EgovStringUtil.string2long(null, 1000000000));
```

□ EgovStringUtil(3/4)

– Substring

- 전체 String 중 일부를 가져온다.

```
String source = "substring test";  
assertEquals("test", EgovStringUtil.toSubString(source, 10));  
assertEquals("string", EgovStringUtil.toSubString(source, 3, 9));
```

– Trim

- 전체 String 중 앞뒤에 존재하는 공백 문자(white character)를 제거한다.

```
String source = " substring ";  
  
assertEquals("substring", EgovStringUtil.trim(str));  
assertEquals("substring", EgovStringUtil.ltrim(str));  
assertEquals(" substring", EgovStringUtil.rtrim(str)); }
```

□ EgovStringUtil(4/4)

– Concatenate

- 두 String을 붙여서 하나의 String을 생성한다.

```
String str1 = "substring";  
String str2 = "test";  
  
assertEquals("substringtest", EgovStringUtil.concat(str1, str2));
```

– Find

- 전체 String 중 특정 String Pattern이 있는지 찾는다.

```
String pattern = "\\d{4}-\\d{1,2}-\\d{1,2}";  
// 일치하는 pattern 을 찾는다.  
Matcher matcher = Pattern.compile(pattern).matcher("2009-02-03");  
assertTrue(matcher.find());  
// 일치하는 pattern 을 찾는다.  
matcher = Pattern.compile(pattern).matcher("abcdef2009-02-03abcdef");  
assertTrue(matcher.find());  
// 일치하는 pattern 을 찾지 못한다.  
matcher = Pattern.compile(pattern).matcher("abcdef2009-02-A3abcdef");  
assertFalse(matcher.find());
```


□ EgovNumericUtil

- 숫자체크 기능
- 더하기 기능
- 빼기 기능
- 곱하기 기능
- 나누기 기능
- 올림, 내림 기능

□ EgovNumericUtil

// 숫자체크

```
assertFalse(EgovNumericUtil.isNumber("abc"));  
assertTrue(EgovNumericUtil.isNumber("1234"));
```

// 덧셈

```
assertEquals("399.9654", EgovNumericUtil.plus("151.7531", "248.2123"));  
assertEquals("399.97", EgovNumericUtil.plus("151.7531", "248.2123", 2, EgovNumericUtil.ROUND_HALF_UP));
```

// 뺄셈

```
assertEquals("89", EgovNumericUtil.minus("240", "151"));  
assertEquals("96.460", EgovNumericUtil.minus("248.2123", "151.7531", 3, EgovNumericUtil.ROUND_UP));
```

// 곱셈

```
assertEquals("180", EgovNumericUtil.multiply("15", "12"));  
assertEquals("189.613", EgovNumericUtil.multiply("15.23", "12.45", 3, EgovNumericUtil.ROUND_DOWN));
```

// 나눗셈

```
assertEquals("1.22", EgovNumericUtil.divide("15.23", "12.45", 5));  
assertEquals("1.224", EgovNumericUtil.divide("15.23", "12.45", 3, EgovNumericUtil.ROUND_UP));
```

□ EgovDateUtil

- 날짜계산 기능
- 현재일자조회 기능
- 요일 기능

□ EgovDateUtil

// 날짜계산

```
assertEquals("20100114", EgovDateUtil.getCalcDateAsString("2009", "3", "20", 300, "day"));
assertEquals("2010", EgovDateUtil.getCalcYearAsString("2009", "3", "20", 300, "day"));
assertEquals(2010, EgovDateUtil.getCalcYearAsInt("2009", "3", "20", 300, "day"));

assertEquals(90, EgovDateUtil.getDayCount("20090101", "20090401"));
assertEquals(90, EgovDateUtil.getDayCountWithFormatter("20090101", "20090401", "yyyyMMdd"));
assertEquals(60000, EgovDateUtil.getTimeCount("20090301000000", "20090301000100"));
```

// 현재일자조회

```
assertEquals(Calendar.getInstance().get(Calendar.YEAR),
    EgovDateUtil.getCurrentYearAsInt());
assertEquals(Calendar.getInstance().get(Calendar.DAY_OF_MONTH),
    EgovDateUtil.getCurrentDayAsInt());
```

// 요일기능

```
assertEquals("일", EgovDateUtil.getDayOfWeekAsString("2009", "03", "22"));
assertEquals(5, EgovDateUtil.getDayOfWeekCount("20090301", "20090331", "일요일"));
```

□ EgovDateUtil

```
// 날짜형식체크
public void testDateFormatCheck() throws Exception {
    // 형식이 틀린경우 ParseException 발생
    Class<Exception> exceptionClass = null;
    try {
        dateFormatCheck = EgovDateUtil.dateFormatCheck("20090300");
    } catch (Exception e) {
        exceptionClass = (Class<Exception>) e.getClass();
    } finally {
        assertEquals(ParseException.class, exceptionClass);
    }
}
```

□ EgovObjectUtil

- Instantiate 기능
- Instantiate 기능 (생성자의 파라미터 포함)

□ EgovObjectUtil

```
@Test
public void testInstantiate() throws Exception {
    String className = "java.lang.String";
    Object object = EgovObjectUtil.instantiate(className);
    String string = (String) object;
    string = "eGovFramework";
    assertEquals("Framework", string.substring(4));
}

@Test
public void testInstantiateParamConstructor() throws Exception {
    String className = "java.lang.StringBuffer";
    String[] types = new String[]{"java.lang.String"};
    Object[] values = new Object[]{"전자정부 공통서비스"};
    StringBuffer sb = (StringBuffer)EgovObjectUtil.instantiate(className, types, values);
    sb.append(" 및 개발프레임워크 구축 사업"); assertEquals("전자정부 공통서비스 및 개발프레임워크 구축 사업", sb.toString());
}
```

□ Jakarta Regexp

- <http://jakarta.apache.org/regexp>

□ 서비스 개요

- XML을 생성하고, 읽기, 쓰기 등의 조작 기능을 제공하는 서비스

□ 주요 기능

- XML Parser
 - XML 문서를 읽어 들이는 역할을 수행하는 파서는 두 가지 종류가 있다. XML 파일의 내용을 트리 구조로 한번에 읽어 들여 객체를 생성하여 처리하는 DOM(Document Object Model) 과 각각의 태그와 내용 등이 인식될 때마다 XML 문서를 읽어 들이는 SAX(Simple API for XML)라는 기술이다.
- DOM(Document Object Model)
 - XML 문서는 **요소(element)**, **속성(attribute)**, **Text** 등으로 구성된 트리 구조의 계층적인 정보이다. ⇒DOM을 이용하면 XML 문서의 각 요소들에 대하여 트리 구조의 객체를 읽어 들인다. DOM은 XML 문서를 나타내는 각각의 객체들에 대한 표준 인터페이스이다. DOM 파서는 XML 문서로부터 DOM 구조를 생성하는 역할을 한다.
- SAX(Simple API for XML)
 - XML 문서를 읽어 들이는 응용 프로그램 API 로써 XML 문서를 하나의 긴 문자열로 간주한다. SAX는 문자열을 앞에서 부터 차례로 읽어 가면서 요소, 속성이 인식될 때 마다 **EVENT**를 발생시킨다. 각각의 EVENT가 발생 될 때 마다 수행하고자 하는 기능을 **이벤트 핸들러 기술**을 이용하여 구현한다.

– Guide Program

- Configuration(context-xmltest.xml, egovxml.properties)

```
<bean id="xmlCofig" class="org.egovframe.rte.fdl.xml.EgovXmlset">
  <property name="domconcrete" ref="domconcreteCont" />
  <property name="saxconcrete" ref="saxconcreteCont" />
</bean>
<bean id="domconcreteCont" class="org.egovframe.rte.fdl.xml.EgovConcreteDOMFactory" />
<bean id="saxconcreteCont" class="org.egovframe.rte.fdl.xml.EgovConcreteSAXFactory" />
```

EgovDOMValidatorService
생성하는 Concrete Class

EgovSAXValidatorService
생성하는 Concrete Class

egovxmlsaved.path=C:\\Temp\\

– Guide Program

- Dom Service 생성


```
@Test
public void ModuleTest() throws UnsupportedOperationException
{
    domValidator = domconcrete.CreateDOMValidator();
    logger.debug("fileName :"+fileName);
    domValidator.setXMLFile(fileName);
}
```

- SAX Service 생성


```
@Test
public void ModuleTest() throws UnsupportedOperationException {
    saxValidator = saxconcrete.CreateSAXValidator();
    logger.debug("fileName :"+fileName);
    saxValidator.setXMLFile(fileName);
}
```

– Guide Program

- Well-formed, Validation 검사(XML문서의 well-formed 검사를 하면서 Validation검사도 동시에 실행(선택))

```
public void WellformedValidate(boolean used,boolean isvalid,AbstractXMLUtility service) throws ValidatorException {  
    if(used) {  
        if( service.parse(isvalid)) {  
            if(isvalid)  'True' 설정 시 Validation 검사 실행  
                logger.debug("Validation 문서입니다.");  
            else  
                logger.debug("well-formed 문서입니다.");  
        }  
    }  
}
```


- XPATH 조회

```
public void XPathResult(boolean used,AbstractXMLUtility service,Document doc) throws JDOMException {  
    if(used) { List list = service.getResult(doc,"//*[ @*]"); viewElement(list);  표현식 사용  
    }  
}
```

– Guide Program

- XML 생성(입력받은 Element를 사용하여 XML문서를 생성)

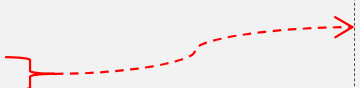
```
public void createNewXML(boolean used, AbstractXMLUtility service, Document doc, String EleName, List list, String path)
throws JDOMException, TransformerException, FileNotFoundException
{
    if(used)
        service.createNewXML(doc, EleName, list, path);
}
```



- doc : document 객체
- EleName : Root 명
- list : 생성 Element List
- path : 생성될 XML문서 경로

- Element 추가(입력받은 Element를 XML문서에 추가)

```
public void addElement(boolean used, AbstractXMLUtility service, Document doc, String EleName, List list, String path)
throws JDOMException, TransformerException, FileNotFoundException
{
    if(used)
        service.addElement(doc, EleName, list, path);
}
```




- doc : document 객체
- EleName : Root 명
- list : 생성 Element List
- path : 생성될 XML문서 경로

– Guide Program

- XmlNode Element 추가(입력받은 Text Element를 XML문서에 추가)

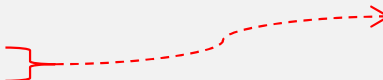
```
public void addTextElement(boolean used, AbstractXMLUtility service, Document doc, String EleName, List list, String path)
throws JDOMException, TransformerException, FileNotFoundException
{
    if(used)
        service.addTextElement(doc, EleName, list, path);
}
```



- doc : document 객체
- EleName : Root 명
- list : 생성 Element List
- path : 생성될 XML문서 경로

- XmlNode Element 수정(입력받은 Text Element로 수정)

```
public void updTextElement(boolean used, AbstractXMLUtility service, Document doc, String EleName, List list, String path)
throws JDOMException, TransformerException, FileNotFoundException
{
    if(used)
        service.updTextElement(doc, list, path);
}
```




- doc : document 객체
- list : 생성 Element List
- path : 생성될 XML문서 경로

– Guide Program

- Element 삭제(입력받은 Element을 삭제)


```
public void addTextElement(boolean used, AbstractXMLUtility service, Document doc, String EleName, List list, String path)
throws JDOMException, TransformerException, FileNotFoundException
{
    if(used)
        service.delElement(doc, EleName, path);
}
```



- doc : document 객체
- EleName : Root 명
- path : 생성될 XML문서 경로

- Element 수정

```
public void updTextElement(boolean used, AbstractXMLUtility service, Document doc, String EleName, List list, String path)
throws JDOMException, TransformerException, FileNotFoundException
{
    if(used)
        service.updElement(doc, oldElement, newElement, path);
}
```



- doc : document 객체
- oldElement : 수정할 Element 명
- newElement : 수정 Element 명
- path : 생성될 XML문서 경로

❑ JDOM

- <http://www.jdom.org/>

❑ Apache Xerces

- <http://xerces.apache.org/>