1 제품 개요 .......................................................................................................................... 4
   가) 제품 설명.................................................................................................................. 4
   나) 시스템 구성도.......................................................................................................... 4
   다) 이벤트 처리과정..................................................................................................... 5
       (1) Trigger Mode......................................................................................................... 5
       (2) Immediate Mode................................................................................................. 6
   라) 제품의 특징............................................................................................................. 6
       (1) 주요 기능............................................................................................................. 6
       (2) 특징..................................................................................................................... 7
       (3) 독창성................................................................................................................ 7
2 제품의 설치 .................................................................................................................... 8
   가) 설치 ....................................................................................................................... 8
   나) WINDOWS NT Service 설치................................................................................ 11
   다) 제거 ....................................................................................................................... 11
   라) WINDOWS NT Service 삭제................................................................................ 12
3 제품의 실행 .................................................................................................................... 13
   가) “XLogic Middleware Platform” 실행................................................................. 13
   나) 웹 인터페이스 실행.............................................................................................. 14
4 제품의 사용 .................................................................................................................... 15
   가) Server Status........................................................................................................ 16
       A Server Settings.................................................................................................... 16
       B Cached Scripts(Cached Repository)..................................................................... 17
       C Script Statistics................................................................................................... 18
       D Subscriber Statistics.......................................................................................... 19
   나) Object ID Provider................................................................................................ 19
       A Provider List........................................................................................................ 19
       B Add Provider....................................................................................................... 20
   다) XLogic Script........................................................................................................ 21
       A Script List............................................................................................................ 21
       B New Script........................................................................................................... 22
   라) User Account........................................................................................................ 23
       A User List.............................................................................................................. 23
       B New User............................................................................................................ 24
5 XLogic Script 작성법 ...................................................................................................... 25
   가) <XLogicScript> Tag.............................................................................................. 25
       (1) 설명.................................................................................................................. 25
       (2) 작성 요령.......................................................................................................... 25
       (3) Attribute........................................................................................................... 25
   나) <Set> Tag.............................................................................................................. 25
다) <InvokeWebService> Tag ............................................................................ 26
   (1) 설명 ........................................................................................................ 26
   (2) 작성 요령 ............................................................................................... 26
   (3) Attribute ............................................................................................... 26
라) <Iterator> Tag ............................................................................................ 26
   (1) 설명 ........................................................................................................ 26
   (2) 작성 요령 ............................................................................................... 26
   (3) Attribute ............................................................................................... 26
마) <Wait> Tag .................................................................................................. 27
   (1) 설명 ........................................................................................................ 27
   (2) 작성 요령 ............................................................................................... 27
바) <If> Tag ....................................................................................................... 27
   (1) 설명 ........................................................................................................ 27
   (2) 작성 요령 ............................................................................................... 27
   (3) Attribute ............................................................................................... 27
사) <Then> Tag .................................................................................................. 27
   (1) 설명 ........................................................................................................ 27
   (2) 작업 요령 ............................................................................................... 27
아) <Else> Tag .................................................................................................. 27
   (1) 설명 ........................................................................................................ 27
   (2) 작성 요령 ............................................................................................... 28
자) <While> Tag ................................................................................................ 28
   (1) 설명 ........................................................................................................ 28
   (2) 작성 요령 ............................................................................................... 28
   (3) Attribute ............................................................................................... 28
차) <ForEach> Tag ............................................................................................ 28
   (1) 설명 ........................................................................................................ 28
   (2) 작성 요령 ............................................................................................... 28
   (3) Attribute ............................................................................................... 28
카) <Break> Tag ................................................................................................ 28
   (1) 설명 ........................................................................................................ 28
   (2) 작성 요령 ............................................................................................... 29
타) <Continue> Tag .......................................................................................... 29
   (1) 설명 ........................................................................................................ 29
   (2) 작성 요령 ............................................................................................... 29
파) <Print> Tag .................................................................................................. 29
(1) 설명........................................................................................................29
(2) 작성 요령..............................................................................................29

6 테스트 시나리오.............................................................................................30

부록 1. xlogic.xsd.........................................................................................45

부록 2. Selecting XML with XQuery and XPath().........................................47
  Using XPath with the selectPath Method....................................................47
  Calling XmlObject.selectPath.....................................................................47
  Calling XmlCursor.selectPath....................................................................48
  Using XQuery with the execQuery Method.................................................50
  Calling XmlObject.execQuery....................................................................50
  Calling XmlCursor.execQuery....................................................................51

부록 3. JEXL..................................................................................................53
  Language Elements.......................................................................................53
  Literals............................................................................................................53
  Functions.......................................................................................................53
  Operators.......................................................................................................54
  Conditional....................................................................................................55

부록 4. ..........................................................................................................56
제품 개요

가) 제품 설명
RFID/USN 환경으로의 변화는 기존의 여러 서비스 시스템들의 교체, 혹은 통합된 새로운 서비스 시스템의 구축을 요구합니다. 이러한 일들은 많은 비용과 시간을 필요로 하지만 전체를 통합한 부분을 제외한다면 각 서비스 시스템의 비즈니스 프로세스는 기존 시스템들의 그것과 크게 다르지 않습니다. 따라서 기존 시스템을 최대한 활용하면서 RFID/USN 환경에 적합한 통합된 서비스 시스템 환경을 손쉽게 구축하기 위한 방법은 매우 중요합니다.

“XLogic Middleware Platform”은 이미 구축되어 있는 기존 서비스 시스템들을 최대한 활용하면서 새로운 RFID/USN 환경에서 요구되는 통합된 시나리오를 지원할 수 있는 미들웨어 플랫폼입니다. 본 미들웨어 플랫폼은 독립적으로 존재하는 센서 네트워크 상의 미들웨어들에 대한 이벤트를 기반으로, 각 서비스 시스템 간의 협력이 필요한 비즈니스적 상황에 대비하여 이를 미리 정의함으로써 서비스 시스템간의 유기적인 상호인식 컴퓨팅 환경을 구축할 수 있도록 합니다.

따라서 기존 서비스 시스템을 그대로 유지하면서 새로운 비즈니스 시나리오를 관리하는 미들웨어 플랫폼을 제공하게 되어 구축 비용 및 기간의 단축은 물론 유지보수의 용이성도 기대할 수 있습니다.

나) 시스템 구성도

![그림 1] XLogic Middleware Platform 구조도
"XLogic Middleware Platform" RFID/USN 환경에서 시스템 통합을 위한 시나리오 기반 협업 지원 기능을 제공합니다. 미들웨어 플랫폼은 상위 시스템간 협업을 해결하기 위하여 네 종류의 Agent 와 이를 관리하는 Agent Manager 로 구성되며 사용자가 정의하는 시나리오를 지원할 수 있는 XML 형태의 XLogic(extensible Logic)이라는 인터프리터 언어를 제공합니다. 각 Agent 는 Agent Manager 를 중심으로 이벤트 정보를 교환하며 이들의 연관관계는 [그림 1]과 같습니다.

미들웨어 플랫폼은 이식성을 고려하여 Java 기반으로 제작하였고 이질적인 웹 서비스들과의 통신 인터페이스는 SOAP(Simple Object Access Protocol)로 정의 하였습니다. Repository 는 DB(Database)를 따로 설치할 필요 없이 플랫폼의 설치만으로 배포할 수 있고 특정 점의에서 ‘비모리 DBMS(Database Management System)’ 수준의 성능을 낼 수 있는 H2 DB 를 사용하였습니다. 사용자를 위한 서비스와 웹 인터페이스 제공을 위해 jetty 임베디드 WAS(Web Application Server)를 장착하였고 이는 플랫폼의 설치만으로 배포할 수 있습니다. 또한 XML 을 자바 객체로 파생할 때의 자원 소비를 줄이기 위해 스크립트를 최근 사용횟수에 따라 캐싱하는 방법을 사용하였습니다.

다) 이벤트 처리과정
플랫폼 내부에서 발생하는 이벤트의 종류는 [표 1]과 같이 두 가지의 방식으로 이벤트를 처리할 수 있습니다.

[표 1] 내부에서 발생하는 이벤트의 종류

<table>
<thead>
<tr>
<th>종류</th>
<th>설명</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReportsArrivedEvent</td>
<td>XML 형태로 전송되는 리포트를 저장하고 있는 Java 객체</td>
</tr>
<tr>
<td>XLogicScriptExecuteRequestEvent</td>
<td>XLogic의 통계정보와 실행계획을 가지고 있는 Java 객체</td>
</tr>
</tbody>
</table>

(1) Trigger Mode

[그림 2] Trigger Mode 이벤트 처리 과정

Agent Manager는 다시 등록 된 Agent들에게 모두 전달하고 ‘XLogicScriptExecuteRequestEvent’를 처리할 수 있는 Instance Manage Agent가 XLogic을 실행 시킵니다.

(2) Immediate Mode

사용자가 작성한 스크립트를 즉시 실행 시키는 방식입니다. 사용자가 EM에서 XLogic을 작성하여 실행 시키면 Instance Manage Agent에서 XML 형태로 작성된 XLogic을 Java 객체로 변환하여 실행합니다[그림 3].

라) 제품의 특징

(1) 주요 기능
- 사용자가 시나리오를 작성하면 시스템이 각 시나리오에 맞게 이동 시스템과의 협업을 지원합니다.
- RFID/USN 환경에서 다수 시스템과의 협업에 대한 시나리오를 ‘XLogic Middleware Platform’에서 직접 관리하여 특정 시스템에 시나리오가 종속되는 것을 방지합니다.
- RFID/USN 환경에서의 다양한 이벤트에 대해 사용자가 원하는 시나리오들을 실행 시킬 수 있습니다.
웹 인터페이스를 제공하여 사용자가 손쉽게 미들웨어 플랫폼의 모니터링을 할 수 있도록 가능하게 합니다.

(2) 특징
- Java 기반으로 제작 되어 있기 때문에 특정 O/S에 독립적임.
- 시나리오의 작성은 미리 지정한 XSD(XML Schema Definition)에 정의한대로 작성할 수 있습니.
- ‘XLogic Middleware Platform’을 사용함으로써 RFID/USN 미들웨어와 기존 시스템을 느슨하게 결합시킬 수 있습니다.
- RFID/USN 환경의 구현을 위해 기존 시스템을 새로 구축 할 필요가 없으므로 비용 절약의 효과를 가져 와옵니다.
- 패키지형 미들웨어 플랫폼입니다.

(3) 독창성
- 기존 시스템들의 최소한의 수정으로 시스템 통합이 가능합니다.
- 이벤트에 맞는 시나리오가 다수라도 다수의 시나리오를 모두 실행 시킬 수 있습니다.
- 기존의 시나리오에 새로운 시나리오의 주입이 가능하여 시나리오 간 상호 연계가 가능합니다.
- RFID/USN 장비에서 생성된 이벤트를 적재적소의 응용 서비스에 전달하는 기존의 USN 미들웨어의 개념과 다르게 USN 미들웨어와 응용 서비스 사이에 ‘XLogic Middleware Platform’을 배치시켜 USN 미들웨어가 전달하는 이벤트에 대한 처리를 응용 서비스간의 협업을 통해 처리합니.
2 제품의 설치

가) 설치

1) 컴퓨터 CD-ROM에 XLogic Middleware Platform 설치 CD를 삽입합니다.

2) 자동 실행이 되면서 설치 화면으로 전환됩니다.

※주의

만약 자동 실행이 되지 않을 경우 CD안에 있는 파일을 실행시킵니다.

3) [다음]을 클릭하면 설치가 진행됩니다.
4) [다음]을 클릭하면 설치가 진행됩니다.

전체설치 사용자 정의 설치

[변경]을 클릭하여 설치 위치를 지정합니다.

5) [설치]을 클릭하면 설치가 완료됩니다.
6) [마침]을 클릭하면 설치가 종료됩니다.

7) 설치한 폴더에 다음과 같은 폴더가 존재하면 올바르게 설치 된 것입니다.
나) WINDOWS NT Service 설치

1) 설치된 ‘xlogic/bin’ 폴더에서 “install-service.bat” 파일을 실행시키면 등록됩니다.

다) 제거

1) [Uninstall XLogic Middleware Platform]을 클릭합니다.

2) [예]를 클릭하면 제거됩니다.
라) WINDOWS NT Service 삭제

1) 설치된 ‘xlogic/bin’ 폴더에서 “uninstall-service.bat” 파일을 실행시키면 등록이 삭제됩니다.

※ 주의
“XLogic Middleware Platform” 삭제 시 반드시 NT 서비스를 먼저 삭제해 주십시오.
3 제품의 실행

가) “XLogic Middleware Platform” 실행

“XLogic Middleware Platform”이 설치된 하위 폴더에 “bin” 폴더에 있는 “run.bat” 파일을 실행시킵니다.

![시작 화면](image1)

command 창이 실행되면서 “XLogic Middleware Platform”이 실행됩니다.

“[Subscriber Agent] 5000 Port에서 Agent를 실행합니다.”라는 메시지가 출력되면서 5000번 포트로 이벤트를 받을 준비를 합니다.

![추가 화면](image2)
나) 웹 인터페이스 실행

“XLogic Middleware Platform”을 가동 시킨 후 아래의 방법으로 본 플랫폼을 관리할 수 있는 “XLogic Enterprise Manager”로 접속합니다. 기본 포트번호는 8010으로 설정되어 있습니다.

http://IP주소:포트번호 -> 예) http://117.16.44.76:8010

Username에 기본 아이디인 admin을 입력합니다.
Password에 기본 패스워드인 spoon을 입력합니다.
[Sign In]을 클릭하여 메인 메뉴로 이동할 수 있습니다.
4 제품의 사용

① Server Status
본 미들웨어 플랫폼에 대한 전반적인 설정사항과 인스턴스 가동 후 XLogic의 실행에 대한 부분을 조회할 수 있습니다.

② Object ID Provider
본 미들웨어 플랫폼이 수집하는 이벤트의 종류에 대한 고유의 값을 추출할 수 있는 공식을 삽입, 수정, 삭제, 조회를 할 수 있습니다.

③ XLogic Script
XLogic 스크립트에 대한 삽입, 수정, 삭제 조회를 할 수 있습니다.

④ User Account
본 미들웨어 플랫폼의 사용자에 대한 삽입, 수정, 삭제, 조회를 할 수 있습니다.

⑤ Add New Object ID Provider
Object ID Provider를 작성하는 페이지로 이동합니다.

⑥ Add New XLogic Script
XLogic Script를 작성하는 페이지로 이동합니다.

⑦ Add New User
User를 등록하는 페이지로 이동합니다.

⑧ User Logout
Logout 기능을 수행합니다.
가) Server Status

A Server Settings

i. Description: 해당 페이지에 대한 설명입니다.

ii. Do Garbage Collection: 클릭하시면 강제적으로 가비지 컬렉션을 수행합니다. 성능의 저하를 가져올 수 있으니 주의하여 사용하시기 바랍니다.

iii. Refresh: 페이지를 Refresh합니다.

iv. Memory Status
   - Used Memory: 현재 JVM(Java Virtual Machine)상의 메모리를 나타냅니다.
   - Free Memory: 현재 사용가능한 메모리를 나타냅니다.
   - Total Memory: User Memory와 Free Memory의 합을 나타냅니다.
   - Max Memory: 최대 사용가능한 메모리를 나타냅니다.

v. XLogic Middleware Platform Property
   - xlogic.agent.proxy.maxSize: 캐시 될 수 있는 스크립트의 최대 개수를 나타냅니다.
   - xlogic.agent.webserver.webappPath: XEM에 대한 웹 루트를 나타냅니다.
xlogic.agent.manager.eventBroadcasterPoolSize:
XLogic에서 일어나는 내부 이벤트의 대한 크기를 나타냅니다.
xlogic.agent.subscriber.poolSize:
본 플랫폼에 연결되는 호스트의 최대치를 나타냅니다.
xlogic.agent.proxy.loadFactor:
캐시의 부하계수를 나타냅니다.
xlogic.agent.webserver.port:
XEM에 대한 포트 번호를 나타냅니다.
xlogic.database.path:
미들웨어 플랫폼에 설치된 H2 DBMS에 대한 루트폴더의 위치를 나타냅니다.
xlogic.agent.subscriber.port:
본 플랫폼에 전달되는 이벤트의 포트를 나타냅니다.
xlogic.agentinstancemanage.maxExecutionThreadPoolSize:
동시에 최대 실행될 수 있는 XLogic의 크기를 나타냅니다.

B Cached Scripts(Cached Repository)

i. Description: 해당 페이지에 대한 설명입니다.
ii. Truncate cached scripts: 현재 캐시 되어 있는 스크립트를 모두 해제합니다.
iii. Refresh: 페이지를 Refresh합니다.
iv. Cached Repository Information

Cached Script Name: 현재 캐시 되어있는 스크립트의 이름을 나타냅니다.
Access Count: 캐시 되어 있는 스크립트에 대한 접근 횟수를 나타냅니다.
Cached Time: 캐시가 된 시간을 나타냅니다.
Last Accessed Time: 최근에 접근된 시간을 나타냅니다.
C Script Statistics

Description: 해당 페이지에 대한 설명입니다.

Refresh: 페이지를 Refresh합니다.

Script Information
- Script Name: 실행된 스크립트에 대한 이름을 나타냅니다.
- Success Count: 실행이 성공된 횟수를 나타냅니다.
- Failure Count: 실행에 실패한 횟수를 나타냅니다.
- Last Executed Time: 마지막으로 실행된 시간을 나타냅니다.
- Current State: 스크립트의 현재 상태를 나타냅니다. DONE은 실행종료를 나타내며, RUNNING은 실행 중인 상태를 나타냅니다.
D Subscriber Statistics

i. Description: 해당 페이지에 대한 설명입니다.

ii. Refresh: 페이지를 Refresh합니다

iii. Script Information
    Hostname: 현재 이벤트를 전송하고 있는 호스트에 대한 IP주소를 나타냅니다.
    Received Count: 현재 전송 받은 이벤트의 횟수를 나타냅니다.
    Last Received Time: 가장 최근에 이벤트를 받은 시간을 나타냅니다.

나) Object ID Provider

A Provider List

i. XML Root Tag: declare namespace ale="urn:epelglobal::alexml1.0"
ii. XPath Expression: /ale:ECReports/ale:ECPredicate[@specName]
i. **Description:** 해당 페이지에 대한 설명 입니다.

ii. **Object ID Provider Information**
   
   **XML Root Tag:** 전송받은 XML 형태의 이벤트에 대한 루트 태그에 대한 값을 나타냅니다.
   
   **XPath Expression:** 루트 태그에서 고유값을 추출하기 위한 XPath 공식을 나타냅니다.
   
   **Update:** Object ID Provider를 수정할 수 있습니다.
   
   **Delete:** Object ID Provider를 삭제합니다.

---

B **Add Provider**

---

i. **Description:** 작성한 부분에 대한 설명을 삽입합니다.

ii. **XML Root Tag:** 전송되는 XML 이벤트에 대한 ROOT tag를 삽입합니다.

iii. **XPath:** 전송되는 이벤트에 대한 고유 값을 추출하는 XPath에 대한 공식을 삽입합니다.

iv. **Description:** 작성한 부분에 대한 설명을 삽입합니다.

v. **Add Provider:** 입력을 등록 합니다.

vi. **Reset Form:** 제 작성 합니다.
다) XLogic Script

A Script List

   ![XLogic Middleware Platform Enterprise Manager](image)

   i. **Description**: 메뉴에 대한 설명 입니다.
   
   ii. **Script Information**

      - **Script Name**: 작성되어 있는 스크립트에 대한 이름을 나타냅니다. 이름을 클릭하면 해당 스크립트에 대한 정보를 보여줍니다.
      - **Source Name**: 스크립트의 고유의 값을 나타냅니다.
      - **Enable**: 스크립트의 사용가능 여부를 나타냅니다. 체크되어 있으면 실행되고, 해제되어 있으면 실행되지 않습니다.
      - **Username**: 작성자의 Username을 나타냅니다.
      - **Registration Date**: 등록된 날짜를 나타냅니다.
      - **Update**: XLogic 스크립트를 수정하는 페이지로 이동합니다.
      - **Delete**: 해당 스크립트를 삭제합니다.
      - **Execute**: 작성한 스크립트를 별도의 이벤트 없이 클릭만으로 실행할 수 있습니다. 이벤트의 실행 과정은 6 페이지를 참조하면 됩니다. (Immediate Mode)
B  New Script

i. Description: 해당 페이지에 대한 설명 입니다.

ii. Script Name: 스크립트에 대한 고유의 이름을 삽입합니다.

iii. Source Name: 스크립트의 고유한 값을 삽입합니다.

iv. Enable: 체크를 하면 사용 가능하고, 체크를 하지 않으면 사용 불가능합니다.

v. Description: 해당 스크립트에 대한 간단한 설명을 작성합니다.

vi. Script Code: 스크립트 코드를 작성합니다. (작성법 참조)

vii. Write Script: 입력한 스크립트를 등록합니다.

viii. Reset Form: 스크립트를 제 작성 합니다.
User Account

A User List

i. Description: 해당 페이지에 대한 설명입니다.

ii. User Information
   
   Username: Username에 대한 정보를 나타냅니다.
   Master: Master권한 여부에 대한 정보를 나타냅니다. Master권한은 XEM에 대한 모든 권한을 가지는 것을 나타냅니다.
   Provider: Object ID Provider에 대한 작성, 수정 권한을 나타냅니다.
   Script: 스크립트에 대한 작성, 수정 권한을 나타냅니다.
   Name: 사용자의 이름을 나타냅니다.
   E-mail: 사용자의 이메일 주소를 나타냅니다.
   Update: 사용자의 정보를 수정하는 페이지로 이동합니다.
   Delete: 사용자의 정보를 삭제합니다.
B. New User

i. Description: 해당 페이지에 대한 설명입니다.
ii. Username: Username을 작성합니다.
iii. Password: 패스워드를 작성합니다.
iv. Confirm Password: 작성했던 패스워드와 동일한 내용을 작성합니다.
v. Name: 사용자 이름을 작성합니다.(단 영어)
vi. E-Mail: 이메일 주소를 작성합니다.
vii. Master Role: 모든 권한을 가진 권한을 설정합니다.
viii. Provider Role: Object ID Provider의 작성, 수정, 삭제에 관한 권한을 설정합니다.
ix. Script Role: XLogic 스크립트에 대한 작성이, 수정, 삭제에 관한 권한을 설정합니다.
x. Add User Account: 입력을 등록 합니다.
xi. Reset Form: 제 입력 합니다.
5 XLogic Script 작성법

기본적인 표현식(Expression Language)은 JEXL(Java Expression Language)을 사용하였고 총 13가지 Tag를 가지고 있습니다. 작성 방법에 대해서는 "xlogic.xsd"라는 XSD(XML Schema Definition)에 정의하여 그 문법에 맞게 작성해야 합니다.

가) <XLogicScript> Tag

(1) 설명
XLogic 스크립트를 작성시 루트 테그를 항상 XLogicScript로 설정하여 작성하게 됩니다. XLogic의 모든 태그들을 포함할 수 있다.

(2) 작성 요령

\[
\begin{align*}
\text{xlogic:XLogicScript} \\
\text{xmlns:xlogic=}
\text{"urn:spoon:xlogic:script:xsd:1"} \\
\text{xmlns:xsi=}
\text{http://www.w3.org/2001/XMLSchema-instance} \\
\text{xsi:schemaLocation=}
\text{"urn:spoon:xlogic:script:xsd:1 http://spoon.smu.ac.kr/xlogic.xsd"} \\
\text{name=} \"" \\
\text{creationDate=} \""
\end{align*}
\]

\[</xlogic:XLogicScript>\]

(3) Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>xmlns:xlogic</td>
<td>xlogic의 Namespace</td>
<td>○</td>
<td>X</td>
</tr>
<tr>
<td>xmlns:xsi</td>
<td>xsi의 Namespace</td>
<td>○</td>
<td>X</td>
</tr>
<tr>
<td>xsi:schemaLocation</td>
<td>Namespace와 Schema문서 위치에 대한 값으로 가짐</td>
<td>○</td>
<td>X</td>
</tr>
<tr>
<td>name</td>
<td>스크립트의 이름</td>
<td>X</td>
<td>○</td>
</tr>
<tr>
<td>creationDate</td>
<td>스크립트의 생성 날짜</td>
<td>X</td>
<td>○</td>
</tr>
</tbody>
</table>

나) <Set> Tag

(1) 설명
Jexl을 이용하여 변수를 설정하는 역할을 하는 태그입니다. XPath를 이용하여 표현식의 값에서 원하는 값만을 추출할 수 있습니다.

(2) 작성 요령

\[
\begin{align*}
\text{xlogic:Set name=}
\text{"" select=""}$()</xlogic:Set>
\end{align*}
\]
다) <InvokeWebService> Tag

(1) 설명
웹 서비스를 호출할 수 있는 태그입니다.

(2) 작성 요령
<xlogic:InvokeWebService name="" url="" service="" port="" namespace="" async=""/>
</xlogic:InvokeWebService>

(3) Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>변수명을 설정함</td>
<td>○</td>
<td>X</td>
</tr>
<tr>
<td>select</td>
<td>Jexl로 얻어오는 값이 XML 태입일 경우의 XPath</td>
<td>X</td>
<td>○</td>
</tr>
</tbody>
</table>

라) <Iterator> Tag

(1) 설명
<InvokeWebService> Tag 안에만 존재할 수 있으며 웹 서비스를 호출할 경우의 인자값을 설정해 주는 역할을 하는 Tag입니다.

(2) 작성 요령
<xlogic:Iterator name="" source=""/></xlogic:Iterator>

(3) Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>Iterator에서 사용할 변수 명</td>
<td>○</td>
<td>X</td>
</tr>
<tr>
<td>source</td>
<td>xlogic 스크립트 상의 변수 명</td>
<td>○</td>
<td>X</td>
</tr>
</tbody>
</table>
마) <Wait> Tag

(1) 설명
Jexl에 숫자의 값을 넣어서 숫자의 값만큼 밀리세컨드 단위로 잠시 멈추게 하는 태그입니다.

(2) 작성 요령
<xlogic:Wait>${}</xlogic:Wait>

바) <If> Tag

(1) 설명
제어문을 나타내는 Tag입니다. 반드시 하나의 <Then> Tag가 들어가야 하고 condition attribute가 참을 나타낼 경우 <Then> Tag안의 문장이 실행됩니다.

(2) 작성 요령
<xlogic:If condition="${}"/></xlogic:If>

(3) Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>If 문의 조건. Boolean 값이 들어가는 표현식이 들어감</td>
<td>○</td>
<td>X</td>
</tr>
</tbody>
</table>

사) <Then> Tag

(1) 설명
<xIf> Tag안에 포함되는 Tag입니다. <If> Tag의 condition이 참일 경우에만 실행됩니다.

(2) 작업 요령
<xlogic:Then></xlogic:Then>

아) <Else> Tag

(1) 설명
<xIf> Tag안에 포함되는 Tag입니다. <If> Tag의 condition이 거짓일 경우에 실행되어야 할 문장이 있을 때 사용합니다.
(2) 작성 요령
<xlogic:Else></xlogic:Else>
자) <While> Tag
(1) 설명
Jexl로 표현된 condition의 값이 거짓을 나타낼 때까지 계속 반복되는 반복문입니다.
(2) 작성 요령
<xlogic:While condition="$1"></xlogic:While>
(3) Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>While 문의 조건. Boolean 값이 들어가는 표현식이 들어감</td>
<td>○</td>
<td>X</td>
</tr>
</tbody>
</table>

차) <ForEach> Tag
(1) 설명
하나 이상의 값을 가진 변수의 값을 하나씩 뽑아내는 반복문입니다.
(2) 작성 요령
<xlogic:ForEach name="" source=""></xlogic:ForEach>
(3) Attribute

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
<th>Required</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>source에 설정된 값을 받을 변수</td>
<td>○</td>
<td>X</td>
</tr>
<tr>
<td>source</td>
<td>하나 이상의 값을 가진 변수</td>
<td>○</td>
<td>X</td>
</tr>
</tbody>
</table>

카) <Break> Tag
(1) 설명
<While> Tag나 <ForEach> Tag등 반복문에서 인접한 반복문을 완전히 빠져 나갈 때 사용하는 Tag입니다.
(2) 작성 요령

<logic:Break/></logic:Break>

타) <Continue> Tag

(1) 설명

<While> Tag나 <ForEach> Tag등 반복문에서 인접한 반복문을 1회 빠져 나갈 때 사용하는 Tag입니다.

(2) 작성 요령

<logic:Continue/></logic:Continue>

파) <Print> Tag

(1) 설명

XLogic Script를 테스트할 목적으로 만든 Tag입니다. Jexl에 맞는 내용을 출력하는 기능을 합니다.

(2) 작성 요령

<logic:Print>${}</logic:Print>
6 테스트 시나리오

1) Tag 를 출력할 수 있는 웹 서비스를 제작합니다.

2) tomcat 루트 폴더에 제작한 웹 서비스 war 파일을 배치합니다.

3) tomcat 을 실행합니다.
4) http://localhost:8080/tagprint/services에서 WSDL을 확인해서 XLogic 스크립트 작성 시 필요한 정보를 추출 합니다.

5) XLogic 서버를 가동합니다.
6) EM 실행 후 로그인(아이디: admin 비밀번호: spoon) 합니다.

7) 스크립트 고유의 값을 TEST 라 정해주고 모든 태그들을 넣은 스크립트를 작성합니다(스크립트는 부록 4에 첨부).
8) RFID Edge Server 2.2 실행합니다.

9) [Next]를 클릭합니다.

10) 설치 동의 후 [Next] 클릭합니다.
11) [Next] 버튼 클릭합니다.

12) [Next] 버튼 클릭합니다.
13) [Next] 버튼 클릭합니다.

Choose Product Installation Directory
Provide the directory where you wish to install WebLogic RFID Edge Server 2.2.

BEA Home Directory
C:\bea

Product Installation Directory
The default product home will contain WebLogic RFID Edge Server, related components and utilities.
Default Product Home:
C:\bea\r\id\edge22

14) [Next] 버튼 클릭합니다.

Choose Shortcut Location
The installer creates shortcuts to BEA components, samples, and tools. As a user with administrative privileges, you can specify where these shortcuts are created.

Select the start menu folder in which you want to create BEA shortcuts:

- [All Users] Start menu folder (recommended)
  If a user without administrative privileges uses the BEA Configuration Wizard in this installation to create configuration, the user may have to create start menu shortcuts manually. Refer to the documentation for more information.

- Local user's Start Menu folder
  Select this option if you need to ensure that other profiles on this machine will not have access to these shortcuts.
15) [Done] 버튼 클릭하면 Configuration Wizard가 실행됩니다.

16) [Next] 버튼 클릭합니다.
17) [Next] 버튼 클릭합니다.

18) [create] 버튼 클릭합니다.
19) [Done] 버튼 클릭으로 설치가 완료됩니다.

20) [Run Reader Simulator] 실행 후 [Start Edge Server]를 실행합니다.
22) logical Readers 에 있는 안테나를 오른쪽으로 움직인 후 전송 주기(Duration)을 5000ms 로 설정합니다.


25) 왼쪽 상단 두 번째에 위치한 [Deploy As] 버튼 클릭 후 Reports 고유의 값으로 “TEST”를 작성 후 확인버튼을 클릭합니다.

27) [Type]을 TCP/IP로 설정합니다.
28) [Host] 항목에 전송해야 할 IP 주소를 작성하고 Port 번호는 5000 으로 설정 후 [Subscribe] 버튼을 클릭합니다.

29) 올바르게 가동중인 Reader Simulator 를 확인합니다.
30) 시나리오에 따라 올바르게 작동하는 것을 확인합니다.
부록 1. xlogic.xsd

```xml
<?xml version="1.0" encoding="UTF-8"?>
  <xsd:annotation>
    <xsd:documentation>
      XLogic Middleware Platform v1.0 Business Logic Script Schema Definition File
    </xsd:documentation>
  </xsd:annotation>
  <xsd:element name="XLogicScript" type="xlogic:XLogicScript" />
  <xsd:complexType name="XLogicScript">
    <xsd:complexContent>
      <xsd:extension base="xlogic:Statement">
        <xsd:attribute name="name" type="xlogic:Name" use="optional" />
        <xsd:attribute name="creationDate" type="xsd:dateTime" use="optional" />
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="Statement">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="set" type="xlogic:Set" />
      <xsd:element name="wait" type="xlogic:Wait" />
      <xsd:element name="print" type="xlogic:Expr" />
      <xsd:element name="if" type="xlogic:If" />
      <xsd:element name="while" type="xlogic:While" />
      <xsd:element name="foreach" type="xlogic:ForEach" />
      <xsd:element name="invokeWebService" type="xlogic:invokeWebService" />
      <xsd:element name="break" type="xlogic:Break" minOccurs="0" maxOccurs="1" />
      <xsd:element name="continue" type="xlogic:Continue" minOccurs="0" maxOccurs="1" />
    </xsd:choice>
  </xsd:complexType>
  <xsd:complexType name="Set">
    <xsd:simpleContent>
      <xsd:extension base="xlogic:Expr">
        <xsd:attribute name="name" type="xlogic:Name" use="required" />
        <xsd:attribute name="select" type="xsd:string" use="optional" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="Wait">
    <xsd:simpleContent>
      <xsd:extension base="xlogic:Expr" />
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:complexType name="If">
    <xsd:sequence>
      <xsd:element name="then" type="xlogic:Statement" minOccurs="1" maxOccurs="1" />
      <xsd:element name="else" type="xlogic:Statement" minOccurs="0" maxOccurs="1" />
    </xsd:sequence>
    <xsd:attribute name="condition" type="xlogic:Expr" use="required" />
  </xsd:complexType>
  <xsd:complexType name="While">
    <xsd:complexContent base="xlogic:Statement">
      <xsd:attribute name="condition" type="xlogic:Expr" use="required" />
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="ForEach">
    <xsd:complexContent>
      <xsd:attribute name="condition" type="xlogic:Expr" use="required" />
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

SMU Univ Spoon team
<xsd:extension base="xlogic:Statement">
  <xsd:attribute name="name" type="xlogic:Name" use="required"/>
  <xsd:attribute name="source" type="xlogic:Name" use="required"/>
</xsd:extension>
</xsd:complexType>

<xsd:complexType name="InvokeWebService">
  <xsd:extension base="xlogic:InvokeWebServiceStatement">
    <xsd:attribute name="name" type="xlogic:Name" use="required"/>
    <xsd:attribute name="url" type="xsd:anyURI" use="required"/>
    <xsd:attribute name="service" type="xsd:string" use="required"/>
    <xsd:attribute name="port" type="xsd:string" use="optional"/>
    <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
    <xsd:attribute name="async" type="xsd:boolean" default="false" use="optional"/>
  </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="InvokeWebServiceStatement">
  <xsd:choice>
    <xsd:element name="iterator" type="xlogic:Iterator" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:any processContents="lax" namespace="#other" minOccurs="1" maxOccurs="unbounded"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="Iterator">
  <xsd:extension base="xlogic:InvokeWebServiceStatement">
    <xsd:attribute name="name" type="xlogic:Name" use="required"/>
    <xsd:attribute name="source" type="xlogic:Name" use="required"/>
  </xsd:extension>
</xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="Break">
  <xsd:sequence/>
</xsd:complexType>

<xsd:complexType name="Continue">
  <xsd:sequence/>
</xsd:complexType>

<xsd:simpleType name="Name">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[\_a-zA-Z0-9]+"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Expr">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[$\[\]{}|\&lt;\&gt;=;]+"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:schema>
부록 2. Selecting XML with XQuery and XPath()

You can use XPath and XQuery to retrieve specific pieces of XML as you might retrieve data from a database. XQuery and XPath provide a syntax for specifying which elements and attributes you're interested in. The XMLBeans API provides two methods for executing XQuery and XPath expressions, and two ways to use them. The methods are `selectPath` for XPath and `execQuery` for XQuery.

You can call them from an `XmlObject` instance (or a generated type inheriting from it) or an `XmlCursor` instance. As noted below, each of the four methods works slightly differently; be sure to keep these differences in mind when choosing your approach.

**Note:** Both XQuery and complex XPath expressions require additional classes on the class path, as noted in the sections that follow. Also, be sure to see the XMLBeans installation instructions.

**Using XPath with the selectPath Method**

You can execute XPath expressions use the `selectPath` method. When you use XPath with the `selectPath` method, the value returned is view of values from the *current document* — not a copy of those values. In other words, changes your code makes to XML returned by the `selectPath` method change the XML in the document queried against. In contrast, with XQuery executed using the `execQuery` method, the value returned is a *copy of values in the XML queried against*.

Note that XPath itself does not provide syntax for declaring prefix to URI bindings. For user convenience, we allow XQuery syntax to be used for such purposes. You can consult the latest XQuery draft when using syntax for declaring namespaces.

**Note:** By default, XMLBeans supports only very simple XPath expressions. To execute complex expressions — such as those with predicates, function calls, and the like — you will need `xbean_xpath.jar` on your class path. This JAR is among those created when you build XMLBeans from source.

**Calling XmlObject.selectPath**

When called from `XmlObject` (or a type that inherits from it), the `selectPath` method returns an array of objects. If the expression is executed against types generated from schema, then the type for the returned array is one of the Java types corresponding to the schema, and you can cast it accordingly.

For example, imagine you have the following XML containing employee information. You've compiled the schema describing this XML and the types generated from schema are available to your code.

```xml
<xs:employees xmlns:xq="http://xmlbeans.apache.org/samples/xquery/employees">
```
If you wanted to find the phone numbers whose area code was 206, you could capture the XPath expression in this way:

String queryExpression =
 "declare namespace xq='http://xmlbeans.apache.org/samples/xquery/employees';" +
 "$this/xq:employees/xq:employee/xq:phone[contains(., '(206)')];":

Notice in the query expression that the variable $this represents the current context node (the XmlObject that you are querying from). In this example you are querying from the document level XmlObject.

You could then print the results with code such as the following:

// Retrieve the matching phone elements and assign the results to the corresponding // generated type.
PhoneType[] phones = (PhoneType[]) empDoc.selectPath(queryExpression);

// Loop through the results, printing the value of the phone element.
for (int i = 0; i < phones.length; i++)
{
    System.out.println(phones[i].stringValue());
}

**Calling XmlCursor.selectPath**

When called from an XmlCursor instance, the selectPath method retrieves a list of selections, or locations in the XML. The selections are remembered by the cursor instance. You can use methods such as toNextSelection to navigate among them.

The selectPath method takes an XPath expression. If the expression returns any results, each of those results is added as a selection to the cursor’s list of selections. You can move through these selections in the way you might use java.util.Iterator methods to move through a collection.

For example, for a path such as $this/employees/employee, the cursor instance from which you called selectPath would include a selection for each employee element found by the expression. Note that the variable $this is always bound to the current context node,
which in this example is the document. After calling the selectPath method, you would use various "selection"-related methods to work with the results. These methods include:

getSelectionCount() to retrieve the number of selections resulting from the query.

toNextSelection() to move the cursor to the next selection in the list (such as to the one pointing at the next employee element found).

toSelection(int) to move the cursor to the selection at the specified index (such as to the third employee element in the selection).

hasNextSelection() to find out if there are more selections after the cursor's current position.

clearSelections() clears the selections from the current cursor. This doesn't modify the document (in other words, it doesn't delete the selected XML): it merely clears the selection list so that the cursor is no longer keeping track of those positions.

The following example shows how you might use selectPath, in combination with the push and pop methods, to maneuver through XML, retrieving specific values.

```java
public void printZipsAndWorkPhones(XmlObject xml) {
    // Declare the namespace that will be used.
    String xqNamespace = "declare namespace xq='http://xmlbeans.apache.org/samples/xquery/employees':";

    // Insert a cursor and move it to the first element.
    XmlCursor cursor = xml.newCursor();
    cursor.toFirstChild();

    // Save the cursor's current location by pushing it onto a stack of saved locations.
    cursor.push();

    // Query for zip elements.
    cursor.selectPath(xqNamespace + "$this//xq:zip");

    // Loop through the list of selections, getting the value of each element.
    while (cursor.toNextSelection())
    {
        System.out.println(cursor.getTextValue());
    }

    // Pop the saved location off the stack.
    cursor.pop();

    // Query again from the top, this time for work phone numbers.
    cursor.selectPath(xqNamespace + "$this//xq:phone[@location='work']");

    // Move the cursor to the first selection, then print that element's value.
    cursor.toNextSelection();
    System.out.println(cursor.getTextValue());

    // Dispose of the cursor.
    cursor.dispose();
}
```
Using selections is somewhat like tracking the locations of multiple cursors with a single cursor. This becomes especially clear when you remove the XML associated with a selection. When you do so the selection itself remains at the location where the removed XML was, but now the selection's location is immediately before the XML that was after the XML you removed. In other words, removing XML created a kind of vacuum that was filled by the XML after it, which shifted up into the space — up into position immediately after the selection location. This is exactly the same as if the selection had been another cursor.

Finally, when using selections keep in mind that the list of selections is in a sense "live". The cursor you're working with is keeping track of the selections in the list. In other words, be sure to call the clearSelections method when you're finished with the selections, just as you should call the XmlCursor.dispose() method when you're finished using the cursor.

### Using XQuery with the execQuery Method

You use the execQuery method to execute XQuery expressions. With XQuery expressions, XML returned is a copy of XML in the document queried against. In other words, changes your code makes to the values returned by execQuery are not reflected in the document queried against.

**Note:** To execute XQuery expressions, you must have the Saxon 8.1.1 JAR on your class path. Look for the download at the Saxon web site. This JAR is also included in the lib directory when you build XMLBeans from source.

**Calling XmlObject.execQuery**

As with selectPath, calling execQuery from an XmlObject instance will return an XmlObject array.

The following example retrieves work `<zip>` elements from the incoming XML, adding the elements as children to a new `<zip-list>` element.

```java
public boolean collectZips(XmlObject empDoc)
{
    String namespaceDeclaration =
        // The query is designed to return results, so return
    String queryExpression =
        "let $e := $this/xq:employees " +
        "return " +
        "<zip-list> " +
        "{for $z in $e/xq:employee/xq:address/xq:zip " +
        "return $z} " +
        "</zip-list>":

    // Execute the query. Results will be copies of the XML queried against,
    // stored as members of an XmlObject array.
    XmlObject[] results = empDoc.execQuery(namespaceDeclaration + queryExpression);

    // Print the results.
}
if (results.length > 0)
{
    hasResults = true;
    System.out.println("The query results: \n");
    System.out.println(results[0].toString() + "\n");
}
return hasResults;

Calling XmlDocument.execQuery

Unlike the selectPath method called from a cursor, the execQuery method doesn't return void. Instead it returns an XmlDocument instance positioned at the beginning of a new XML document representing the query results. Rather than accessing results as selections, you use the cursor to move through the results in typical cursor fashion (for more information, see Navigating XML with Cursors). The models are very different.

As always, you can cast the results to a type generated from schema if you know that the results conform to that type.

The following example retrieves work <phone> elements from the incoming XML, then changes the number in the results.

```java
public boolean updateWorkPhone(XmlObject empDoc)
{
    boolean hasResults = false;
    // A cursor instance to query with.
    XmlDocument empCursor = empDoc.newCursor();
    empCursor.toNextToken();
    // The expression: Get the employees elements whose
    // value is "WA".
    String queryExpression =
        "for $e in $this/xq:employees/xq:employee " +
        "let $s := $e/xq:address/xq:state " +
        "where $s = 'WA' " +
        "return $e/xq:phone[@location='work']/";
    // Execute the query. Results, if any, will be available at
    // the position of the resultCursor in a new XML document.
    XmlDocument resultCursor =
        empCursor.execQuery(namespaceDeclaration + queryExpression);
    System.out.println("The query results, element copies made " +
        "from the received document: \n");
    System.out.println(resultCursor.getObject().toString() + "\n");
    if (resultCursor.toFirstChild())
    {
        hasResults = true;
        // Use the cursor to loop through the results, printing each sibling
        // element returned by the query.
        int i = 0;
        do
        {
            // Change the phone numbers.
            XmlDocument editCursor = resultCursor.newCursor();
```
editCursor.toLastAttribute();
editCursor.toNextToken();
editCursor.removeXml();
editCursor.insertChars("(206)555-1234");
) while (resultCursor.toNextSibling());

resultCursor.toStartDoc();
System.out.println("The query results after changes: \n");
System.out.println(resultCursor.getObject().toString() + "\n");

System.out.println("The received document -- note that it is unchanged. " +
"Changes were made to the copy created by the execQuery method. \n");
System.out.println(empDoc + "\n");
}
return hasResults;
}

[참고] http://xmlbeans.apache.org/docs/2.0.0/guide/conSelectingXMLwithXQueryPathXPath.html
## Language Elements

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comments</td>
<td>Specified using <code>##</code> and extend to the end of line, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>## This is a comment</code></td>
</tr>
<tr>
<td>Identifiers / variables</td>
<td>Must start with <code>a-z</code>, <code>A-Z</code>, <code>_</code> or <code>$</code>. Can then be followed by <code>0-9</code>, <code>a-z</code>, <code>A-Z</code>, <code>_</code> or <code>$</code>. e.g.</td>
</tr>
<tr>
<td></td>
<td>- Valid: <code>var1_a99,$1</code></td>
</tr>
<tr>
<td></td>
<td>- Invalid: <code>9v_a99,$1</code></td>
</tr>
<tr>
<td>JEXL also supports ant-style variables, e.g.</td>
<td>my.dotted.var is a valid variable name.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE:</strong> JEXL does not support variables with hyphens in them, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>commons-logging</code></td>
</tr>
<tr>
<td></td>
<td>is not a valid variable, but instead is treated as a subtraction of the variable <code>logging</code> from the variable <code>commons</code></td>
</tr>
<tr>
<td>Scripts</td>
<td>A script in Jexl is made up of zero or more statements. Scripts can be read from a String, File or URL.</td>
</tr>
<tr>
<td>Statements</td>
<td>A statement can be the empty statement, the semicolon (<code>;</code>), block, assignment or an expression. Statements are optionally terminated with a semicolon.</td>
</tr>
<tr>
<td>Block</td>
<td>A block is simply multiple statements inside curly braces (<code>{,}</code>).</td>
</tr>
</tbody>
</table>

### Literals

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer Literals</td>
<td>1 or more digits from <code>0</code> to <code>9</code></td>
</tr>
<tr>
<td>Floating point Literals</td>
<td>1 or more digits from <code>0</code> to <code>9</code>, followed by a decimal point and then one or more digits from <code>0</code> to <code>9</code>.</td>
</tr>
<tr>
<td>String literals</td>
<td>Can start and end with either <code>'</code> or <code>&quot;</code>, e.g.</td>
</tr>
<tr>
<td></td>
<td>- &quot;Hello world&quot;</td>
</tr>
<tr>
<td></td>
<td>and</td>
</tr>
<tr>
<td></td>
<td>- 'Hello world'</td>
</tr>
<tr>
<td></td>
<td>are equivalent.</td>
</tr>
<tr>
<td>Boolean literals</td>
<td>The literals <code>true</code> and <code>false</code> can be used, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>vall == true</code></td>
</tr>
<tr>
<td>Null literal</td>
<td>The null value is represented as in java using the literal <code>null</code>, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>vall == null</code></td>
</tr>
<tr>
<td>Map literal</td>
<td>You can construct a <code>java.util.HashMap</code> by using</td>
</tr>
<tr>
<td></td>
<td>[ key =&gt; value, key =&gt; value ]</td>
</tr>
<tr>
<td></td>
<td>.key and value can be any valid expression.</td>
</tr>
<tr>
<td>Array literal</td>
<td>You can construct an <code>Object[]</code> array by using</td>
</tr>
<tr>
<td></td>
<td>[ item, item, item ]</td>
</tr>
<tr>
<td></td>
<td>.item can be any valid expression.</td>
</tr>
</tbody>
</table>

### Functions

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>empty</td>
<td>Returns true if the expression following is either:</td>
</tr>
<tr>
<td></td>
<td>1. <code>null</code></td>
</tr>
<tr>
<td></td>
<td>2. An empty string</td>
</tr>
<tr>
<td></td>
<td>3. An array of length zero</td>
</tr>
</tbody>
</table>
4. A collection of size zero
5. An empty map

```
empty(var1)
```

<table>
<thead>
<tr>
<th>size</th>
<th>Returns the information about the expression:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Length of an array</td>
</tr>
<tr>
<td>2.</td>
<td>Size of a List</td>
</tr>
<tr>
<td>3.</td>
<td>Size of a Map</td>
</tr>
<tr>
<td>4.</td>
<td>Size of a Set</td>
</tr>
<tr>
<td>5.</td>
<td>Length of a string</td>
</tr>
</tbody>
</table>

```
size("Hello") returns 5.
```

### Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td><strong>and</strong> The usual <code>&amp;</code> operator can be used as well as the word <code>and</code>, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>cond1 and cond2</code></td>
</tr>
<tr>
<td></td>
<td><code>cond1 &amp;&amp; cond2</code></td>
</tr>
<tr>
<td></td>
<td>are equivalent</td>
</tr>
<tr>
<td>Boolean</td>
<td><strong>or</strong> The usual `</td>
</tr>
<tr>
<td></td>
<td><code>cond1 or cond2</code></td>
</tr>
<tr>
<td></td>
<td>`cond1</td>
</tr>
<tr>
<td></td>
<td>are equivalent</td>
</tr>
<tr>
<td>Boolean</td>
<td><strong>not</strong> The usual <code>!</code> operator can be used as well as the word <code>not</code>, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>!cond1</code></td>
</tr>
<tr>
<td></td>
<td><code>not cond1</code></td>
</tr>
<tr>
<td></td>
<td>are equivalent</td>
</tr>
<tr>
<td>Bitwise</td>
<td><strong>and</strong> The usual <code>&amp;</code> operator is used, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>33 &amp; 4</code></td>
</tr>
<tr>
<td></td>
<td>, <code>0010 0001 &amp; 0000 0100 = 0.</code></td>
</tr>
<tr>
<td>Bitwise</td>
<td><strong>or</strong> The usual `</td>
</tr>
<tr>
<td></td>
<td>`33</td>
</tr>
<tr>
<td></td>
<td>, `0010 0001</td>
</tr>
<tr>
<td>Bitwise</td>
<td><strong>xor</strong> The usual <code>^</code> operator is used, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>33 ^ 4</code></td>
</tr>
<tr>
<td></td>
<td>, <code>0010 0001 ^ 0000 0100 = 0010 0100 = 37.</code></td>
</tr>
<tr>
<td>Bitwise</td>
<td><strong>complement</strong> The usual <code>~</code> operator is used, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>~33</code></td>
</tr>
<tr>
<td></td>
<td>, <code>~0010 0001 = 1101 1110 = -34.</code></td>
</tr>
<tr>
<td>Equality</td>
<td><strong>==</strong> operator can be used as well as the abbreviation <code>eq</code>. For example</td>
</tr>
<tr>
<td></td>
<td><code>val1 == val2</code></td>
</tr>
<tr>
<td></td>
<td><code>val1 eq val2</code></td>
</tr>
<tr>
<td></td>
<td>are equivalent</td>
</tr>
<tr>
<td></td>
<td>1. null is only ever equal to null, that is if you compare null to any</td>
</tr>
<tr>
<td></td>
<td>non-null value, the result is false.</td>
</tr>
<tr>
<td></td>
<td>2. Equality uses the java <code>equals</code> method</td>
</tr>
<tr>
<td>Inequality</td>
<td><strong>!=</strong> operator can be used as well as the abbreviation <code>ne</code>. For example</td>
</tr>
<tr>
<td></td>
<td><code>val1 != val2</code></td>
</tr>
<tr>
<td></td>
<td>and</td>
</tr>
<tr>
<td>Operator</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><code>val1 ne val2</code></td>
<td>are equivalent.</td>
</tr>
<tr>
<td><strong>Less Than</strong></td>
<td>The usual <code>&lt;</code> operator can be used as well as the abbreviation <code>lt</code>. For example</td>
</tr>
<tr>
<td></td>
<td><code>val1 &lt; val2</code></td>
</tr>
<tr>
<td></td>
<td>and <code>val1 lt val2</code></td>
</tr>
<tr>
<td></td>
<td>are equivalent.</td>
</tr>
<tr>
<td><strong>Less Than Or Equal To</strong></td>
<td>The usual <code>&lt;=</code> operator can be used as well as the abbreviation <code>le</code>. For example</td>
</tr>
<tr>
<td></td>
<td><code>val1 &lt;= val2</code></td>
</tr>
<tr>
<td></td>
<td>and <code>val1 le val2</code></td>
</tr>
<tr>
<td></td>
<td>are equivalent.</td>
</tr>
<tr>
<td><strong>Greater Than</strong></td>
<td>The usual <code>&gt;</code> operator can be used as well as the abbreviation <code>gt</code>. For example</td>
</tr>
<tr>
<td></td>
<td><code>val1 &gt; val2</code></td>
</tr>
<tr>
<td></td>
<td>and <code>val1 gt val2</code></td>
</tr>
<tr>
<td></td>
<td>are equivalent.</td>
</tr>
<tr>
<td><strong>Greater Than Or Equal To</strong></td>
<td>The usual <code>&gt;=</code> operator can be used as well as the abbreviation <code>ge</code>. For example</td>
</tr>
<tr>
<td></td>
<td><code>val1 &gt;= val2</code></td>
</tr>
<tr>
<td></td>
<td>and <code>val1 ge val2</code></td>
</tr>
<tr>
<td></td>
<td>are equivalent.</td>
</tr>
<tr>
<td><strong>Addition</strong></td>
<td>The usual <code>+</code> operator is used. For example <code>val1 + val2</code></td>
</tr>
<tr>
<td><strong>Subtraction</strong></td>
<td>The usual <code>-</code> operator is used. For example <code>val1 - val2</code></td>
</tr>
<tr>
<td><strong>Multiplication</strong></td>
<td>The usual <code>*</code> operator is used. For example <code>val1 * val2</code></td>
</tr>
<tr>
<td><strong>Division</strong></td>
<td>The usual <code>/</code> operator is used. For example <code>val1 / val2</code></td>
</tr>
<tr>
<td><strong>Integer Division</strong></td>
<td>The <code>div</code> operator is used. For example <code>4 div 3</code> gives 1.</td>
</tr>
<tr>
<td><strong>Modulus (or remainder)</strong></td>
<td>The <code>%</code> operator is used. An alternative is the <code>mod</code> operator. For example</td>
</tr>
<tr>
<td></td>
<td><code>5 mod 2</code></td>
</tr>
<tr>
<td></td>
<td>gives 1 and is equivalent to <code>5 % 2</code></td>
</tr>
<tr>
<td><strong>Negation</strong></td>
<td>The unary <code>-</code> operator is used. For example <code>-12</code></td>
</tr>
<tr>
<td><strong>Array access</strong></td>
<td>Array elements may be accessed using either square brackets or a dotted numeral, e.g.</td>
</tr>
<tr>
<td></td>
<td><code>arr1[0]</code></td>
</tr>
<tr>
<td></td>
<td>and <code>arr1.0</code></td>
</tr>
<tr>
<td></td>
<td>are equivalent</td>
</tr>
</tbody>
</table>

**Conditional**

- **if**
  - Classic, if/else statement, e.g. `if ((x * 2) == 5) {y = 1; } else {y = 2; }`

- **foreach**
  - Loop through items of an Array, Collection, Map, Iterator or Enumeration, e.g. `foreach (item in list) { x = x + item; } Where item and list are variables.`

- **while**
  - Loop until a condition is satisfied, e.g. `while (x lt 10) { x = x + 2; }`

부록 4.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xlogic:XLogicScript xmlns:xlogic="urn:spoon:xlogic:script:xsd:1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:spoon:xlogic:script:xsd:1 http://spoon.smu.ac.kr/xlogic.xsd">
    <xlogic:set name="tags" select="/tag">${_REPORTS}</xlogic:set>
    <xlogic:invokeWebService service="TagPrintWebService" port="TagPrintWebServiceHttpPort"
        url="http://localhost:8080/tagprint/services/TagPrintWebService"
        name="result" namespace="http://print">
        <tagPrint xmlns="http://tagprint.webservice.spoon.org">
            <ArrayOfString>
                <xlogic:iterator name="tag" source="tags">
                    <string>${tag}</string>
                </xlogic:iterator>
            </ArrayOfString>
        </tagPrint>
    </xlogic:invokeWebService>
    <xlogic:wait>${5000}</xlogic:wait>
    <xlogic:print>${"Test Script"}</xlogic:print>
    <xlogic:set name="i">${1}</xlogic:set>
    <xlogic:while condition="${i lt 11}">
        <xlogic:print>${i}</xlogic:print>
        <xlogic:set name="i">${i+1}</xlogic:set>
        <xlogic:if condition="${i eq 5}">
            <xlogic:then><xlogic:break></xlogic:break></xlogic:then>
        </xlogic:if>
        <xlogic:if condition="${i eq 2}">
            <xlogic:then><xlogic:print>${"thank you"}</xlogic:print></xlogic:then>
        </xlogic:if>
    </xlogic:while>
    <xlogic:foreach name="tag" source="tags">
        <xlogic:print>${tag}</xlogic:print>
    </xlogic:foreach>
</xlogic:XLogicScript>
```