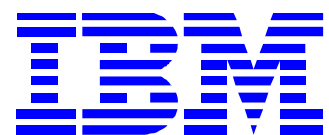


WebSphere eXtreme Scale v8.6

WXS 를 활용한 In-Memory DataGrid 구성 가이드 2

(2015. 01.)

IBM SWG WebSphere



0) IBM WXS 를 활용한 In-Memory DataGrid(IMDG) 구성이란?

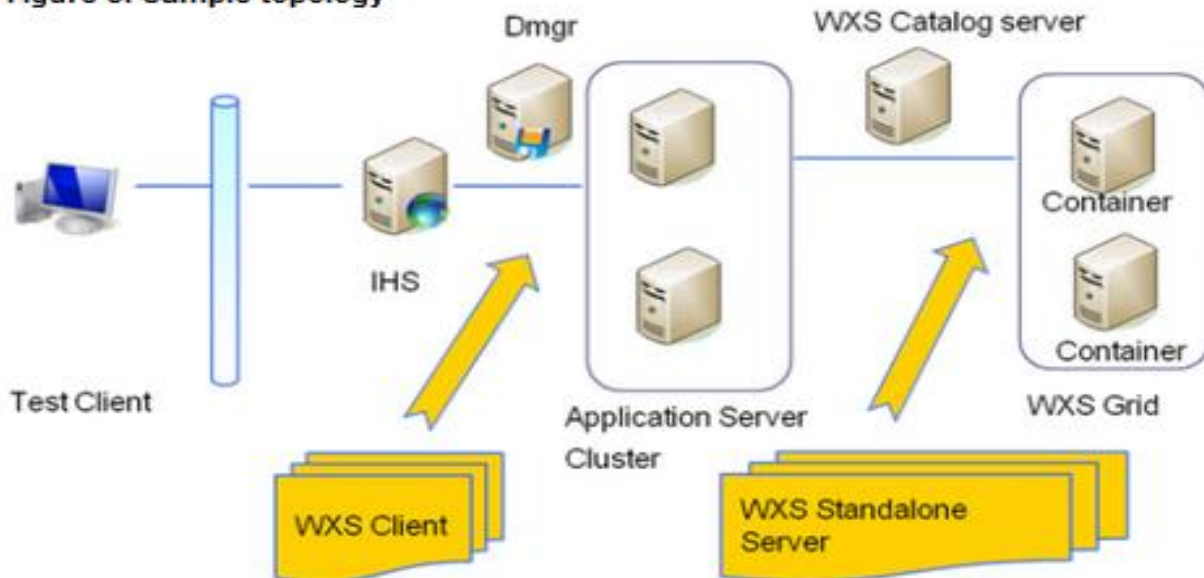
안녕하세요 freeman 입니다.

지난 강좌에서 DB 앞에 WXS 를 In-Memory DataGrid 로 활용하는 간단한 샘플을 살펴보도록 하겠습니다. 특히, 지난 강좌에서는 JPA 라는 Java EE 표준 OR Mapping Framework 를 사용하는 형태로 강좌를 진행하였습니다. 그러나 현재 많은 프로젝트에서는 JPA 라는 OR Mapping 방식보다는 JDBC 를 이용한 방식을 많이 사용하고 있습니다. 그럼 JDBC 를 활용한 DB Call 에서 어떻게 WXS 의 IMDG 를 활용할 수 있는지 살펴보도록 하겠습니다.

1) IBM WebSphere eXtreme Scale(WXS) 설치 및 기본 구성

설치 및 구성의 경우는 당연히 지난 강좌의 구성 형태와 동일하게 WXS 가 Client(WAS)와 Server 로 구분되어 설치 되어있다고 가정을 하고 진행하도록 하겠습니다.

Figure 5. Sample topology



2) 테스트를 위한 DB Table, JDBC 애플리케이션 작성

① JPA 애플리케이션으로 조회, 수정을 수행하기 위한 샘플 DB 와 Table 을 하나 작성

DB Table 은 새롭게 작성하지 않고 지난 강좌에서 사용한 DB Table 을 그대로 활용하도록 하겠습니다.

```
db2 => select * from PERSON
```

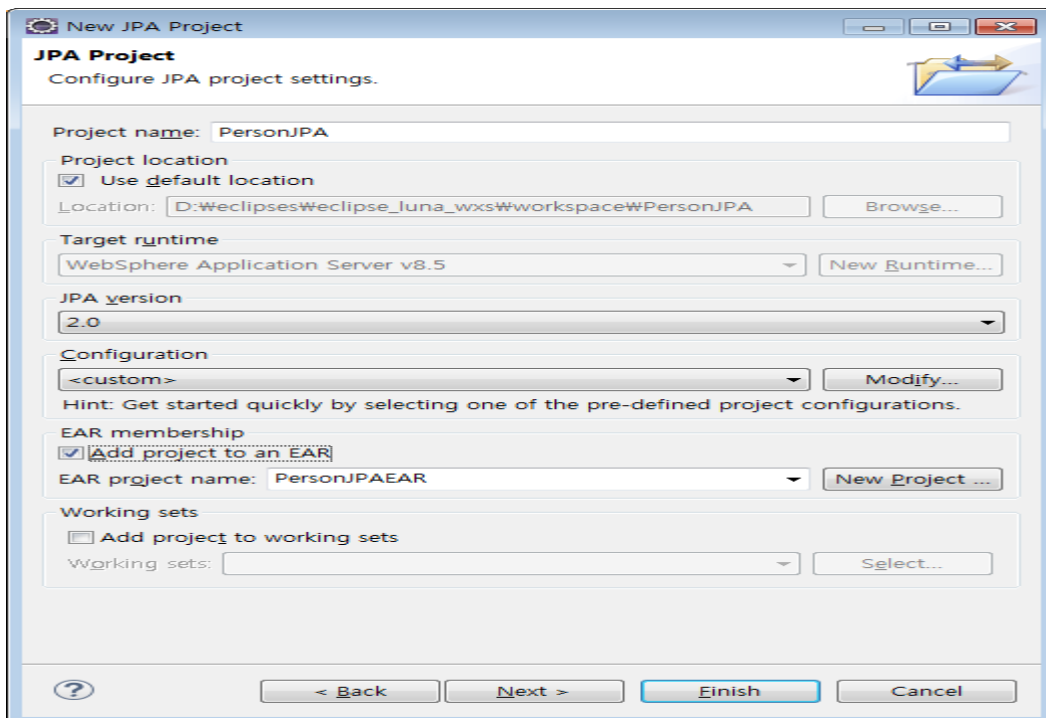
ID	NAME	JOIN_DATE
7	JungWoon Lee	2015-01-12
8	JungWoo Lee	2015-01-05
9	JungWo Lee	2015-01-12
10	JungW Lee	2015-01-06
11	Jung Lee	2015-01-12
12	Jun Lee	2015-01-07

6 레코드가 선택되었습니다.

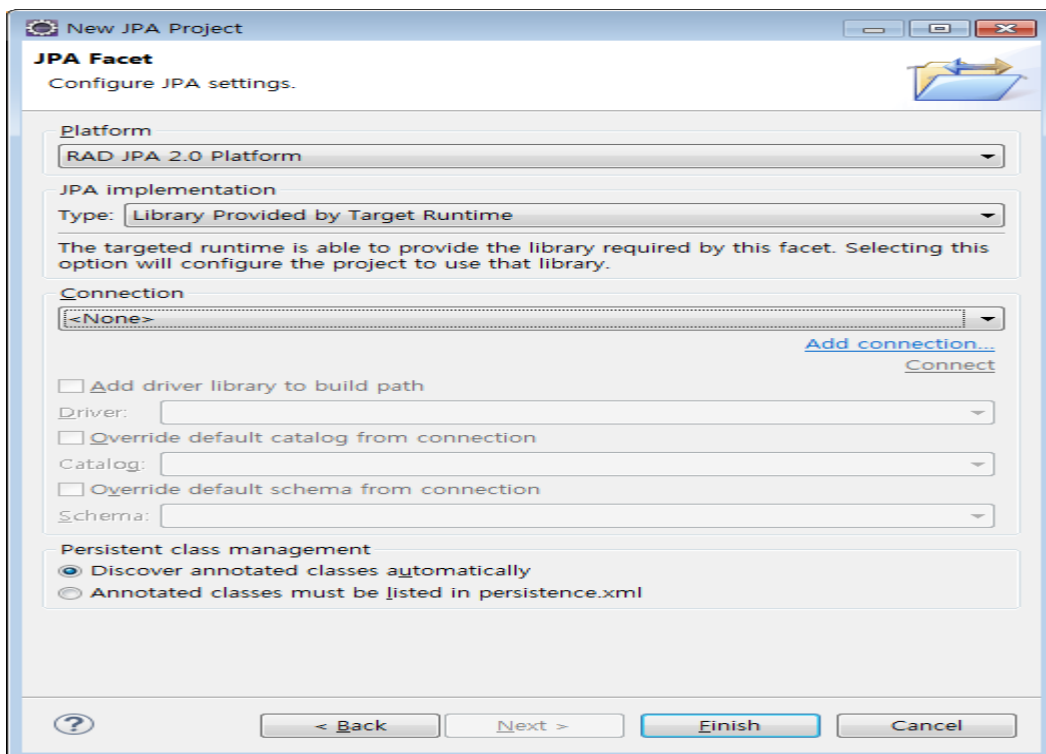
② 조회를 수행하기 위한 간단한 JDBC 애플리케이션 작성

DB 에 임시 데이터를 넣었으면 해당 DB 를 조회할 수 있는 기본 JPA 프로젝트를 작성합니다. 이를 위해서 하단과 같이 eclipse 를 이용해서 JPA 프로젝트를 하나 만듭니다.

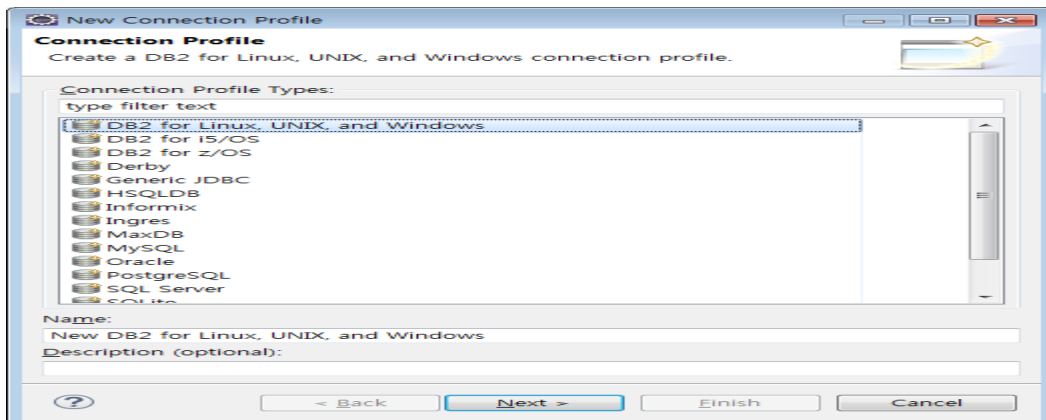
(이때 JPA 버전은 2.0 을 선택하고 WAR 프로젝트를 넣기 위하여 EAR 프로젝트도 같이 생성합니다.)



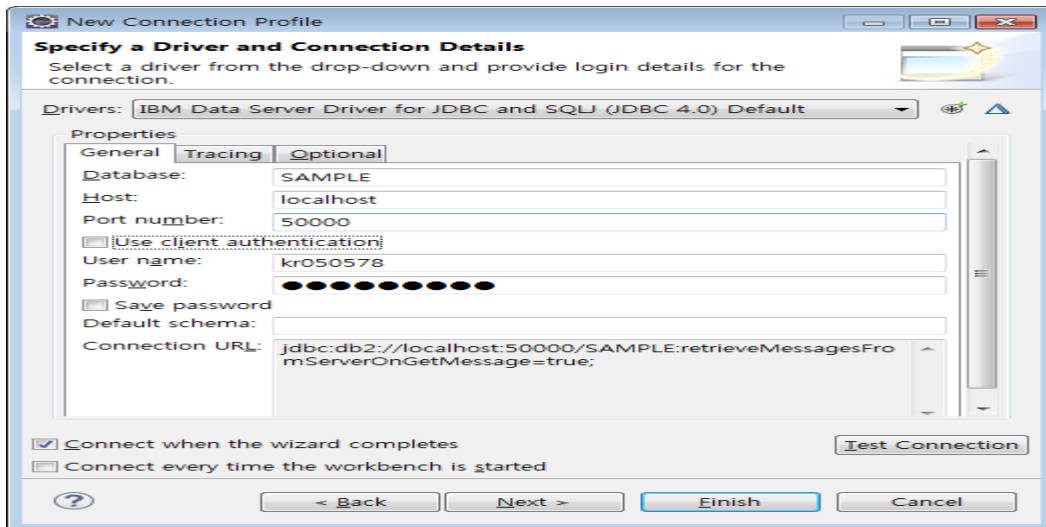
JPA 프로젝트 생성중에 Connection 관련한 마법사가 나오면 DB 에 연결하는 작업을 수행하기 위하여 'Add connection' 버튼을 클릭합니다



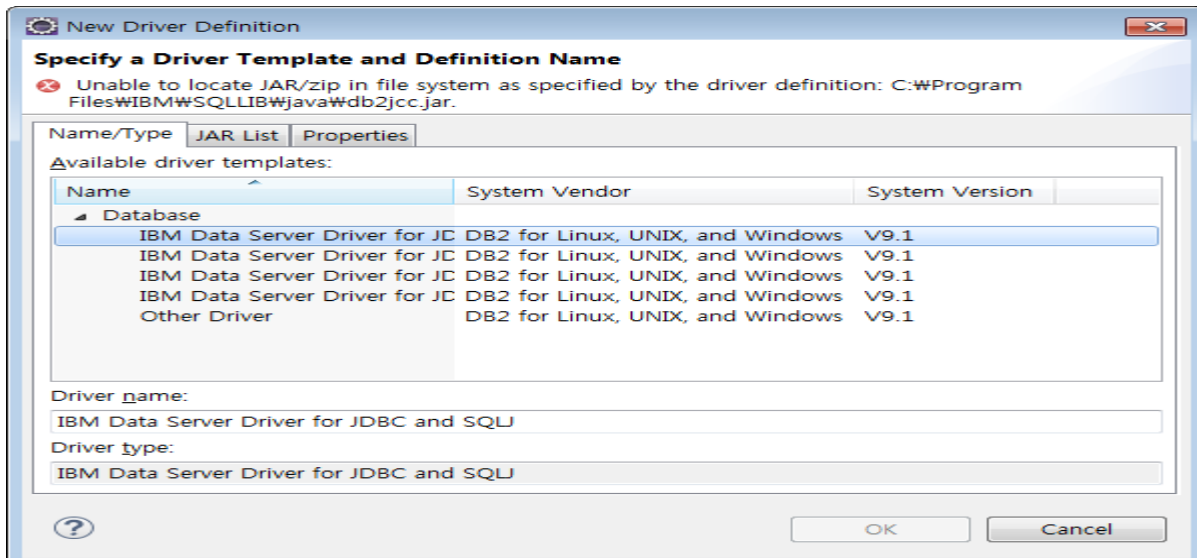
Connection profile 에서는 연결되는 DB를 선택하고 다음을 클릭합니다.



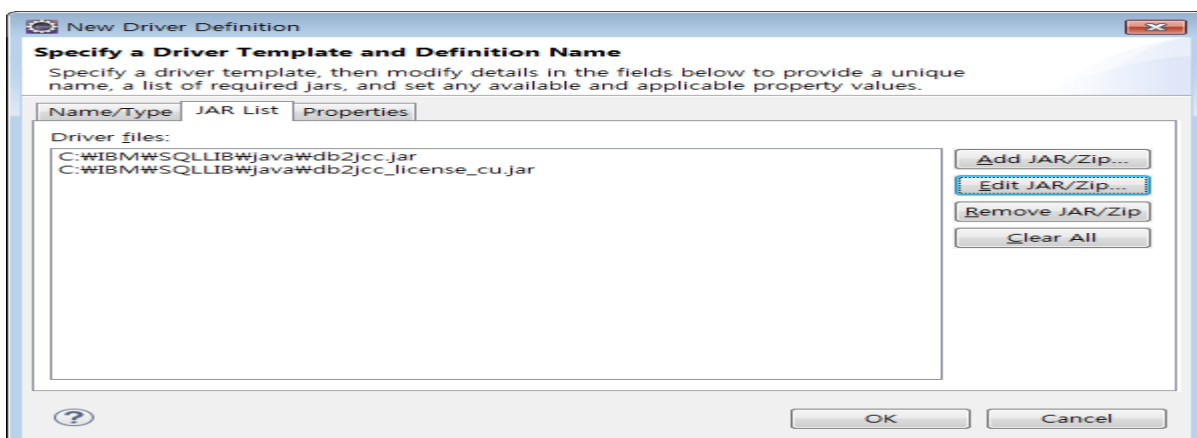
연결을 위한 User/Password 정보를 입력한 후에 Driver 설정을 위하여 Drivers 메뉴 옆의 아이콘을 클릭합니다.



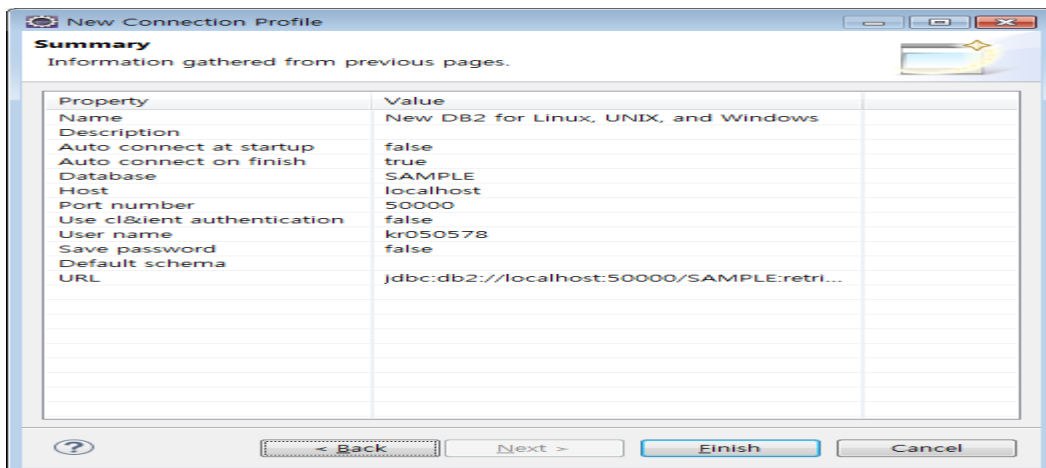
DataBase 기본 정보를 선택한 후에 DB Driver jar 파일을 설정하기 위하여 JAR List 탭을 클릭합니다.



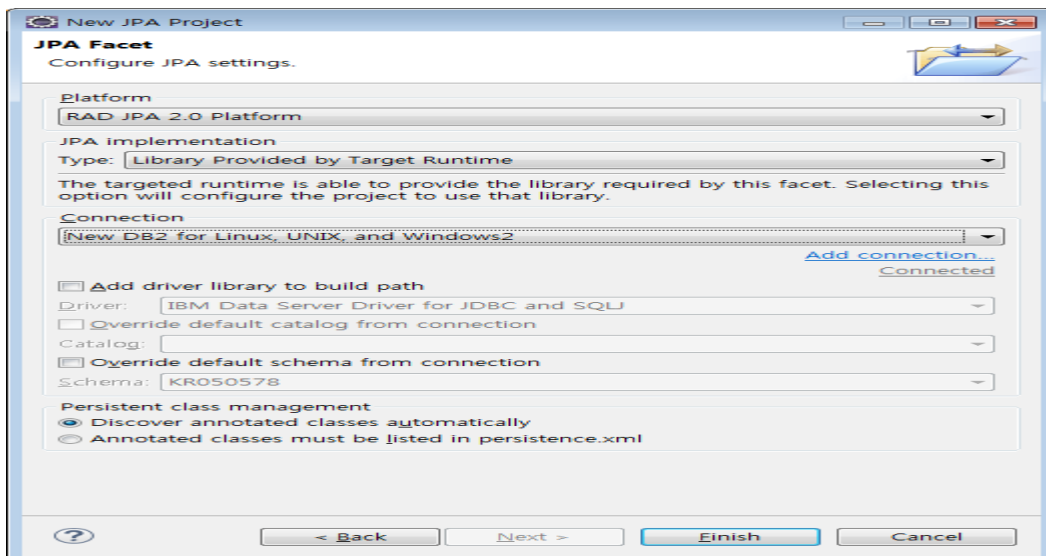
Driver Jar 파일의 위치를 기본이 아니라 가지고 있는 DB Driver 위치로 변경한 후 OK 를 클릭합니다.



요약정보를 확인하고 이상이 없다면 Finish 를 클릭합니다.



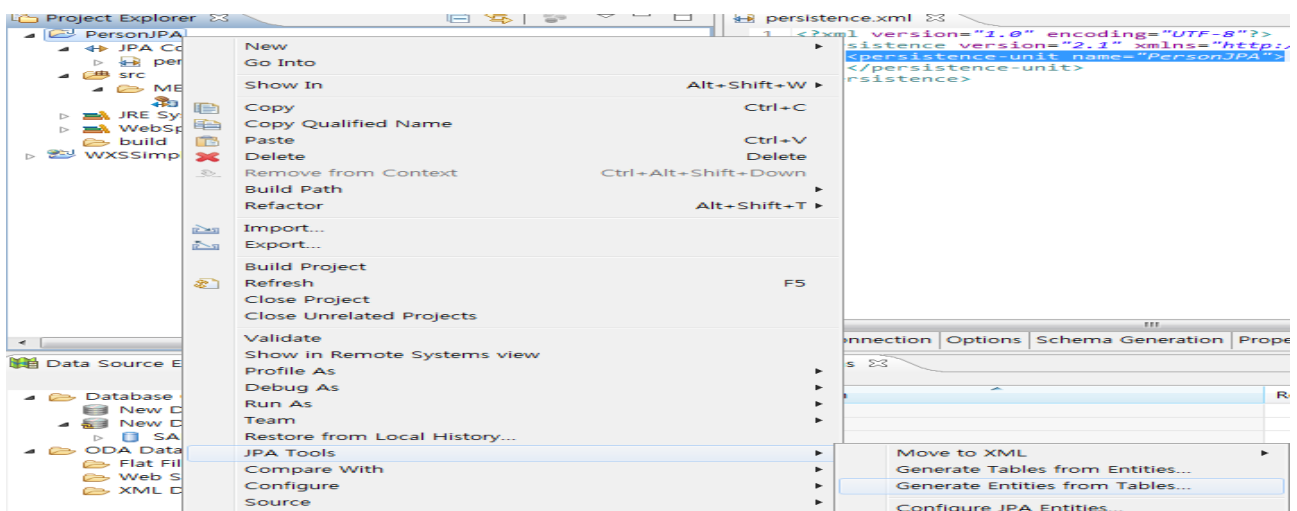
마지막으로 정보를 확인하고 Finish 를 클릭하여 JPA 생성을 완료합니다.



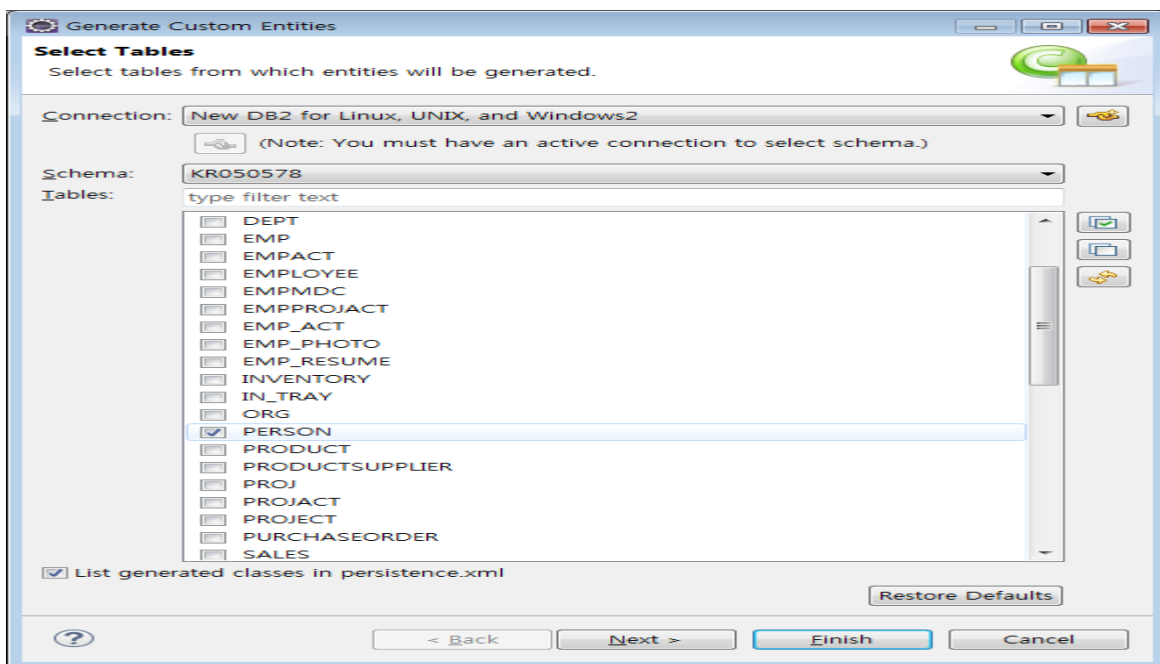
여기까지 작업을 문제없이 수행했다면 하단과 같은 구조로 JPA 프로젝트가 생성된 것을 확인할 수 있습니다.



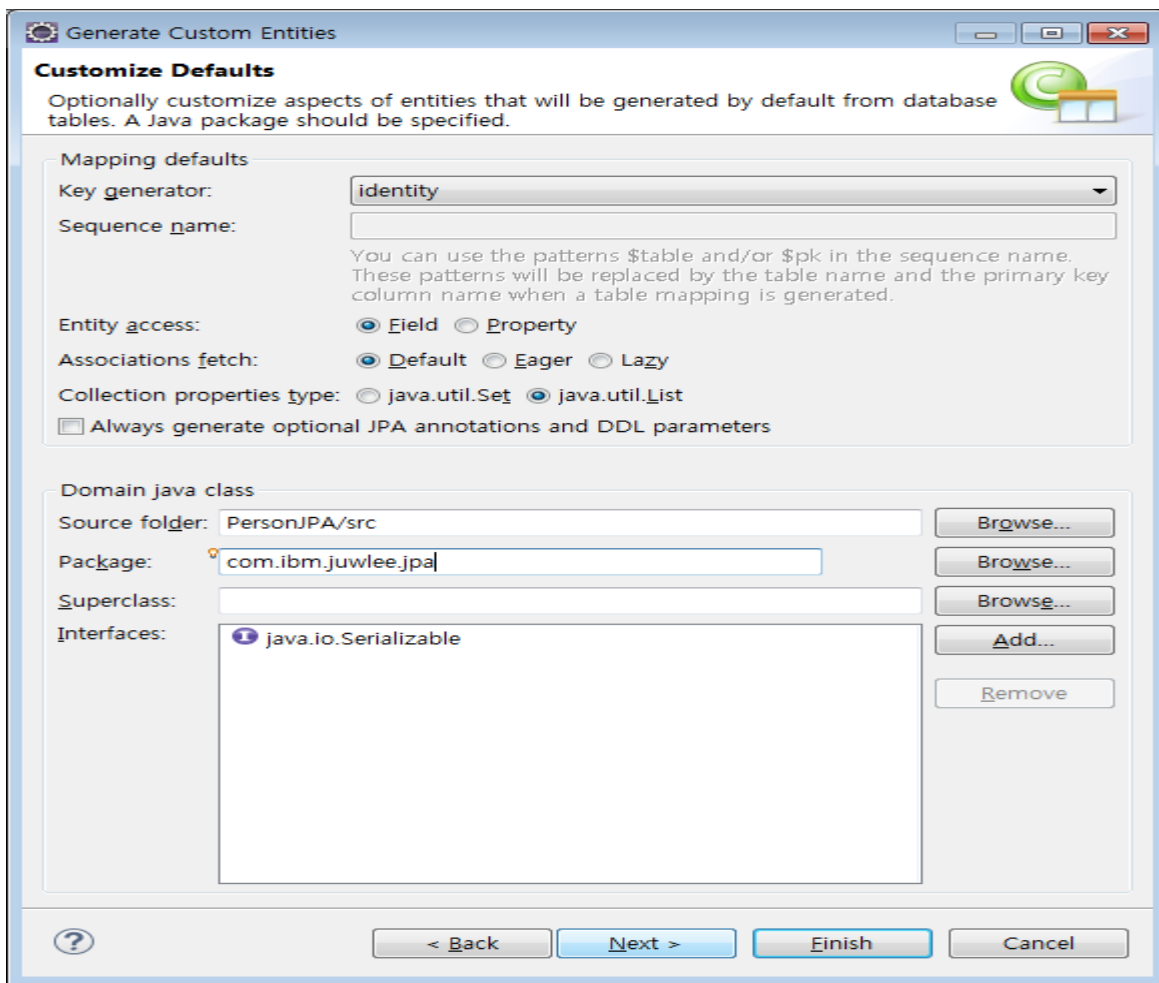
이제 실제로 DB 와 연결된 JPA 소스를 자동으로 생성하기 위하여 프로젝트에서 마우스 우 클릭 후에 JPA Tools > Generate Entities from Tables 를 클릭합니다.



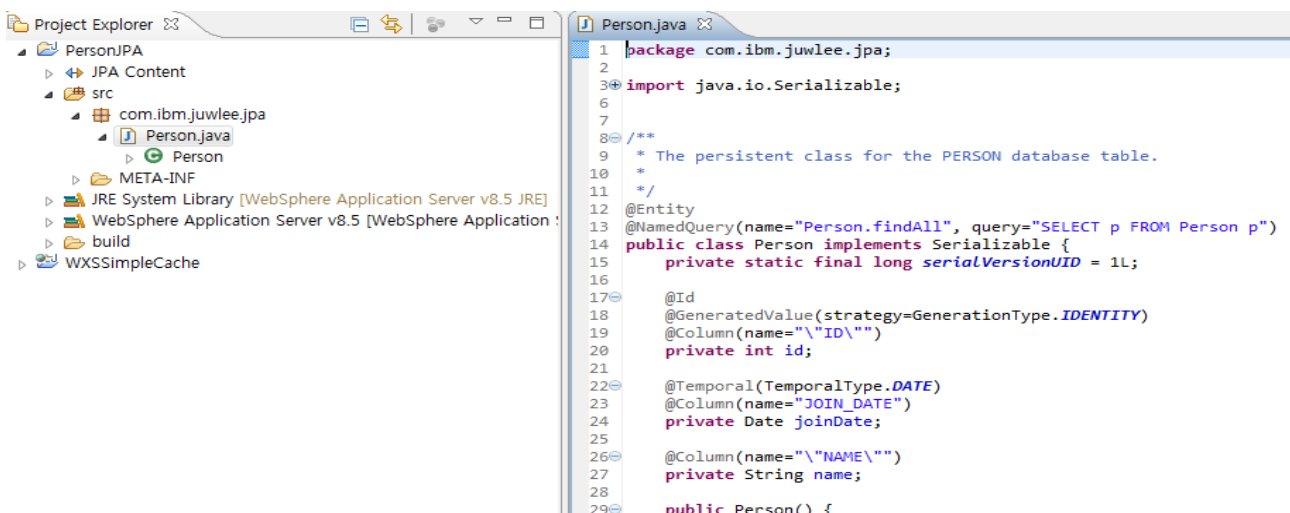
이전에 만들어진 Connection 에서 table 을 선택하는 마법사가 나오면 방금 생성한 Person table 을 선택하고 다음을 클릭합니다.



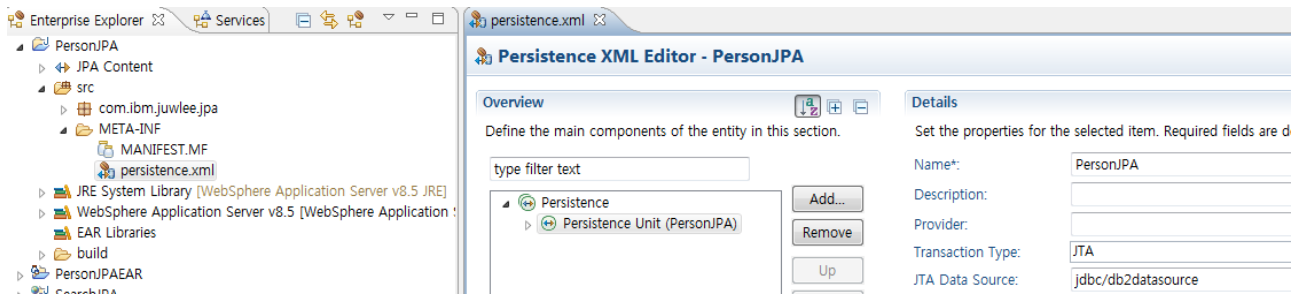
이전에 Person table 을 생성할 때, ID 값을 자동생성 했기 때문에 Key generator 를 identity 로 선택하고 package 명도 입력한 후 완료를 클릭합니다.



해당 작업이 정상적으로 완료되면 하단과 같이 Person 이라는 Entity class 가 Tool 에 의해서 자동으로 생성된 것을 확인할 수 있습니다.



마지막으로 JPA 프로젝트의 설정(persistence.xml) 부분에서 JTA Data Source 명을 입력합니다. (향후 해당 JNDI 이름으로 WAS 에서 JDBC Data source 를 생성해서 사용할 것입니다.)



JPA 프로젝트를 잘 만들었으면 이를 조회하기 위하여 샘플 servlet 를 이용한 동적 웹 프로젝트를 하단과 같이 만들고 이를 IBM WAS 에 배포합니다. 하단의 내용은 Servlet 의 snippet 으로서 전체코드는 첨부된 어플리케이션을 참조하시기 바라며 하단의 snippet 만으로도 JPA 조회 servlet 을 만드는 것은 어렵지 않으실 것 입니다. 첨부한 것을 간단히만 설명드리자면 SearchPersonJPA에서 JPA 호출을 위한 retrievePerson 메소드로서 JPA 방식으로 EntityManagerFactory 를 받아와서 EntityManager 를 생성하고 이를 이용해 DB 의 select 작업(EntityManager 의 find() 메소드)을 진행하는 샘플입니다. 다만 JPA 강좌가 아니기때문에 해당 소스 설명을 일일이 진행하지는 않도록 하겠습니다.

```
@PersistenceUnit(unitName="PersonJPA")
private EntityManagerFactory emf;
@Resource
private javax.transaction.UserTransaction utx;
private String TARGET_SERVLET;

protected Person retrievePerson(int Id) throws Exception {

    EntityManager em = null;
    Person person = null;

    try {
        utx.begin();
        em = emf.createEntityManager();
        person = (Person)em.find(Person.class, Id);
        if (person != null) {
            utx.commit();
        } else {
            utx.rollback();
            throw new Exception("person is not found : " + Id);
        }
    } catch (NotSupportedException | SystemException e) {
        e.printStackTrace();
    } finally {
        em.close();
    }

    return person;
}
```

3) JPA 테스트를 위한 WAS 설정 및 준비

JPA 테스트를 위하여 client 로 사용할 WAS 에 DB 를 연결하기 위한 정보인 JDBC Provider 와 Datasource 를 작성합니다. (이 부분은 다른 강좌에서도 이미 많이 다뤘으므로 다른 기본 강좌를 참고하시기 바라겠습니다.)

kr050578Node04 노드에 있는 server1 서버의 DB2 Universal JDBC Driver DataSource 데이터 소스에 대한 연결 테스트 조작에 성공했습니다.

데이터 소스

선택된 JDBC 제공자에 연결된 데이터 소스의 설정을 편집하려면 이 페이지를 사용하십시오. 데이터 소스 오브젝트는 데이터베이스에 대한 액세스를 위한 연결을 애플리케이션에 제공합니다. [안내된 활동](#) 내 이 태스크에 대한 자세한 내용을 학습하십시오. 안내된 활동에서는 태스크 단계의 목록과 주제에 대한 좀더 일반적인 정보가 제공됩니다.

범위: 셀=**kr050578Node03Cell**, 노드=**kr050578Node04**

범위는 자원 정의를 볼 수 있는 레벨을 지정합니다. 범위 및 작동 방법에 관한 자세한 정보는 [범위 설정 도움말을 참조하십시오](#).

노드=kr050578Node04

환경 설정

<div> <div>새로 작성...</div> <div>삭제</div> <div>연결 테스트</div> <div>상태 관리...</div> </div>					
<div> <div>선택</div> <div>이름</div> <div>JNDI 이름</div> <div>범위</div> <div>제공자</div> <div>설명</div> <div>카테고리</div> </div>					
다음 자원을 관리할 수 있습니다.					
<input type="checkbox"/>	DB2 Universal JDBC Driver DataSource	jdbc/db2datasource	노드=kr050578Node04	DB2 Universal JDBC Driver Provider	DB2 Universal Driver Datasource

이후 이전 파트에서 작성하였던 샘플 애플리케이션을 IBM WAS 로 배포합니다.

WebSphere. software

보기: 모든 태스크

시작

안내된 활동

서버

서버 클러스터

애플리케이션

새 애플리케이션

애플리케이션 목록

WebSphere 엔터프라이즈 애플리케이션

비즈니스 레퍼 애플리케이션

자산

응용 프로그램 배치 설정

서비스

자원

셀=kr050578Node03Cell, 프로파일=AppSrv01

엔터프라이즈 애플리케이션

엔터프라이즈 애플리케이션

설치된 애플리케이션을 관리하려면 이 페이지를 사용하십시오. 단일 애플리케이션을 여러 서버로 배치시킬 수 있습니다.

환경 설정

시작

중지

설치

설치 제거

업데이트

클라우드 업데이트

파일 제거

내보내기

DDL 내보내기

선택

이름

애플리케이션 상태

다음 자원을 관리할 수 있습니다.

☐

DefaultApplication

➡

☐

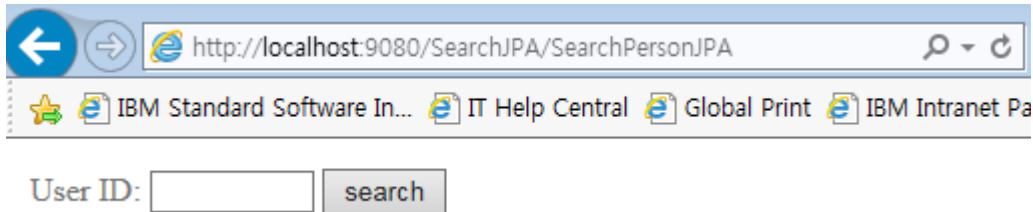
PersonJPAEAR

➡

4) 샘플 JPA 테스트

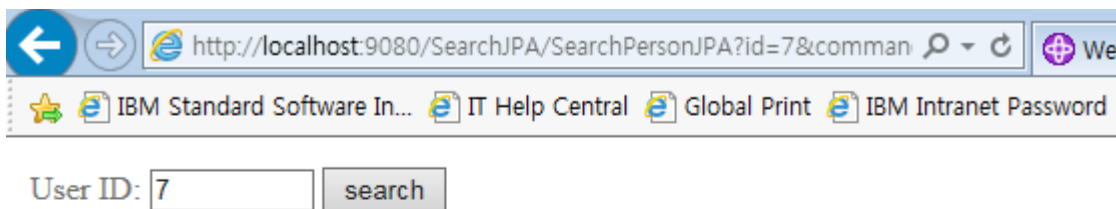
이제 JPA 테스트를 위한 준비가 다 된 것이므로 샘플 테스트를 수행해보도록 하겠습니다. 먼저 배포된 애플리케이션의 만들어진 서블릿 페이지를 호출합니다.

(<http://localhost:9080/SerchJPA/SerchPersonJPA>)



Search form showing the URL <http://localhost:9080/SerchJPA/SearchPersonJPA>. The form includes a 'User ID' input field and a 'search' button.

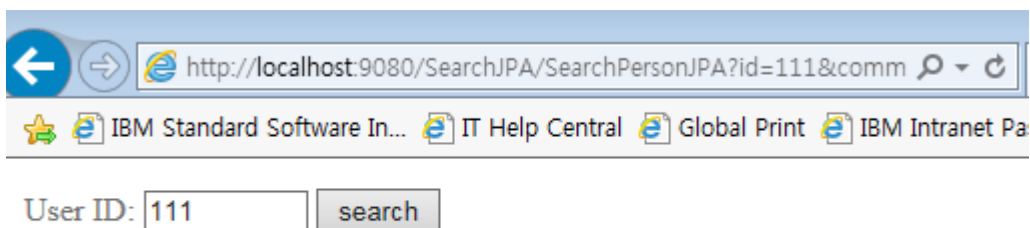
호출이 정상적으로 완료되면 Sample DB 에 넣어둔 data 를 하나 조회합니다.



Search form showing the URL <http://localhost:9080/SerchJPA/SearchPersonJPA?id=7&command=search>. The 'User ID' input field contains the value '7'.

ID:	7
Name:	JungWoon Lee
Date Joined:	Mon Jan 12 00:00:00 KST 2015
Spend Time:	479.449

Sample DB 에 없는 ID 도 테스트로 조회해 봅니다.



Search form showing the URL <http://localhost:9080/SerchJPA/SearchPersonJPA?id=111&command=search>. The 'User ID' input field contains the value '111'.

person is not found : 111

여기까지 동일한 결과를 확인하셨다면 문제없이 JPA 테스트가 완료된 것입니다.

5) IBM WXS 테스트를 위한 JPA 애플리케이션 변경

이전 파트에서 JPA 애플리케이션을 작성해서 간단하게 DB 조회하는 샘플을 만들어서 테스트해봤습니다. 이번에는 동일한 애플리케이션을 활용해서 WXS 의 Data Grid 를 통해서 Data 를 조회하고 없을 경우에 DB 를 조회하여 Data 를 Data Grid 에 자동으로 넣는 애플리케이션을 작성해 보도록 하겠습니다.

하단에 snippet 을 보시면 아시겠지만 이전에 진행한 WXS 를 활용한 글로벌 캐시 구성과 거의 동일하며 다만 String 을 반환받는 것이 아니라 JPA 기반으로 되어 있는 entity class 인 Person 을 리턴 받는 형태만 조금 다릅니다.

```
private String mapName = "Person";
@PersistenceUnit(unitName="PersonJPA")
private EntityManagerFactory emf;
@Resource
private javax.transaction.UserTransaction utx;
private String TARGET_SERVLET;

private ObjectGrid og=null;

protected Person retrievePerson(int Id) throws Exception {
    Person person = null;
    Session session = null;
    try {
        // Get a session
        session = og.getSession();
        // Get the ObjectMap
        ObjectMap map1 = session.getMap(mapName);
        //transaction related with session begin
        session.begin();
        // Insert data into the map.This is implicitly transactional
        person = (Person)map1.get(Id);
        // Close the session so that it can be managed in a pool
        session.commit();
    } catch (ObjectGridException e) {
        session.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
    return person;
}
```

DB 데이터를 자동으로 캐시하고 조회하는 WXS 를 활용한 In-Memory DataGrid 구성을 위하여 Sample DB 를 활용하여 새롭게 table

6) IBM WXS 테스트를 위한 WXS Server 설정

이전 파트에서 JPA 애플리케이션을 작성해서 간단하게 DB 조회하는 샘플을 만들어서 테스트해봤습니다. 이번에는 동일한 애플리케이션을 활용해서 WXS 의 Data Grid 를 통해서 Data 를 조회하고 없을 경우에 DB 를 조회하여

카탈로그 서버

```
startXsServer cat1 -listenerPort 4809 -catalogServiceEndPoints cat1:kr050578:6600:6601
```

컨테이너 서버

```
startXsServer c1 -objectGridFile
D:\W00.biz_work\W01.SWG\W06.WASv85\WASv855_guide\WJPAObjectGrid.xml -deploymentPolicyFile
D:\W00.biz_work\W01.SWG\W06.WASv85\WASv855_guide\WJPAObjectGridDeployment.xml -
catalogServiceEndPoints kr050578:4809 -jvmArgs -cp
D:\W00.biz_work\W01.SWG\W09.SampleSource\WXS\WPersonJPA.jar
```

```
xscmd -c showMapSizes -cep kr050578:4809
```

```
C:\IBM\WebSphere85\ExtremeScale\ObjectGrid\bin>xscmd -c showMapSizes -cep kr050578:4809
시작 시간 :2015-01-12 21:53:17.148

CWXS10068I: 명령 실행 중: showMapSizes

*** PersonGrid 데이터 그리드 및 PersonMapSet 맵 세트에 대한 결과 표시.

총 카탈로그 서비스 도메인 수: 0 (0 B)
<사용된 바이트 통계는 단순 오브젝트 또는 COPY_TO_BYTES 복사 모드를 사용 중인 경우에만 정확합니다.>

CWXS10040I: showMapSizes 명령이 완료되었습니다.

종료 시간: 2015-01-12 21:53:20.684

C:\IBM\WebSphere85\ExtremeScale\ObjectGrid\bin>
```

 kr050578Node04 노드에 있는 server1 서버의 DB2 Universal JDBC Driver DataSource 데이터 소스에 대한 연결 테스트 조작에 성공했습니다.

데이터 소스

선택된 JDBC 제공자에 연관된 데이터 소스의 설정을 편집하려면 이 페이지를 사용하십시오. 데이터 소스 오브젝트는 데이터베이스에 대한 액세스를 위한 연결을 애플리케이션에 제공합니다. [안내된 활동](#) 내 이 태스크에 대한 자세한 내용을 학습하십시오. 안내된 활동에서는 태스크 단계의 목록과 주제에 대한 좀더 일반적인 정보가 제공됩니다.

범위: 셀=**kr050578Node03Cell**, 노드=**kr050578Node04**

범위는 자원 정의를 볼 수 있는 레벨을 지정합니다. 범위 및 작동 방법에 관한 자세한 정보는 [범위 설정 도움말을 참조하십시오](#).

노드=**kr050578Node04**

환경 설정

<div> <div>새로 작성...</div> <div>삭제</div> <div>연결 테스트</div> <div>상태 관리...</div> </div>					
<div> <div> <div></div> <div></div> <div></div> <div></div> </div> </div>					
선택	이름	JNDI 이름	범위	제공자	설명
다음 자원을 관리할 수 있습니다.					
<input type="checkbox"/>	DB2 Universal JDBC Driver DataSource	jdbc/db2datasource	노드=kr050578Node04	DB2 Universal JDBC Driver Provider	DB2 Universal Driver DataSource

① IBM WXS 의 카탈로그 서버와 컨테이너 서버의 구성을 결정하는 설정 정보 파일 준비

일반적으로 설치되어 있는 [WXS 설치디렉토리]/ObjectGrid/wesb 디렉토리에 설정 샘플파일이 들어있으며 해당 샘플파일을(wesb_objectGrid.xml, wesb_objectGridDeployment.xml) 약간 변형해서 사용하는 형태로 가이드를 진행하도록 하겠습니다.

중요 부분 in wesb_objectGrid.xml

```
<objectGrids>
  <objectGrid name="Grid">
    <backingMap name="Customer" lockStrategy="PESSIMISTIC"
ttlEvictorType="CREATION_TIME" timeToLive="600"/>
  </objectGrid>
</objectGrids>
```

샘플로 제공되는 objectGrid 설정 파일을 살펴보면 단순히 Customer 라는 backingMap 을 하나 만드는 설정입니다. backingMap 은 실제 캐시된 객체가 저장되며 이전에 설명했던 Grid 내부의 파티션 안에 존재합니다. 그 뒤의 설정들을 계속 살펴보면 락 전략은 pessimistic 방식으로 수행하고 ttl Evictor 를 지정하는데 ttl Evictor 란 시간 기반으로 캐시를 버리는 형태의 모듈입니다. 설정을 읽어보면 바로 이해할

수 있는 것처럼 생성된 시간 이후로 600 초가 지나면 캐시가 무효하다고 보고 버리도록 설정된 것입니다.

변경 부분 in wesb_objectGrid2.xml

```
<objectGrids>
  <objectGrid name="Grid">
    <backingMap          name="CacheData01"          lockStrategy="PESSIMISTIC"
copyMode="COPY_TO_BYTES" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="600"/>
  </objectGrid>
</objectGrids>
```

사용의 편의를 위해 이를 상단과 같이 변경합니다. 변경된 부분을 간단하게 설명드리면 이름을 CacheData01 로 변경했고 copyMode="COPY_TO_BYTES" 를 추가하여 eXtreme data format (XDF) 을 사용할수 있도록 하였습니다. XDF 는 IBM 이 만든 WXS 고유의 포맷으로 Java 뿐만이 아니라 C# 이나 .NET 에서도 사용될 수 있는 직렬화 포맷으로 성능 및 메모리 사용율이 우수한 포맷입니다. ttlEvictor 는 "LAST_ACCESS_TIME" 으로 변경하여 생성 시간 대신에 마지막 접속 시간을 기준으로 캐시된 데이터를 만료시키도록 설정을 변경하였습니다.

다음으로 Grid Container 배치관련 설정파일을 살펴보도록 하겠습니다.

중요 부분 in wesb_objectGridDeployment.xml

```
<objectgridDeployment objectgridName="Grid">
    <mapSet name="mapSet" numberOfPartitions="13" minSyncReplicas="0" maxSyncReplicas="1"
>
    <map ref="Customer"/>
</mapSet>
</objectgridDeployment>
```

샘플로 제공되는 objectGridDeployment 설정 파일을 살펴보면 objectgrid 이름은 Grid 이며 mapSet 이라는 이름의 mapSet 을 하나 만들고 파티션의 개수는 13개, 동기화 형태로 복제를 수행해서 가지고 있는 복제본의 최대 개수는 1 로 설정하였습니다. 이렇게 만든 mapSet 안에는 Customer 라는 map 이 하나 들어 있는 형태입니다..

중요 부분 in wesb_objectGridDeployment2.xml

```
<objectgridDeployment objectgridName="Grid">
    <mapSet name="mapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="1" >
        <map ref="CacheData01"/>
    </mapSet>
</objectgridDeployment>
```

간단하게 테스트하는게 목적이니 해당 설정의 파티션 개수를 줄이고 map 이름만 objectGrid 와 동일하게 변경하고 수정을 마칩니다.

나머지 설정은 지난 강좌와 동일하게 사용하도록 하며 여기까지 하셨다면 WXS Server 에 대한 준비가 완료된 것 입니다. 참고로 말씀드린 것처럼 나머지 부분은 지난 강좌와 동일하게 사용해도 되지만 간단한 테스트를 위해서 제가 테스트시에 간단하게 만들었던 시작 스크립트를 첨부드리오니 참고하시기 바랍니다.

카탈로그 서버

```
startXsServer cat1 -listenerPort 4809 -catalogServiceEndPoints cat1:kr050578:6600:6601
```

컨테이너 서버

```
startXsServer c1 -objectGridFile
C:\IBM\WebSphere85\ExtremeScale\ObjectGrid\wesb\wesb_objectGrid2.xml -deploymentPolicyFile
C:\IBM\WebSphere85\ExtremeScale\ObjectGrid\wesb\wesb_objectGridDeployment2.xml -
catalogServiceEndPoints kr050578:4809
```

이를 이용해서 WXS 의 카탈로그 서버와 컨테이너 서버를 구동하면 하단과 같은 결과를 확인할 수 있습니다. (참고로 복제본은 그리드 설정이 minSyncReplicas="0" maxSyncReplicas="1" 이기 때문에 컨테이너 서버를 하나 더 두게 되는 경우에 확인하실 수 있습니다.)

```
xscmd -c showMapSizes -cep kr050578:4809
```

```
C:\IBM\WebSphere85\WeXtremeScale\ObjectGrid\bin>xscmd -c showMapSizes -cep kr050578:4809
시작 시간 :2015-01-07 17:14:21.387

CWXS10068I: 명령 실행 중: showMapSizes

*** Grid 데이터 그리드 및 mapSet 맵 세트에 대한 결과 표시.

*** c1에 대한 맵 나열 ***
맵 이름      파티션 맵 항목      사용한 바이트      샤드 유형      컨테이너
-----
CacheData01  0          0          0          Primary      c1_C-1
CacheData01  1          0          0          Primary      c1_C-1
CacheData01  2          0          0          Primary      c1_C-1
CacheData01  3          0          0          Primary      c1_C-1
CacheData01  4          0          0          Primary      c1_C-1
서버 총계: 0 <0 B>

총 카탈로그 서비스 도메인 수: 0 <0 B>
<사용된 바이트 통계는 단순 오브젝트 또는 COPY_TO_BYTES 복사 모드를 사용 중인 경우에만 정확합니다.>

CWXS10040I: showMapSizes 명령이 완료되었습니다.
```

3) WebSphere eXtream Scale Client 애플리케이션 개발

당연히 WXS 를 활용한 글로벌 캐시에 접속하고 데이터를 input 하거나 get 하기 위해서는 WXS 에서 제공되는 API 를 사용해야 합니다. 이전 강좌처럼 WXS client 가 설치된 IBM WAS 제품이라면 별도의 라이브러리 추가나 설정이 필요하지 않습니다. 그러나 WXS client 가 설치되지 않은 경우라면 ogclient.jar 파일이나 wsogclient.jar 파일만 애플리케이션에서 참조하면 됩니다.

그럼 이러한 환경에서 간단한 샘플 애플리케이션을 개발해 보도록 하겠습니다. 우선은 WXS 에서 제공되는 API 를 활용하여 카탈로그 서버에 접속하는 코드를 하단과 같이 작성합니다. 참고적으로 해당 부분은 계속 재사용되기 때문에 싱글톤 형태로 구성하는 것도 하나의 좋은 방법입니다.

```
// Retrieve an ObjectGridManager instance.
ogm = ObjectGridManagerFactory.getObjectGridManager();

// Obtain a ClientClusterContext by connecting to a catalog
// service domain, manually supplying the catalog service endpoints,
// and optionally specifying the ClientSecurityConfiguration and
// client ObjectGrid override XML file URL.
String catalogServiceEndpoints = endpoint;

try {
    ccc = ogm.connect(catalogServiceEndpoints, (ClientSecurityConfiguration) null, (URL) null);
    // Obtain a distributed ObjectGrid using ObjectGridManager and providing
    // the ClientClusterContext.
    og = ogm.getObjectGrid(ccc, gridName);
    System.out.println("CatalogServer connection!");
} catch (ConnectException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

보시면 아시겠지만 ObjectGridManagerFactory 를 사용하여 ObjectGridManager 를 받아와서 연결을 수행하여 ObjectGrid 를 가지고 오는 형태로 간단하게 구성가능 합니다. (참고로 endpoint 는 카탈로그 서버 주소:포트 형태로 "kr050578:4809" 을 gridName 은 Grid 이름으로 "Grid" 를 입력합니다.)

다음으로 서블릿 페이지를 하나 만들어서 하단의 코드를 이용해서 WXS 캐시에 데이터를 input 하는 코드를 작성합니다.

```
PrintWriter out = response.getWriter();

try {
    // Get a session
    Session session = og.getSession();
    // Get the ObjectMap
    ObjectMap map1 = session.getMap(mapName);
    //transaction related with session begin
    session.begin();
    // Insert data into the map.This is implicitly transactional
    map1.insert(1, "TestTestTest");
    // Close the session so that it can be managed in a pool
    session.commit();
    out.println("Data Insert Completed.");
} catch (ObjectGridException e) {
    e.printStackTrace();
}
```

여기서는 GridObject 를 이용하여 session 을 맺고 ObjectMap 에 원하는 데이터를 insert 하는 형태로 진행됩니다. 하단의 Grid 의 내부 구조를 보시면 좀 더 이해에 도움이 됩니다.

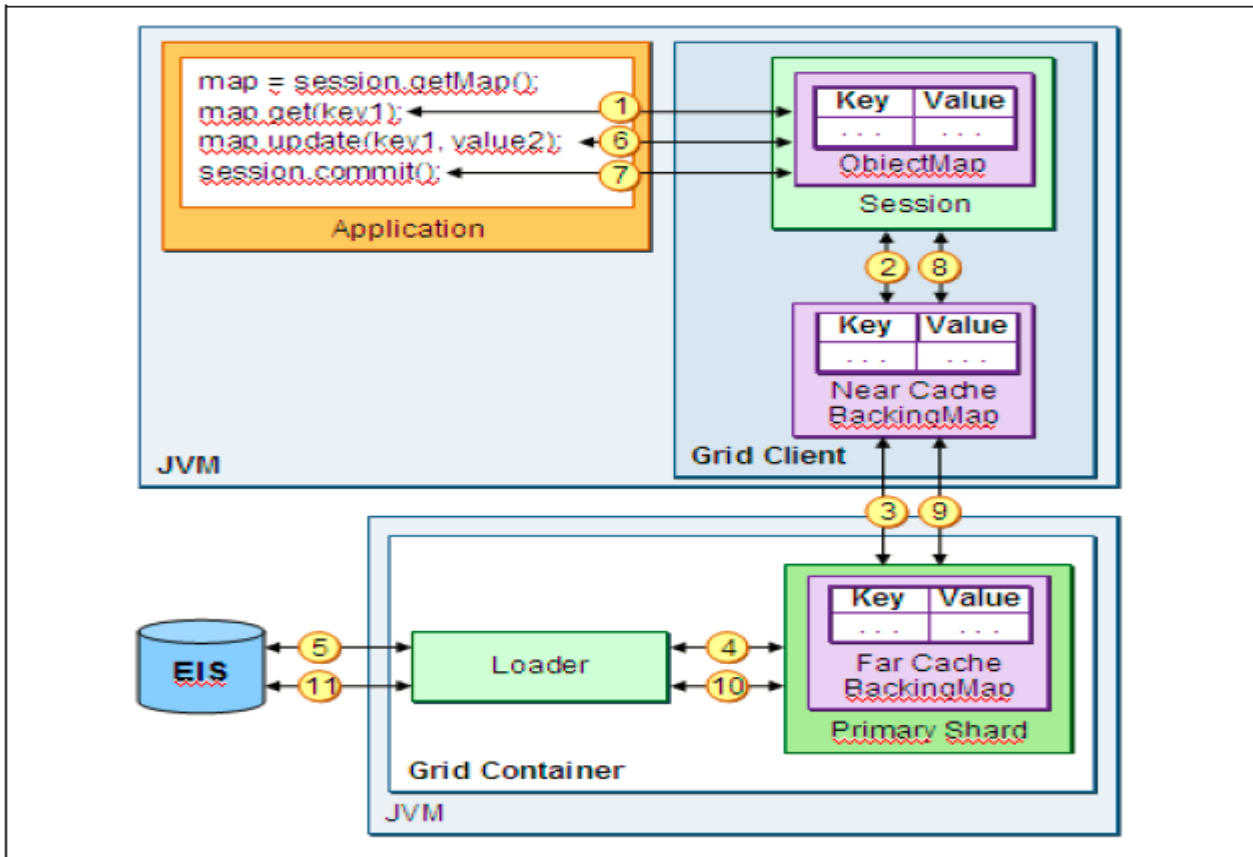


Figure 2-18 Component interactions for simple grid access

(참고로 클라이언트에서 보관하는 Near Cache 는 락 전략을 NONE이나 OPTIMISTIC 을 사용했을 때 같이 사용 가능해지며 현재 강좌에서는 락 전략을 PESSIMISTIC 을 사용하고 있기 때문에 설명은 생략하도록 하겠습니다.)

즉, ObjectMap 은 Grid 에 저장된 BackingMap 과 연결되어 실제로 가지고 오는 Map 객체이며 내부적으로 Key, Value 쌍을 가지고 있게 됩니다.

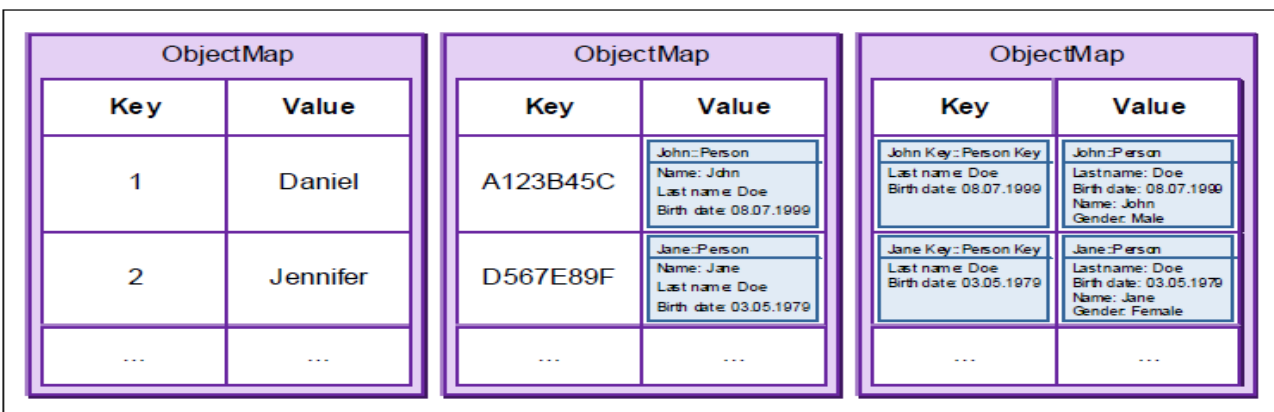


Figure 2-4 ObjectMap examples

위의 작업과 동일하게 서블릿 페이지를 하나 더 만들어서 이번에는 WXS Grid 에 캐시된 정보를 가지고 오는 코드를 작성합니다.

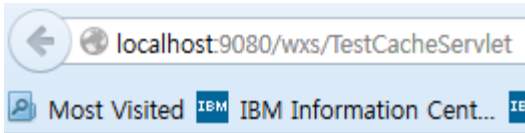
```
PrintWriter out = response.getWriter();

try {
    // Get a session
    Session session = og.getSession();
    // Get the ObjectMap
    ObjectMap map1 = session.getMap(mapName);
    // Insert data into the map. This is implicitly transactional
    String data = (String)map1.get(1);
    // Close the session so that it can be managed in a pool
    session.close();
    out.println("Data get Completed : " + data);
} catch (ObjectGridException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

여기까지 하시면 단순하지만 WXS 로 구성된 글로벌 캐시에 데이터를 저장하고 가지고 오는 테스트 애플리케이션 작성이 완료된 것입니다.

4) 테스트

작성된 애플리케이션을 WAS 에 모두 배포한 후에 하단과 같이 WXS 캐시에 데이터를 input 하는 서블릿을 호출합니다.



Data Insert Completed.

호출이 정상적으로 수행되면 하단과 같이 WXS Grid 컨테이너에 데이터가 들어온 것을 확인할 수 있습니다.

```
C:\W\IBM\WWebSphere85\WExtremeScale\WObjectGrid\Wbin>xscmd -c showMapSizes -cep kr0505
78:4809
시작 시간 :2015-01-07 18:20:45.018

CWXS10068I: 명령 실행 중: showMapSizes

*** Grid 데이터 그리드 및 mapSet 맵 세트에 대한 결과 표시.

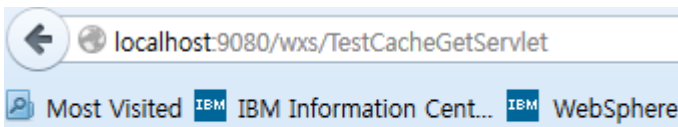
*** c1에 대한 맵 나열 ***
맵 이름      파티션 맵 항목      사용한 바이트      샤드 유형      컨테이너
-----
CacheData01  0          0          0          Primary      c1_C-1
CacheData01  1          1          440 B      Primary      c1_C-1
CacheData01  2          0          0          Primary      c1_C-1
CacheData01  3          0          0          Primary      c1_C-1
CacheData01  4          0          0          Primary      c1_C-1
서버 총계: 1 <440 B>

총 카탈로그 서비스 도메인 수: 1 <440 B>
<사용된 바이트 총계는 단순 오브젝트 또는 COPY_TO_BYTES 복사 모드를 사용 중인 경
우에만 정확합니다.>

CWXS10040I: showMapSizes 명령이 완료되었습니다.

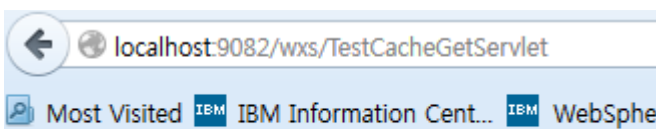
종료 시간: 2015-01-07 18:20:48.574
```

다음으로 WXS 캐시에서 데이터를 가지고 오는 서블릿 페이지를 호출합니다.



Data get Completed : TestTestTestTest

정상적으로 캐시에 저장된 데이터를 화면으로 확인할 수 있습니다. 이와 마찬가지로 다른 WAS 서버로 해당 애플리케이션을 배포하여 데이터를 가지고 오는 서블릿 페이지를 호출해보면 글로벌 캐시 형태와 동일하게 이전에 1번 서버에서 저장한 데이터를 정상적으로 반환하는 것을 확인할 수 있습니다.



Data get Completed : TestTestTestTest

이를 좀더 쉽게 설명 드리면 최초로 언급한 것과 같은 형태의 글로벌 캐시 구성을 수행한 것이며 여러 클라이언트(WAS)가 공통된 글로벌 캐시에 접속하여 데이터를 작성하거나 읽는 작업을 수행할 수 있습니다.

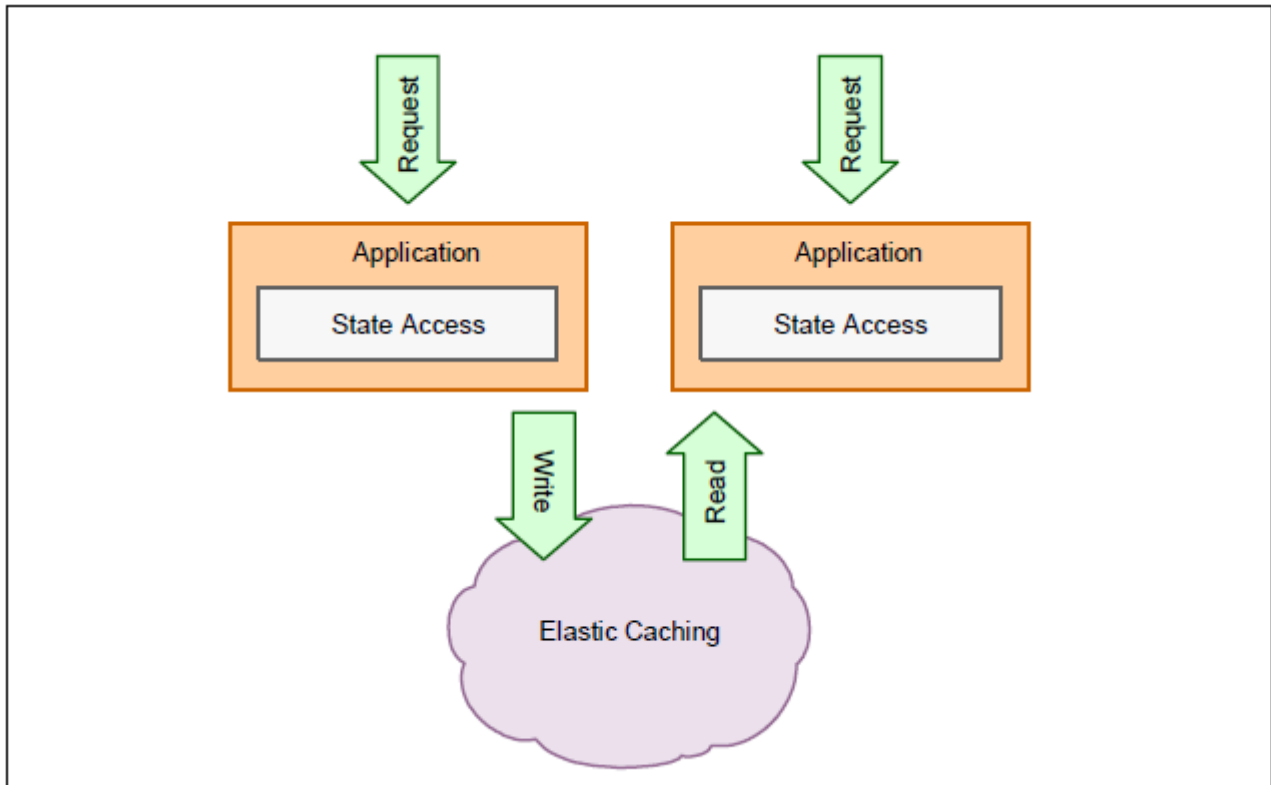
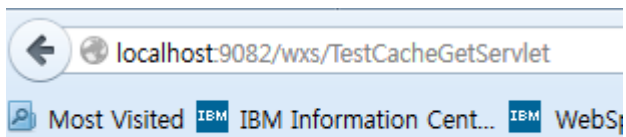


Figure 3-1 Application state store scenario

이렇게 하면 아주 간단하게 WXS 를 활용한 글로벌 캐시 구성이 완료된 것입니다.

마지막으로 아무 호출을 하지 않고 600 초를 기다린 후 조회 서블릿을 호출해보면 캐시된 데이터가 만료되어서 하단과 같이 null 을 반환하는 것을 확인할 수 있습니다.

(당연히 이전 objectGrid 설정에서 ttlEvictorType 를 NONE 으로 하면 제거되지 않습니다.)



Data get Completed : null

9) 참고 자료

IBM WebSphere eXtreme Scale Version 8.6 Information Center

http://pic.dhe.ibm.com/infocenter/wxsinfo/v8r6/index.jsp?topic=%2Fcom.ibm.websphere.extremescale.doc%2Fwelcome%2Fwelcome_xs.html