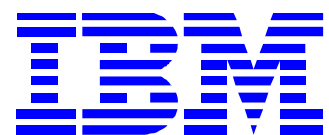


WebSphere eXtreme Scale v8.6

WXS 를 활용한 In-Memory DataGrid 구성 가이드

(2015. 01.)

IBM SWG WebSphere



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

안녕하세요 freeman 입니다.

이미 설명을 드렸지만 IBM WXS 란 IBM WebSphere 제품 중에서 분산 캐싱 플랫폼(distributed caching platform) 을 제공할 수 있는 Middleware 제품이며 이를 통해서 In-Memory DataGrid 를 구성할 수 있습니다. 여기서 분산 캐싱 플랫폼이란 다수의 JVM 에 분산되어 있는, 사용하지 않는 메모리를 취합하여 하나의 논리적인 캐시처럼 사용하는 기술을 의미하며 언급한 것과 같이 In-Memory DataGrid 라고 불리워집니다. 이를 잘 활용하면 거대한 트랜잭션 볼륨을 가진 중요한 비즈니스 어플리케이션에 지속적인 고성능과 확장성, 가용성을 실현해 줄 수 있습니다.

좀 더 자세히 이야기 하자면 분산 캐싱 기술을 이용해서 하단과 같이 아키텍처적으로 DB layer 앞 단에 위치하여 고비용이며 빈번하게 발생하는 DB 작업 대신에 미리 만들어 둔 Cache layer 에서 실제 데이터를 가지고 와서 고성능, 저비용으로 작업이 가능한 기술을 의미합니다. 여기서 Cache layer 에는 DB 데이터가 들어가며 아키텍처적으로 Pre-loader 라는 것을 사용하여 사전에 미리 다 가져다 놓을 수도 있고 요청이 있을 때마다 가지고 올 수도 있습니다. 미리 언급한 것처럼 비싸게 DB 에 직접 access 하는 것보다 Cache layer 에서 데이터를 가지고 오는 것이 훨씬 저비용으로 고성능을 제공할 수 있습니다.

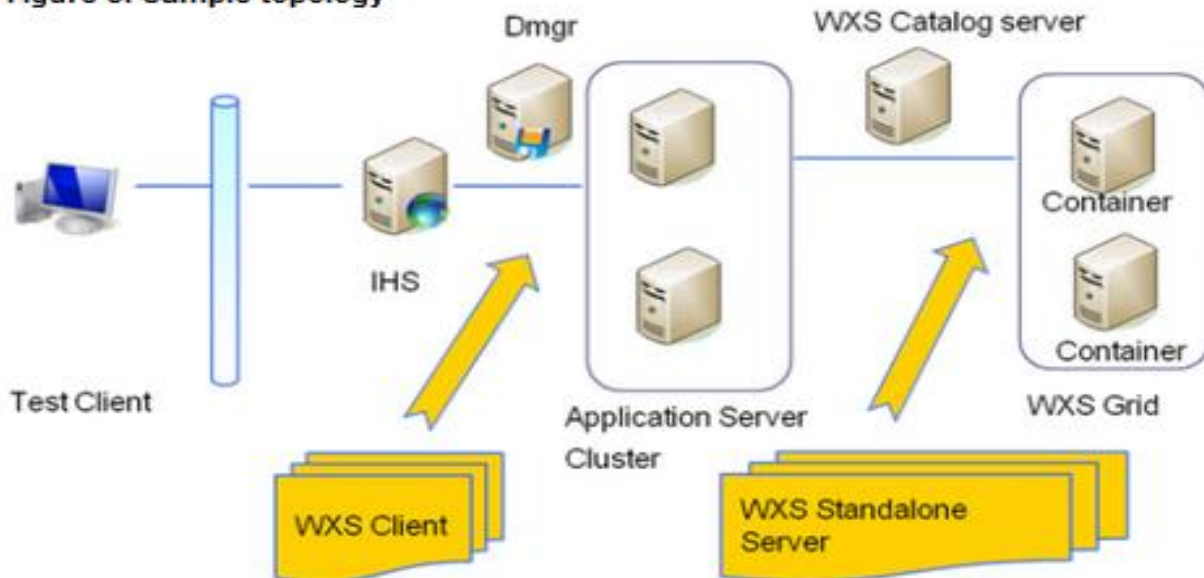


이전 강좌에서 IBM WXS 를 활용한 글로벌 캐시에 대해서 간단한 샘플과 함께 설명을 진행했는데 이번 강좌에서는 실제로 상단의 아키텍처와 동일하게 DB 앞에 WXS 를 In-Memory DataGrid 로 활용하는 간단한 샘플을 살펴보도록 하겠습니다. 특히, 이번 강좌에서는 JPA 라는 Java EE 표준 OR Mapping Framework 를 사용하는 형태로 강좌를 진행하도록 하겠습니다.

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

설치 및 구성의 경우는 지난 강좌인 글로벌 캐시 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, JPA 애플리케이션 작성

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

DB 데이터를 자동으로 캐시하고 조회하는 WXS 를 활용한 In-Memory DataGrid 구성을 위하여 Sample DB 를 활용하여 새롭게 table 을 생성합니다. WXS 에 대한 기본개념에 대한 이해를 돕기 위하여 간단한 table 을 작성합니다.

(create table PERSON(ID Integer not null GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1, NO CACHE), Name Varchar(20), Join_date Date, PRIMARY KEY (ID)))

```
db2 => create table PERSON(ID Integer not null GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT BY 1, NO CACHE), Name Varchar(20), Join_date Date, PRIMARY KEY (ID))
DB20000I  SQL 명령이 완료되었습니다.
db2 => describe table PERSON
```

컬럼 이름	데이터 스키마	데이터 유형	컬럼 길이	크기	N
ID	SYSIBM	INTEGER		4	0
NAME	SYSIBM	VARCHAR	20		0
JOIN_DATE	SYSIBM	DATE	4		0

```
3 레코드가 선택되었습니다.
db2 =>
```

PERSON 이라는 table 작성을 완료하였으면 조회를 위한 간단한 임시 데이터를 하단과 같이 입력합니다.

(Insert into PERSON(Name, Join_date) values('JungWoon Lee', CURRENT DATE))

Insert into PERSON(Name, Join_date) values('JungWoo Lee', DATE ('2015-1-5'))

Insert into PERSON(Name, Join_date) values('JungWo Lee', CURRENT DATE)

Insert into PERSON(Name, Join_date) values('JungW Lee', DATE ('2015-1-6'))

Insert into PERSON(Name, Join_date) values('Jung Lee', CURRENT DATE)

Insert into PERSON(Name, Join_date) values('Jun Lee', DATE ('2015-1-7')))

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

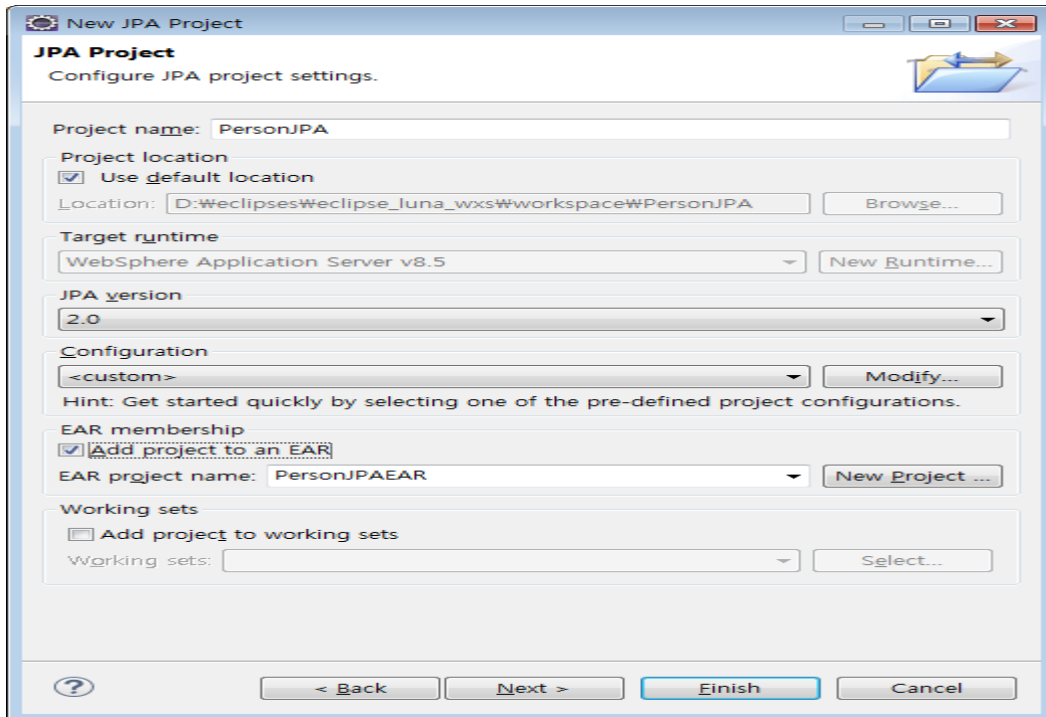
ID	NAME	JOIN_DATE
7	JungWoon Lee	2015-01-12
8	JungWoo Lee	2015-01-05
9	JungWo Lee	2015-01-12
10	JungW Lee	2015-01-06
11	Jung Lee	2015-01-12
12	Jun Lee	2015-01-07

```
6 레코드가 선택되었습니다.
```

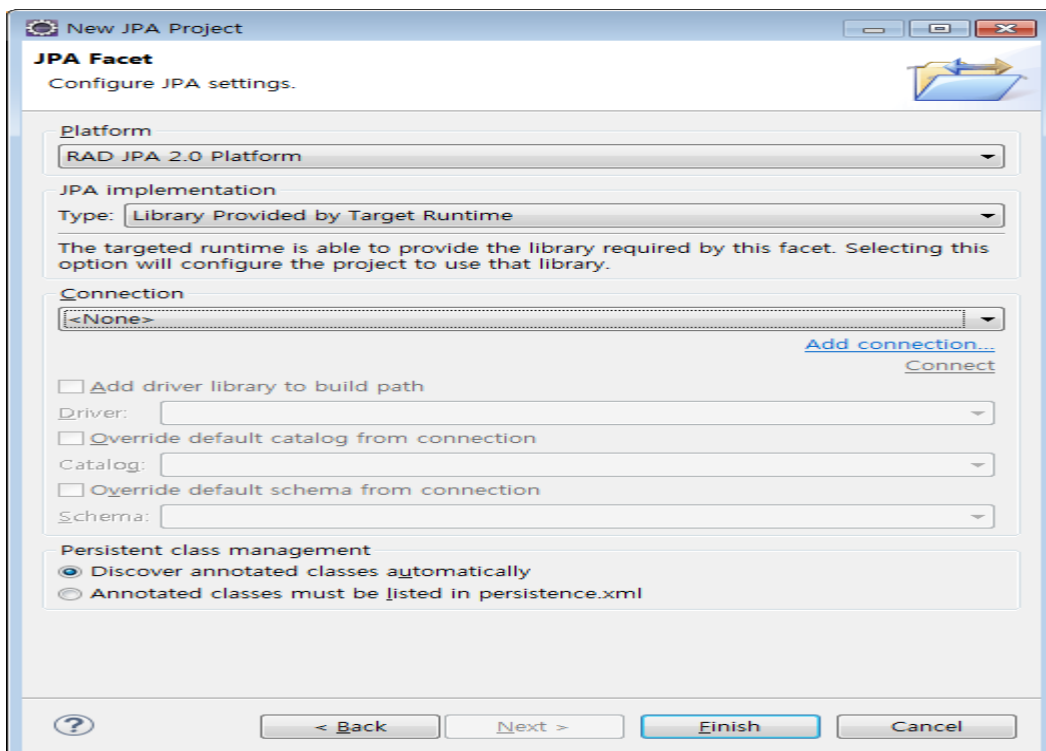
② 조회를 수행하기 위한 간단한 JPA 애플리케이션 작성

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.

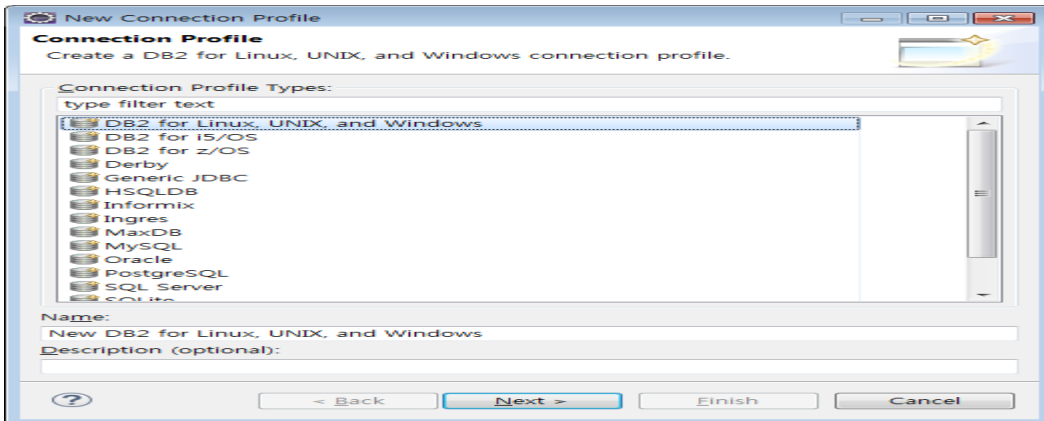
(이때 JPA 버전은 2.0 을 선택하고 WAR 프로젝트를 넣기 위하여 EAR 프로젝트도 같이 생성합니다.)



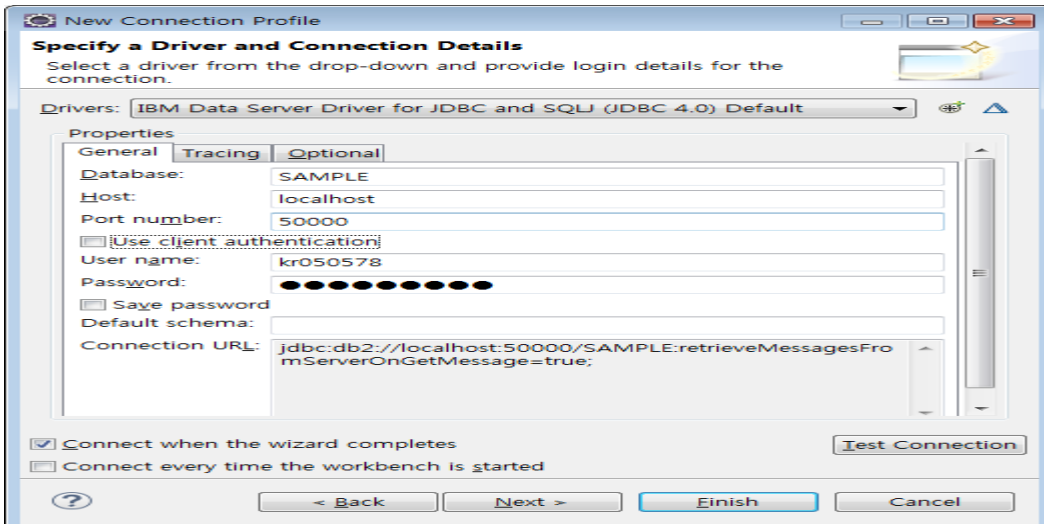
JPA 프로젝트 생성중에 Connection 관련한 마법사가 나오면 DB 에 연결하는 작업을 수행하기 위하여 'Add connection' 버튼을 클릭합니다



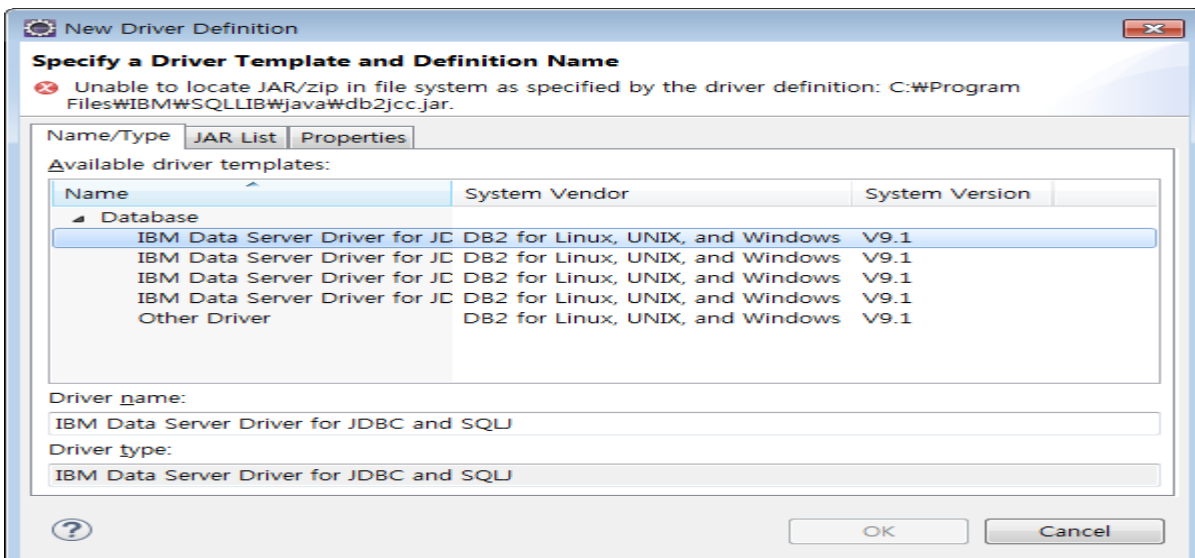
Connection profile 에서는 연결되는 DB를 선택하고 다음을 클릭합니다.



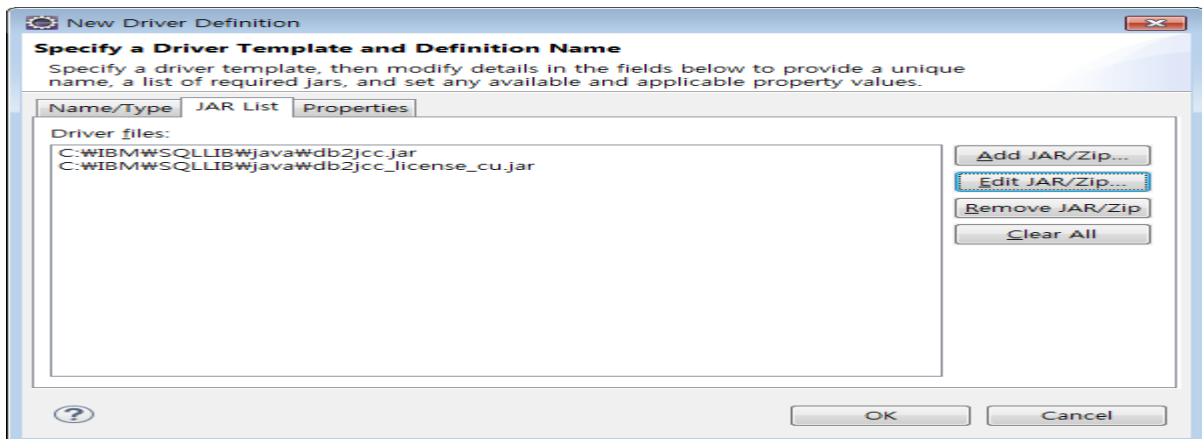
연결을 위한 User/Password 정보를 입력한 후에 Driver 설정을 위하여 Drivers 메뉴 옆의 아이콘을 클릭합니다.



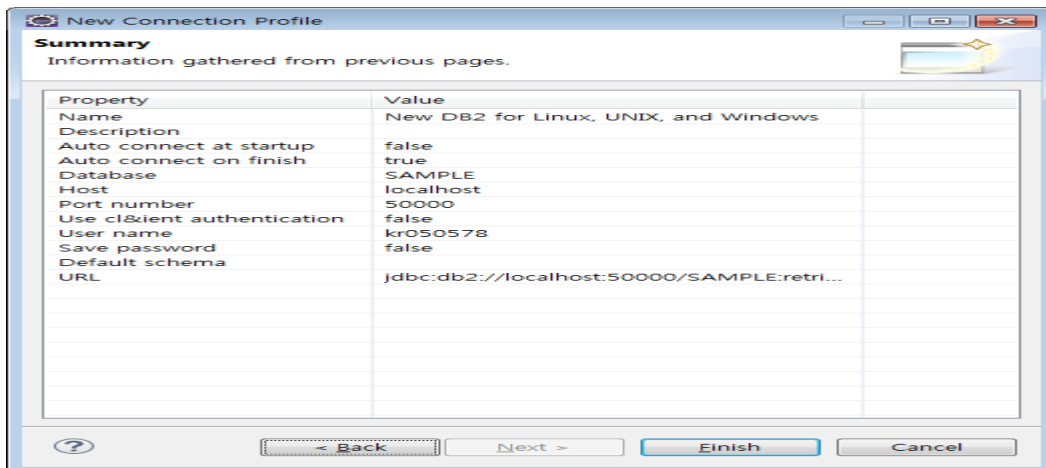
DataBase 기본 정보를 선택한 후에 DB Driver jar 파일을 설정하기 위하여 JAR List 탭을 클릭합니다.



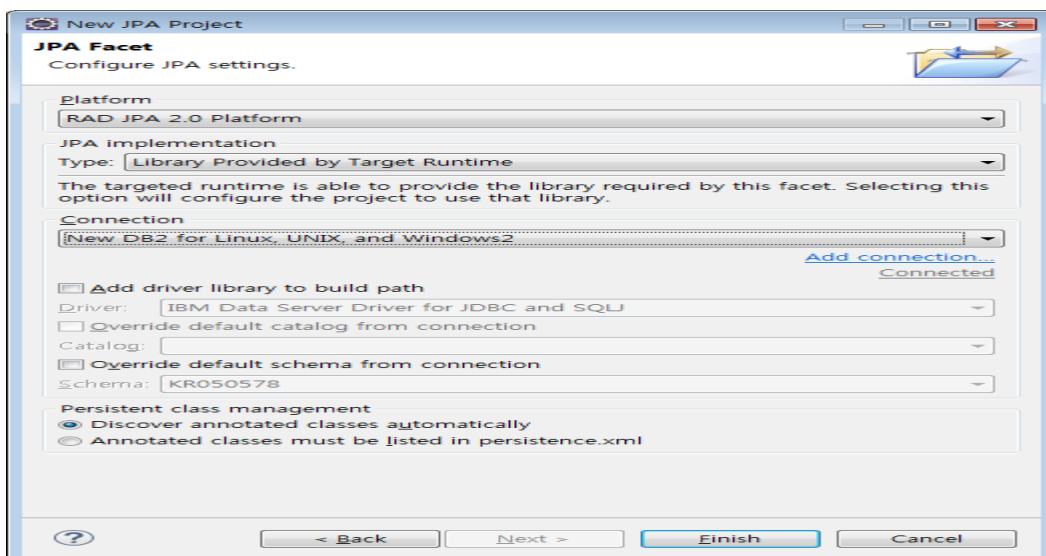
Driver Jar 파일의 위치를 기본이 아니라 가지고 있는 DB Driver 위치로 변경한 후 OK 를 클릭합니다.



요약정보를 확인하고 이상이 없다면 Finish 를 클릭합니다.



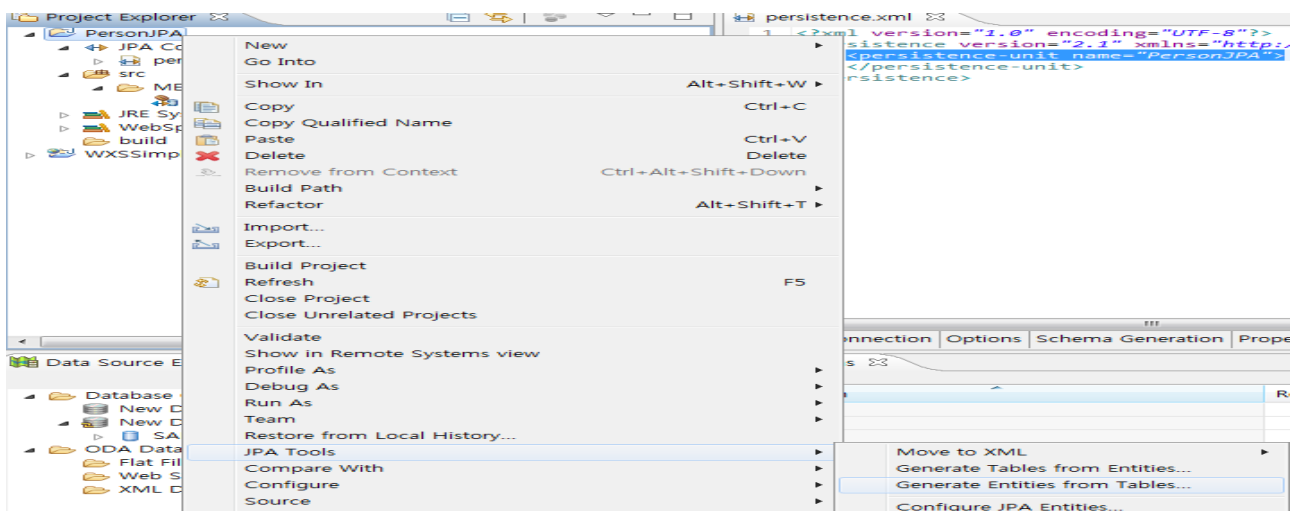
마지막으로 정보를 확인하고 Finish 를 클릭하여 JPA 생성을 완료합니다.



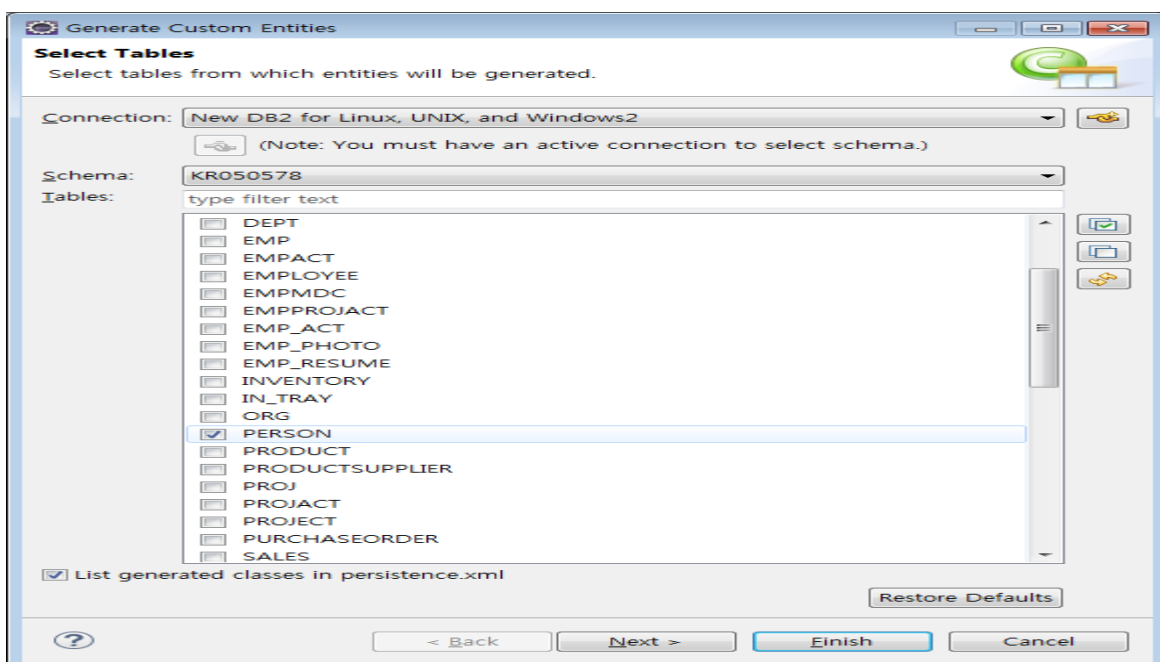
여기까지 작업을 문제없이 수행했다면 하단과 같은 구조로 JPA 프로젝트가 생성된 것을 확인할 수 있습니다.



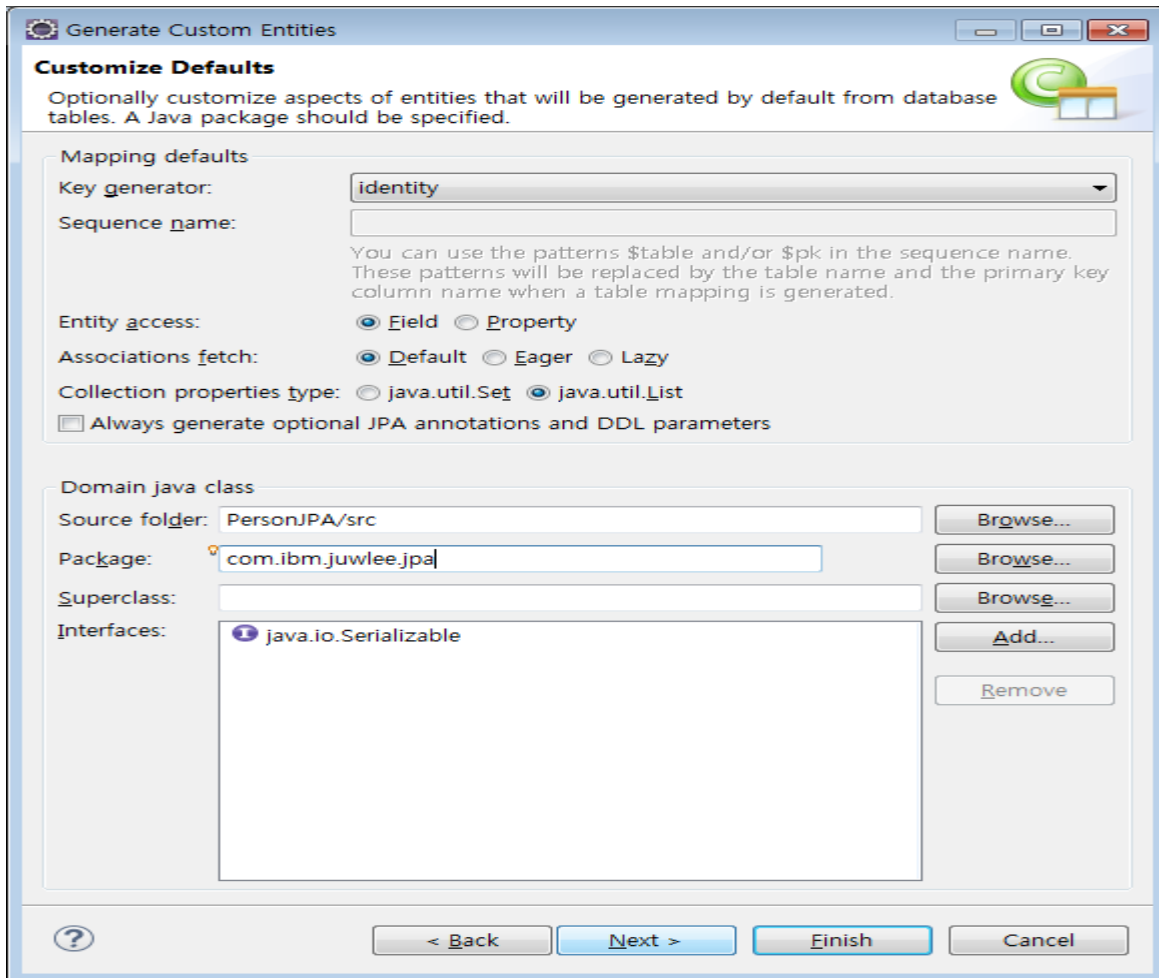
이제 실제로 DB 와 연결된 JPA 소스를 자동으로 생성하기 위하여 프로젝트에서 마우스 우 클릭 후에 JPA Tools > Generate Entities from Tables 를 클릭합니다.



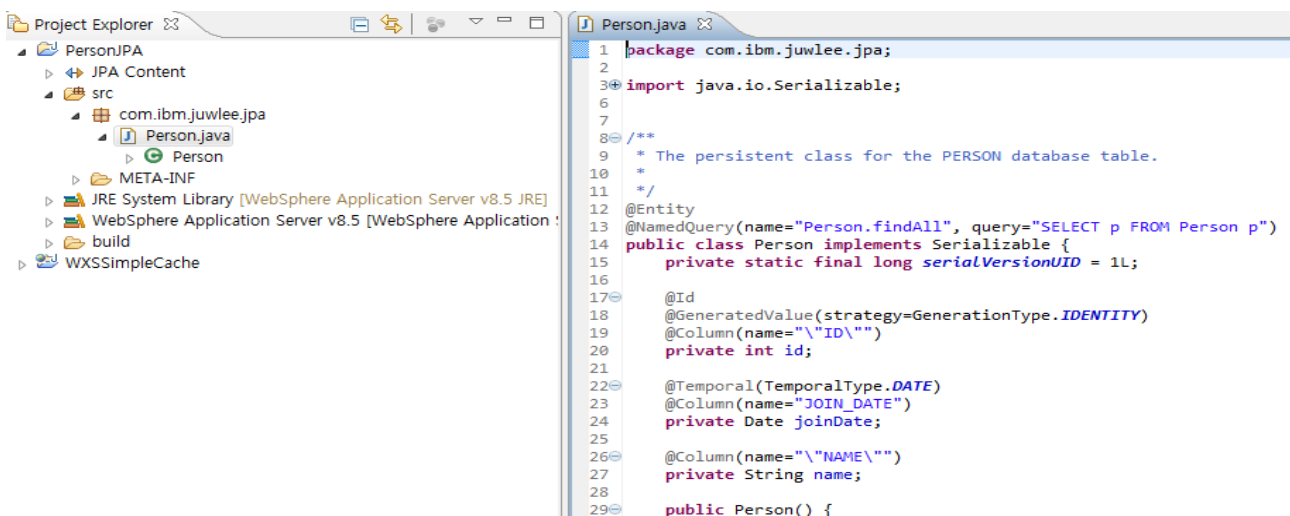
이전에 만들어진 Connection 에서 table 을 선택하는 마법사가 나오면 방금 생성한 Person table 을 선택하고 다음을 클릭합니다.



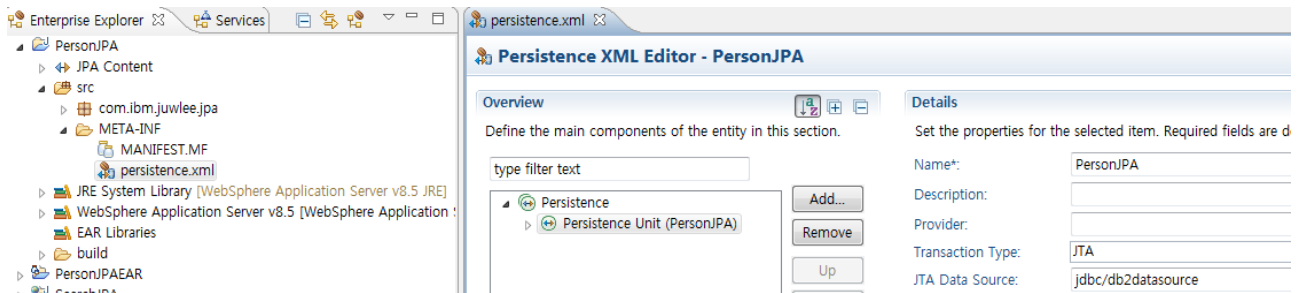
이전에 Person table 을 생성할 때, ID 값을 자동생성 했기 때문에 Key generator 를 identity 로 선택하고 package 명도 입력한 후 완료를 클릭합니다.



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 것입니다. 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

새로 작성... 삭제 연결 테스트 상태 관리...					
선택 이름 JNDI 이름 범위 제공자 설명 카테고리					
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 그룹

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

롤아웃 업데이트

파일 제거

내보내기

DDL 내보내기

선택

이름

애플리케이션 상태

다음 자원을 관리할 수 있습니다.

☐

[DefaultApplication](#)

➡

☐

[PersonJPAEAR](#)

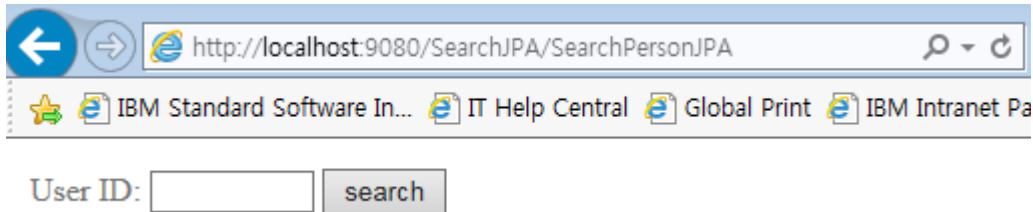
➡

- 10 -

4) 샘플 JPA 테스트

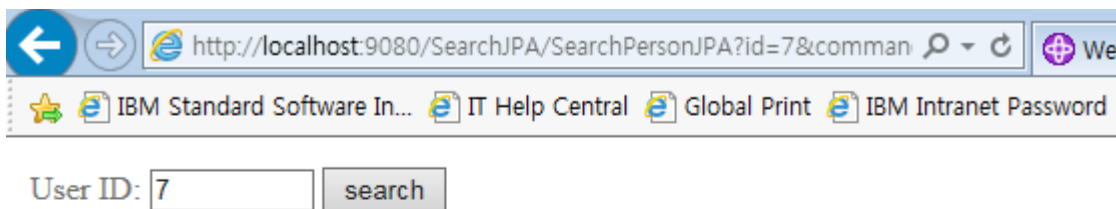
이제 JPA 테스트를 위한 준비가 다 된 것이므로 샘플 테스트를 수행해보도록 하겠습니다. 먼저 배포된 애플리케이션의 만들어진 서블릿 페이지를 호출합니다.

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



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

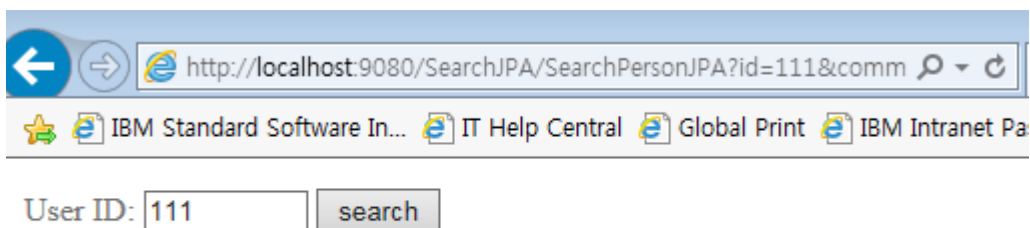
호출이 정상적으로 완료되면 Sample DB 에 넣어둔 data 를 하나 조회합니다.



Search form showing the URL <http://localhost:9080/SearchJPA/SearchPersonJPA?id=7&command=search>. The form includes a 'User ID' input field with the value '7' and a 'search' button.

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/SearchJPA/SearchPersonJPA?id=111&command=search>. The form includes a 'User ID' input field with the value '111' and a 'search' button.

person is not found : 111

여기까지 동일한 결과를 확인하셨다면 문제없이 샘플 JPA 테스트가 완료된 것입니다.

5) IBM WXS 테스트를 위한 JPA 애플리케이션 변경

이전 파트에서 JPA 애플리케이션을 작성해서 간단하게 DB 조회하는 샘플을 만들어서 테스트해봤습니다. JPA 가 동작하는 방식을 보여주는 간단한 샘플이었고 이제 해당 강좌의 본론으로 들어가서 동일한 애플리케이션을 활용해서 WXS 의 Grid 를 통해서 Data 를 조회하고 없을 경우에 DB 를 조회하여 Data 를 Grid 에 자동으로 넣는 애플리케이션을 작성해 보도록 하겠습니다. 이를 위해서는 Loader 라는 plugin 을 하단과 같은 구조로 사용됩니다. (기본적으로 WXS 는 JPALoader 를 제공합니다.)

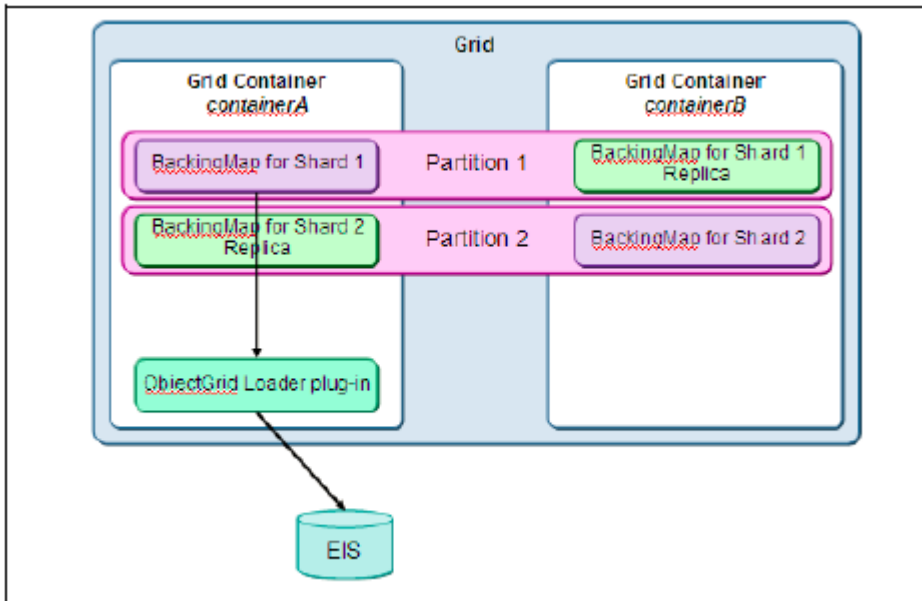


Figure 2-5 Example of using a BackingMap with a loader

이를 좀 더 보강해서 설명하자면 Write-Through 방식의 Grid 사용방식으로 JPA 를 사용해서 DB 조회하는 것과 거의 동일하게 API 만 변경해서 작업하면 실제 뒷단의 WXS 와 DB 에는 JPALoader 라는 미리 만들어진 plugin 을 사용하여 필요한 Grid 와 DB 의 동기화와 같은 작업을 알아서 처리해줍니다. Write-Through 라는 방식은 Write-Behind 방식과 상대되는 말로 IMDG 로 사용되는 WXS 와 DB 간에 동기화를 Sync 방식으로 할 것이냐 aSync 방식으로 할 것이냐의 차이입니다. 예를 들어서 Write-Through 방식은 CRUD 와 같은 DB 작업이 발생할 때 WXS 와 DB 간에 매번 바로 동기화(트랜잭션 연결)를 수행하는 것이고 Write-Behind 방식은 CRUD 작업이 벌어져도 WXS 와 DB 간에 매번 바로 동기화 하는 것이 아니나 지정된 횟수나 시간 방식으로 동기화가 이루어지는 작업입니다. 현재 강좌에서는 우선 Write-Through 방식으로 처리되는 형태로 예제를 진행하도록 하겠습니다.

이제 해당 방식으로 애플리케이션을 작성해보겠습니다. 우선 이전 단순 JPA 샘플에서는 없었지만 WXS 에 데이터를 저장하기 위해서 카탈로그 서버에 연결해서 WXS 의 ObjectGrid 객체를 받아오는 소스 부분이 하단과 같이 추가되어야 합니다. 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";
private String TARGET_SERVLET;

private ObjectGrid og=null;

public SearchPersonGrid() {
    super();

    CatalogClass cc = CatalogClass.getInstance();
    og = cc.getObjectGrid();
}

protected Person retrievePerson(int Id) throws RuntimeException {
    Person person = null;
    Session session = null;
    try {
        // Get a session
        session = og.getSession();
        // Get a ObjectMap
        ObjectMap map = session.getMap(mapName);

        //Session start with transaction
        session.begin();

        person = (Person)map.get(Id);
        System.out.println("Person " + person);

        if (person == null) {
            throw new RuntimeException("Person is not found, id = " + Id);
        }
        session.commit();

    } catch (ObjectGridException|RuntimeException e) {
        try {
            session.rollback();
        } catch (NoActiveTransactionException | TransactionException e1) {
            e1.printStackTrace();
        }
        e.printStackTrace();
    }
    return person;
}

```

또한, 이전처럼 WAS 에서 JPA 를 수행하는 형태가 아니라 실제적으로 JPA 의 CRUD 작업은 JPALoader 라는 plugin 을 통해서 WXS 와 DB 간에서 동작하고 그 결과만 client 입장인 WAS 에 넘겨주는 형태이므로 실제로 Java SE 를 쓰고 standalond 형태로 구동시킬 WXS 가 JPA 를 사용하기 위해서 apache 의 OpenJPA 를 사용하는 형태의 구성을 persistence.xml 파일에 추가합니다. (WAS 도 기본적으로는 apache 의 OpenJPA 의 구현체를 사용합니다.) 하단을 보시면 아시겠지만 DB driver 나 접속 정보를 property 형태로 넣어주시기만 하면 됩니다.

```

<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001,
  <persistence-unit name="PersonJPA" transaction-type="JTA">
    <jta-data-source>jdbc/db2datasource</jta-data-source>
    <class>com.ibm.juwlee.jpa.Person</class>
  </persistence-unit>
  <persistence-unit name="PersonJPAwithWXS">
    <provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>
    <class>com.ibm.juwlee.jpa.Person</class>
    <properties>
      <property name="openjpa.ConnectionURL" value="jdbc:db2://localhost:50000/SAMPLE" />
      <property name="openjpa.ConnectionPassword" value="y6u7i8o9" />
      <property name="openjpa.ConnectionUserName" value="kr050578" />
      <property name="openjpa.ConnectionDriverName" value="com.ibm.db2.jcc.DB2Driver" />
    </properties>
  </persistence-unit>
</persistence>

```

마지막으로 변경된 애플리케이션을 WAS 에 배포하고 이전과 다르게 JPA 형태로 자동 작성된 PersonJPA 프로젝트는 PersonJPA.jar 로 추출하여 별도로 보관합니다.(향후 해당 jar 를 가지고 WXS 를 기동할 때 참조해야 합니다.)

6) IBM WXS 테스트를 위한 WXS Server 설정

애플리케이션이 작성되었으면 이제 JPAloader plugin 을 적용하여 WXS Server 를 구동하기 위하여 설정 작업을 진행하도록 하겠습니다.

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

중요 부분 JPAObjectGrid.xml

```
<objectGrids>
  <objectGrid name="PersonGrid" txTimeout="60">
    <bean id="TransactionCallback"
className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
      <property name="persistenceUnitName" type="java.lang.String"
value="PersonJPAwithWXS"/>
    </bean>
    <backingMap name="Person" lockStrategy="PESSIMISTIC" pluginCollectionRef="Person"/>
  </objectGrid>
</objectGrids>
<backingMapPluginCollections>
  <backingMapPluginCollection id="Person" >
    <bean id="Loader" className="com.ibm.websphere.objectgrid.jpa.JPALoader">
      <property name="entityClassName" type="java.lang.String"
value="com.ibm.juwlee.jpa.Person"/>
    </bean>
  </backingMapPluginCollection>
</backingMapPluginCollections>
```

ObjectGrid 설정 파일을 보시면 이전과 다른 com.ibm.websphere.objectgrid.jpa.JPATxCallback 빈이 보이는데 해당 빈은 이름 그대로 JPA 트랜잭션 콜백을 위해 사용하는 빈으로 JPA 를 사용하여 DB 동기화를 할 때 사용하는 빈으로 이해하시면 됩니다. 즉, JPA 관련 트랜잭션을 담당합니다. 그리고 무엇을 가지고 트랜잭션을 수행하는 가를 알리기 위해서 persistence.xml 에서 정의한 persistenceUnitName 을 넣어주면 됩니다.

그 다음으로 보실 부분이 backingMapPluginCollections 이고요 해당 backingMap 에 Plugin 을 연결하는 설정입니다. 여기서는 JPALoader 라고 미리 만들어져서 제공되는 Loader Plugin 을 연결하는 설정입니다. 이전에 언급한 것처럼 Loader 는 WXS 에 저장된 캐시 객체와 DB 의 동기화 작업 별도 작업없이 자동으로 수행하는 역할을 하는 plugin 입니다.

중요 부분 JPAObjectGridDeployment.xml

```

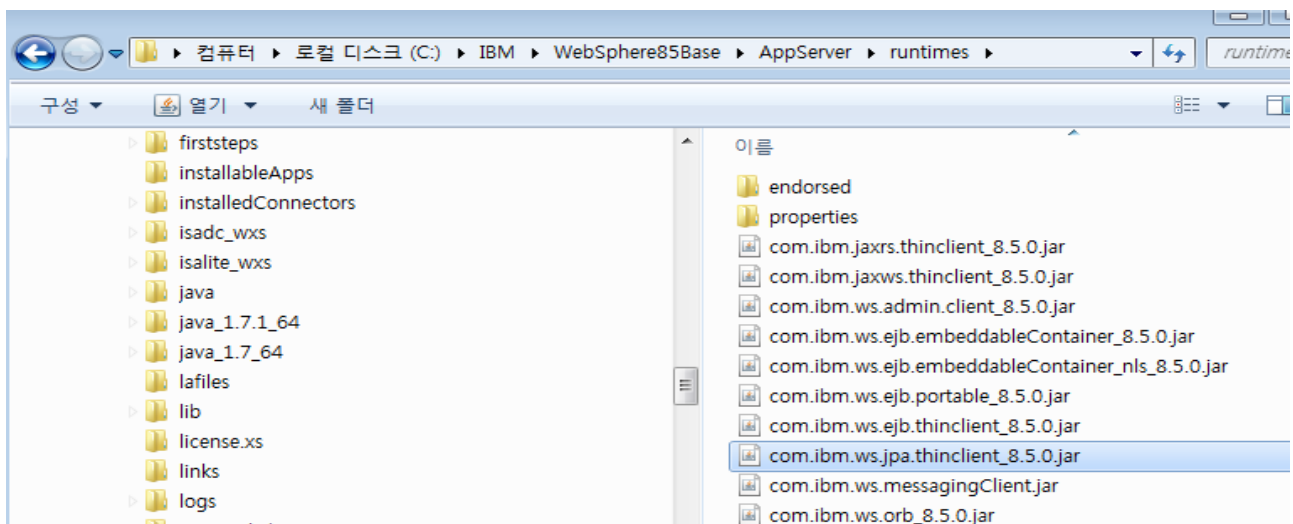
<objectgridDeployment objectgridName="PersonGrid">
    <mapSet      name="PersonMapSet"      numberOfPartitions="5"      minSyncReplicas="0"
maxSyncReplicas="1" developmentMode="true">
        <map ref="Person"/>
    </mapSet>
</objectgridDeployment>
</deploymentPolicy>

```

ObjectGridDeployment 설정 파일은 이전 강좌인 글로벌 캐시와 거의 비슷하므로 생략하겠습니다.

② IBM WXS 의 카탈로그 서버와 컨테이너 서버의 구동을 위한 script 준비

나머지 설정은 지난 강좌와 동일하게 사용하도록 하며 WXS 에서 OpenJPA 를 구동하기 위해서 IBM WAS 설치디렉토리/runtimes/com.ibm.ws.jpathinclient_8.5.0.jar 위치를 확인해 둡니다.



간단한 테스트를 위해서 제가 테스트시에 간단하게 만들었던 시작 스크립트를 첨부드리오니 참고하시기 바랍니다.

카탈로그 서버

```

startXsServer  cat1  -listenerPort  4809  -JMXServicePort  4899  -catalogServiceEndPoints
cat1:kr050578:6600:6601

```

카탈로그 서버 스크립트는 이전과 동일하지만 좀 더 명확하게 필요한 포트를 명시하기 위해서 서비스를 위한 JMXServicePort 를 지정했습니다.(지정하지 않으면 비어있는 포트 자동 할당)

컨테이너 서버

```

startXsServer          c1          -haManagerPort          5009          -objectGridFile
D:\W00.biz_work\W01.SWG\W06.WASv85\WASv855_guide\JPAObjectGrid.xml          -deploymentPolicyFile
D:\W00.biz_work\W01.SWG\W06.WASv85\WASv855_guide\JPAObjectGridDeployment.xml          -
catalogServiceEndpoints          kr050578:4809          -jvmArgs          -
javaagent:C:\IBM\WebSphere85Base\AppServer\runtimes\com.ibm.ws.jpa.thinclient_8.5.0.jar          -
javaagent:C:\IBM\WebSphere85WeXtremeScale\ObjectGrid\lib\wxssizeagent.jar          -cp
D:\W00.biz_work\W01.SWG\W09.SampleSource\WXS\PersonJPA.jar;D:\Weclipses\Weclipse_luna_wxs\utilLib\wb2jcc.jar

```

컨테이너 서버 스크립트도 이전과 동일하지만 마찬가지로 좀 더 명확하게 haManagerPort 를 명시적으로 지정했습니다. 해당 포트는 컨테이너가 살아있는지 확인하는 heartbeat 를 주고 받는 포트입니다. 마찬가지로 별도 지정하지 않으면 비어있는 포트가 자동으로 할당 됩니다.

그리고 또한, -javaagent: 옵션을 사용해서 OpenJPA 를 구동하기 위한 IBM 이 강화한 JPA thin client 라이브러리를 참조하고 마찬가지로 실제 JPA 프로젝트인 PersonJPA.jar 파일과 DB2 JDBC 드라이버 jar 를 classpath 에 포함하도록 되어 있습니다. 이는 standalone 으로 구동되는 WXS 가 OpenJPA 를 활용하여 실제 DB 랑 작업을 수행하기 때문에 참조하는 것입니다.(wxssizeagent.jar 성능 튜닝용이라 반드시 사용할 필요는 없습니다.)

이를 이용해서 WXS 의 카탈로그 서버와 컨테이너 서버를 구동하면 하단과 같은 결과를 확인할 수 있습니다.

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

```

C:\IBM\WebSphere85WeXtremeScale\ObjectGrid\bin>xscmd -c showMapSizes -cep kr050578:4809
시작 시간 :2015-01-30 15:18:08.499

CWXS10068I: 명령 실행 중: showMapSizes

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

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

총 카탈로그 서비스 도메인 수: 0 <0 B>

```

이에 추가하여 참고로 haManagerPort 만 변경하여 c2 라는 컨테이너 서버를 하나 더 기동하게 되면 하단과 같이 두 개의 컨테이너가 구동된 것을 확인할 수 있으며 보다 중요한 것은 하나의 컨테이너를 더 두게 되면 복제본이 같이 위치하는 것을 보실 수 있습니다.

(단, 해당 부분은 ObjectGridDeployment 설정의 developmentMode 가 true 이기 때문에 한대의 HW 만으로도 해당 부분을 확인할 수 있는 것입니다. 일반적으로 프로덕션 환경은 여러 대의 서버를 권장하고 WXS 도 마찬가지로 이므로 해당 설정이 아니라면 각 HW 별로 원본과 복제본이 분리해서 위치하게 됩니다.)

```
C:\W\IBM\WebSphere85\ExtremeScale\WObjectGrid\bin>xscmd -c showMapSizes -cep kr050578:4809
시작 시간 :2015-01-30 16:47:20.979

CWXS10068I: 명령 실행 중: showMapSizes

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

*** c2에 대한 맵 나열 ***
맵 이름 파티션 맵 항목 사용한 바이트 샤드 유형 컨테이너
-----
Person 0 0 0 SynchronousReplica c2_C-0
Person 1 0 0 SynchronousReplica c2_C-0
Person 2 0 0 SynchronousReplica c2_C-0
Person 3 0 0 SynchronousReplica c2_C-0
Person 4 0 0 SynchronousReplica c2_C-0
서버 총계: 0 <0 B>

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

현재 컨테이너에 원본과 복제본이 몰아져 있는데 시간이 지나면 최적화 알고리즘에 의해 자동으로 밸런싱 됩니다. 또는 하단과 같이 수동으로 밸런싱 명령을 수행하는 방안을 통해 밸런싱 가능합니다.

xscmd -c balanceShardTypes -cep kr050578:4809

```
CWXS10068I: 명령 실행 중: showMapSizes

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

*** c2에 대한 맵 나열 ***
맵 이름 파티션 맵 항목 사용한 바이트 샤드 유형 컨테이너
-----
Person 0 0 0 Primary c2_C-0
Person 1 0 0 SynchronousReplica c2_C-0
Person 2 0 0 Primary c2_C-0
Person 3 0 0 SynchronousReplica c2_C-0
Person 4 0 0 SynchronousReplica c2_C-0
서버 총계: 0 <0 B>

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

7) IBM WXS 테스트

작성된 애플리케이션을 WAS 에 모두 배포한 후에 해당 애플리케이션에서 WXS 의 Grid 를 조회하는 호출을 수행해보면 하단과 같이 정상으로 결과가 나오는 것을 확인할 수 있습니다. 뿐만 아니라 연속적 호출을 통해서 WXS 의 Grid 로 캐시된 데이터를 반환하는 두 번째 호출이 첫 번째 호출에 비해 성능이 상당히 좋아진 것을 확인할 수 있습니다.

localhost:9080/SearchJPA/SearchPersonGrid?id=7&command=search

Most Visited IBM IBM Information Cent... IBM WebSphere eXtreme ... IBM WAIT

User ID:

ID:	7
Name:	JungWoon Lee
Date Joined:	Mon Jan 12 00:00:00 KST 2015
Spend Time:	1096.383

localhost:9080/SearchJPA/SearchPersonGrid?id=7&command=search

Most Visited IBM IBM Information Cent... IBM WebSphere eXtreme ... IBM WAIT

User ID:

ID:	7
Name:	JungWoon Lee
Date Joined:	Mon Jan 12 00:00:00 KST 2015
Spend Time:	6.325

뿐만 아니라 다시한번 xscmd 명령을 이용해서 Grid 내부를 살펴보면 하단과 같이 실제로 DB 의 데이터가 향후 캐시로 사용하기위해 Grid 에 저장된 것을 확인할 수 있습니다.

xscmd -c showMapSizes -cep kr050578:4809

```
관리자: C:\Windows\System32\cmd.exe
C:\IBM\WebSphere85\WextremeScale\WObjectGrid\bin>xscmd -c showMapSizes -cep kr050578:4809
시작 시간 :2015-01-29 23:55:46.551
CWXS10068I: 명령 실행 중: showMapSizes
*** PersonGrid 데이터 그리드 및 PersonMapSet 맵 세트에 대한 결과 표시.
*** c1에 대한 맵 나열 ***
맵 이름 파티션 맵 항목 사용한 바이트 샤드 유형 컨테이너
-----
Person 0 0 0 Primary c1_C-0
Person 1 0 0 Primary c1_C-0
Person 2 1 328 B Primary c1_C-0
Person 3 0 0 Primary c1_C-0
Person 4 0 0 Primary c1_C-0
서버 총계: 1 <328 B>
총 카탈로그 서비스 도메인 수: 1 <328 B>
<사용된 바이트 통계는 단순 오브젝트 또는 COPY_TO_BYTES 복사 모드를 사용 중인 경우에만 정확합니다.>
```

추가적으로 하단과 같은 명령을 이용해서 실제 저장된 객체를 직접 확인하는 것도 가능합니다.

xscmd -c findByKey -g PersonGrid -m Person -fs "*" -rv -cep kr050578:4809

```
C:\WIBM\WebSphere85\WeXtremeScale\WObjectGrid\bin>xscmd -c findByKey -g PersonGrid
-m Person -fs "*" -rv -cep kr050578:4809
시작 시간 :2015-01-29 23:58:46.663

CWXS10068I: 명령 실행 중: findByKey

1 일치 키를 찾았습니다.

파티션 키 값
----- --
2      7  com.ibm.juwlee.jpa.Person@f6511fef

CWXS10040I: findByKey 명령이 완료되었습니다.

종료 시간: 2015-01-29 23:58:50.560

C:\WIBM\WebSphere85\WeXtremeScale\WObjectGrid\bin>
```

이렇게 하면 WXS 를 활용한 In-Memory DataGrid 구성 및 테스트가 완료된 것입니다.

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

WebSphere eXtreme Scale v8.6 Key Concepts and Usage Scenarios

<http://www.redbooks.ibm.com/abstracts/sg247683.html?Open>

WebSphere eXtreme Scale Best Practices for Operation and Management

<http://www.redbooks.ibm.com/abstracts/sg247964.html?Open>