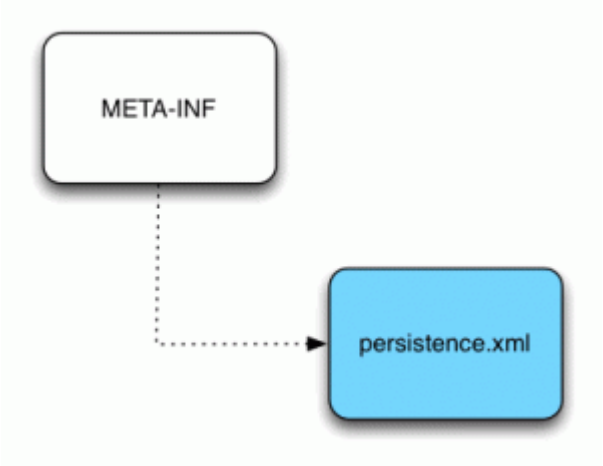
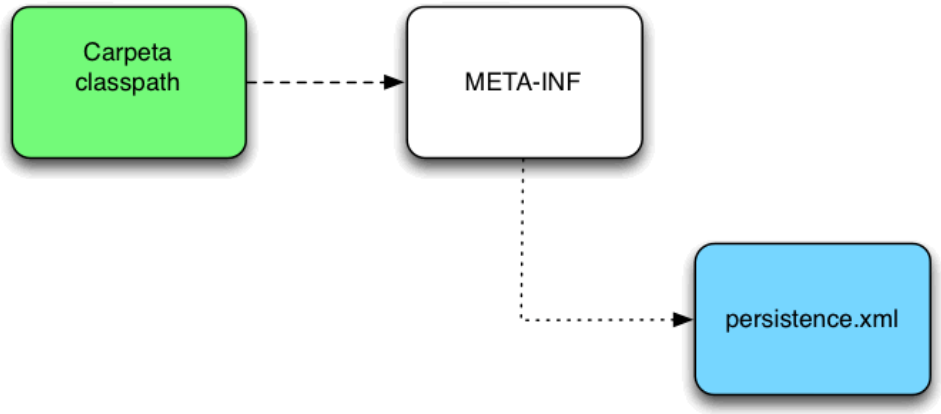
Antes de ayer Carlos Oliva un lector de mi blog me preguntó si habría forma de configurar de una manera mas flexible el fichero persistence.xml. Me pareció una pregunta interesante ya que aunque muchas veces con la configuración de JPA por defecto nos es suficiente, existen otras situaciones en las que podemos necesitar una mayor flexibilidad a nivel de persistence.xml. Este fichero se suele ubicar en la META-INF de nuestra aplicación por lo menos es lo que el standard recomienda.



Ahora bien ¿puede estar el fichero persistence.xml en algún otro sitio?

Persistence.xml y classpath

La respuesta es SI el fichero de persistence.xml se puede ubicar en cualquier carpeta que pertenezca al classpath siempre que este ubicado dentro de una carpeta META-INF.



ORM.XML

De esa forma podemos ubicar el fichero de persistencia en otras rutas que nos resulten mas interesantes .Ahora bien normalmente esto no es suficiente .Cuando se precisa flexibilidad a nivel de JPA es habitual recurrir al fichero orm.xml .¿Para que sirve este fichero? . Bueno sirve para muchas cosas pero quizás su característica principal es que se usa para eliminar las anotaciones de JPA del código y substituirlas por etiquetas XML . Vamos a ver un ejemplo sencillo de este fichero.

```
&amp;amp;lt;?xml version=&quot;1.0&quot;
encoding=&quot;UTF-8&quot;?&amp;amp;gt;
&amp;amp;lt;entity-mappings
xmlns=&quot;http://java.sun.com/xml/ns/persistence/orm&quot;
xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
xsi:schemaLocation=&quot;http://java.sun.com/xml/ns/persistence/orm
http://java.sun.com/xml/ns/persistence/orm_2_0.xsd"
version=&quot;2.0&quot;&amp;amp;gt;

&amp;amp;lt;package&amp;amp;gt;
```

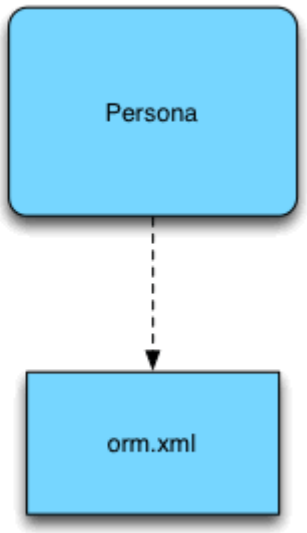
```
com.arquitecturajava
&amp;amp;lt;/package&amp;amp;gt;

&amp;amp;lt;entity class=&quot;Persona&quot;&amp;amp;gt;
&amp;amp;lt;attributes&amp;amp;gt;
&amp;amp;lt;id
name=&quot;nombre&quot;&amp;amp;gt;&amp;amp;lt;/id&amp;amp;gt;
&amp;amp;lt;basic
name=&quot;edad&quot;&amp;amp;gt;&amp;amp;lt;/basic&amp;amp;gt;
&amp;amp;lt;/attributes&amp;amp;gt;
&amp;amp;lt;/entity&amp;amp;gt;

&amp;amp;lt;/entity-mappings&amp;amp;gt;
```

Como podemos ver el fichero incluye los mismos conceptos elementales de JPA pero en vez de a traves de anotaciones a traves del uso de etiquetas xml . Esta solución aporta un enfoque “diferente” ya que las clases java no hará falta anotarlas y quedarán mucho mas limpias .





Ahora bien para mapear correctamente el fichero orm.xml .Este debe estar ubicado en el classpath y ser referenciado desde el persistence.xml a traves de la etiqueta <mapping-file></mapping-file>

```
&amp;amp;lt;persistence
xmlns=&quot;http://java.sun.com/xml/ns/persistence&quot;
xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
xsi:schemaLocation=&quot;http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_2_0.xsd"
version=&quot;2.0&quot;&amp;amp;gt;
&amp;amp;lt; persistence-unit
name=&quot;UnidadPersonas&quot;&amp;amp;gt;
&amp;amp;lt; mapping-file&amp;amp;gt; orm.xml&amp;amp;lt;  /mapping-
file&amp;amp;gt;
&amp;amp;lt;properties&amp;amp;gt;
&amp;amp;lt;property name= &quot;hibernate.show_sql&quot;
value=&quot;true&quot; /&amp;amp;gt;
&amp;amp;lt;property name=&quot;hibernate.dialect&quot;
value=&quot;org.hibernate.dialect.MySQLDialect&quot; /&amp;amp;gt;
&amp;amp;lt;property name=&quot;javax.persistence.jdbc.driver&quot;
value=&quot;com.mysql.jdbc.Driver&quot; /&amp;amp;gt;
&amp;amp;lt;property name=&quot;javax.persistence.jdbc.user&quot;
value=&quot;root&quot; /&amp;amp;gt;
&amp;amp;lt;property name=&quot;javax.persistence.jdbc.password&quot;
value=&quot;jboss&quot; /&amp;amp;gt;
&amp;amp;lt;property name=&quot;javax.persistence.jdbc.url&quot;
value=&quot;jdbc:mysql://localhost/jpa&quot; /&amp;amp;gt;

&amp;amp;lt;/properties&amp;amp;gt;
```

```
&amp;amp;lt;/persistence-unit&amp;amp;gt;

&amp;amp;lt;/persistence&amp;amp;gt;
```

Vamos a ver el código fuente de la clase Persona para ver como queda en el caso de que nos apoyemos en el fichero orm.xml

```
package com.arquitecturajava;
public class Persona {

private String nombre;
private int edad;
public Persona() {
super();

}

public Persona(String nombre, int edad) {
super();
setEdad(edad);
setNombre(nombre);
}
public String getNombre() {

return nombre;
}
public void setNombre(String nombre) {
this.nombre = nombre;

}
```

```
public int getEdad() {

return edad;
}
public void setEdad(int edad) {
this.edad = edad;

}

}
```

Podemos ver que ya no necesita de ningún tipo de anotación y se convierte en un objeto de negocio clásico . Creada la clase es momento de ver como la podríamos usar en un programa habitual de JPA que realice una inserción a traves del método persist.

```
package com.arquitecturajava;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class Principal01Add {

public static void main(String[] args) {

Persona yo = new
Persona(&amp;amp;amp;quot;pedro&amp;amp;amp;quot;,25);
EntityManagerFactory emf =
Persistence.createEntityManagerFactory(&amp;amp;amp;quot;UnidadPersona
s&amp;amp;amp;quot;);
```

```
EntityManager em = emf.createEntityManager();
try {
em.getTransaction().begin();

em.persist(yo);
em.getTransaction().commit();
} catch (Exception e) {

e.printStackTrace();
}finally {
em.close();

}

}

}
```

El objeto persistirá en la base de datos sin problemas



#	nombre	edad
1	pedro	25
*	NULL	NULL

Hemos terminado de construir un ejemplo de JPA que se apoya en el fichero orm.xml para realizar todos los mareos. Normalmente suele ser mas cómodo usar anotaciones sin embargo si necesitamos una gran flexibilidad en el desarrollo de nuestras aplicaciones es una buena idea utilizar el fichero orm.xml para ello.

Descargar :cursoJpa01ORM