Mecanismos de Almacenamiento

Programación Web-Dinámico

Dr. Carlos Roberto Jaimez González

Almacenamiento para Objetos

- Necesidad de guardar el estado de un objeto (es decir, los valores de sus propiedades/atributos) para uso futuro.

- En una aplicación web el estado de un objeto se pierde cuando el servidor de aplicaciones o servidor web (Tomcat) es reiniciado. Es por esto que necesitamos un mecanismo para almacenamiento de datos.

- Algunos ejemplos de mecanismos de almacenamiento en la siguiente diapositiva.

Diferentes Mecanismos

- Sistema de archivos. Mediante código escrito manualmente, o con algún mecanismo de serialización de objetos, tal como WOX – Web Objects in XML.

- Base de Datos Relacional (JDBC, ODBC). Existe una gran cantidad de RDBMS, tales como MySQL, Oracle, SQL Server, MS Access, etc.

- Base de Datos Orientada a Objetos. Ejemplos: ObjectStore, FastObjects, db4o.

- Base de Datos XML.

Sistema de Archivos

- Sencillo de utilizar para objetos simples.
- Puede resultar tedioso.
- Es necesario sincronizar los cambios para leer/escribir el estado del objeto; lo cual puede llegar a ser difícil de manejar.

Sistema de Archivos (continuación)

- Difícil de realizar consultas (queries) sobre los archivos.
- Tiene usos muy particulares (por ejemplo, logs).
- El uso de serializadores XML, tal como WOX, resuelven algunos de estos problemas. ¿Cuáles?

Base de Datos Relacional

- Utilizada ampliamente.

- No es un buen mecanismo de almacenamiento para objetos. Es necesario realizar un mapeo del modelo orientado a objetos al modelo relacional.

- Existen herramientas, tales como Hibernate, para realizar el mapeo oo-relacional. Generan código con queries como el siguiente:

SELECT * FROM Student WHERE name='Carlos'

Base de Datos Relacional (continuación)

- ¿Qué sucede si no existe el campo en la base de datos, que corresponde a la propiedad/atributo del objeto?

  - Errores no se encuentran en tiempo de compilación

  - Error de aplicación: SQL Exception...

Base de Datos Orientada a Objetos

- Potencialmente la solución ideal para guardar objetos, ya que los almacena directamente en la BDOO.

- No se requiere ningún mapeo especial.

- Almacena grandes objetos con una sola línea de código (ver ejemplo).

Base de Datos Orientada a Objetos (continuación)

- En la práctica, existen algunos problemas interesantes; tales como integridad referencial, transacciones, etc.

- Han comenzado a usarse comercialmente.

- ¿Algún día las BD Orientadas a Objetos rebasarán a las BD Relacionales?

Bases de Datos OO (viejo estilo)
ObjectStore, FastObjects

- Se adhieren estrictamente a los principios oo.

- Refuerzan la integridad referencial. Un objeto no puede ser borrado si existen objetos que hacen referencia a él.

- Refuerzan transacciones. Los objetos sólo pueden ser accesados dentro de una transacción. Esto tiene un efecto en cascada, de tal forma que los objetos referenciados por el objeto raíz, no podrían ser accesados mientras el objeto raíz está siendo accesado.

- Difícil hacerlo funcionar en Java, ya que las clases necesitan ser modificadas (al menos hace algunos años!)

Bases de Datos OO ligera db4objects

- Abandona todos los principios mencionados en la dispositiva anterior.

- El programador tiene más responsabilidades:
  - Cuidado con el manejo de objetos
  - Base de datos es más simple de usar
  - Base de datos es mucho más simple de implementar

- URL para descargar db4o:
http://www.db4o.com/DownloadNow.aspx
Versiones para Java & .NET

Ideas principales en db4o

- La BD es representada como un ObjectContainer:

```
ObjectContainer db;
```

- El ObjectContainer es utilizado para guardar o actualizar el estado de un objeto (`Object o`):

```
db.set(o);
```

- Para recuperar todos los objetos de una clase `x`:

```
List objetos = db.get(x);
List objetos = db.get(Producto.class);
```

Ideas principales en db4o

- Query by Example (QBE)

```
List objetos = db.get(new Producto("Sony"));
```

- Las consultas también se pueden ejecutar proporcionando un Predicado (`Predicate p`)

```
List objects = db.query(p);
```

- Para borrar un objeto (`Object o`)

```
db.delete(o);
```

Guardando objetos con db4o

- `db.set(o)` guarda todos los objetos que son alcanzables desde `o`.

- Pero, si uno de los objetos alcanzables cambia, entonces llamando `db.set(o)` de nuevo, no guardará el cambio!

- El algoritmo para guardar un objeto es el siguiente:
  - Para cada objeto alcanzable desde la raíz (`o`)
    Guárdalo si aún no ha sido guardado

Ejemplo: Tienda Virtual

- En este ejemplo presentaremos una tienda virtual que veremos durante nuestras clases. La tienda vende productos electrodomésticos.

- Cada producto tiene las siguientes propiedades o atributos: id, nombre, descripción, precio, e imagen.

- Cada producto será un objeto de la clase `Producto`.

- En nuestro ejemplo veremos como guardar este tipo de objetos en la BD Orientada a Objetos. También veremos como escribir consultas (queries) que recuperen una lista de productos, y un producto en particular.

Clase Producto

Esta es la clase que representará los objetos que serán guardados en db4o:

```
public class Producto {
    private int id;
    private String nombre;
    private String descripcion;
    private double precio;
    private String imagen;

    // los constructores,
    // los métodos get/set para cada atributo, y
    // el método toString() son omitidos
}
```

Interface DBInterface

- `DBInterface` es la interface que definirá los métodos que podrán ser invocados desde la aplicación. Para nuestro ejemplo sólo tendremos tres métodos.

```
public interface DBInterface {

    public Producto recuperaProducto(int id);

    public ArrayList recuperaProductos();

    public void insertaProducto(Producto producto);
}
```

Clase DBObject: Implementación de DBInterface

- `DBObject` es la implementación de la interface `DBInterface`. La clase `DBObject` implementará los tres métodos definidos en `DBInterface`.

```
public class DBObject implements DBInterface{

    public Producto recuperaProducto(int id){
        ...  ...
    }
    public ArrayList recuperaProductos(){
        ...  ...
    }
    public void insertaProducto(Producto producto){
        ...  ...
    }
}
```

Guardando algunos productos

```
public void insertaProducto(Producto producto){
    Object Container db = Db4o.openFile(dbName);
    //inserta el producto en la BD
    db.set(producto);
    System.out.println("Producto insertado...");
    db.close();
}
```

Para llamar al método `insertaProducto`:

```
public static void main(String[] args) {
    DBObject db = new DBObject();
    db.insertaProducto(new Producto (6, "Horno",
       "Descripción horno", 1599, "horno.jpg"));
    db.insertaProducto(new Producto (7, "Plancha",
       "Descripción plancha", 560, "plancha.jpg"));
}
```

Recuperando todos los productos

```
public ArrayList recuperaProductos(){
   ObjectContainer db = Db4o.openFile(dbName);
   ArrayList listaProductos = new ArrayList();

   //recupera todos los productos
   ObjectSet res=db.queryByExample(Producto.class);

   //iteramos sobre el ObjectSet para sacar cada
   //objeto Producto y lo añadimos a la lista de
   //productos que vamos a regresar
   while(result.hasNext()) {
      Producto prod = (Producto)result.next();
      listaProductos.add(prod);
   }

   db.close();
   return listaProductos;
}
```

Recuperando un producto particular

```
public Producto recuperaProducto(int idProducto){
    ObjectContainer db = Db4o.openFile(dbName);
    Producto prod = null;

    //proporcionamos el patrón para recuperar
    //el objeto que deseamos, y utilizamos QBE
    Producto producto = new Producto(idProducto,
                        null, null, 0.0, null);
    ObjectSet result = db.queryByExample(producto);

    if (result.hasNext()) {
        prod = (Producto)result.next();
    }

    db.close();
    return prod;
}
```

JSP que muestra todos los productos (fragmento 1 de 2)

```
<body>
    <h1>Tienda Virtual</h1>
    <table border="1">
        <tr>
            <td>Imagen</td>
            <td>Id</td>
            <td>Nombre</td>
            <td>Descripci&oacute;</td>
            <td>Precio</td>
        </tr>
    <%
    DBInterface db = new DBObject(ruta);
    ArrayList lista = db.recuperaProductos();
    Iterator it = lista.iterator();
    while (it.hasNext()){
        Producto prod = (Producto)it.next();
    %>
```

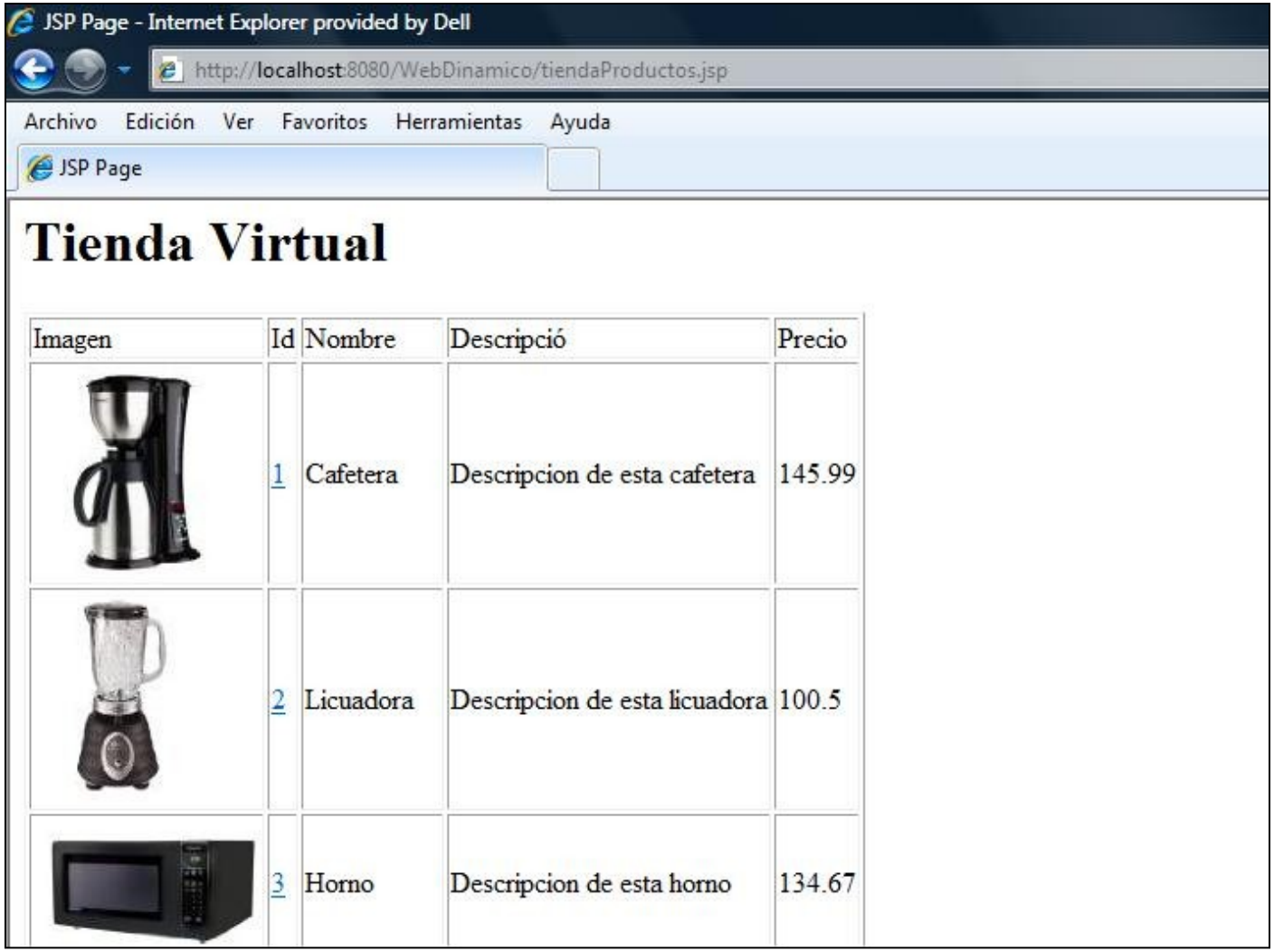
JSP que muestra todos los productos (fragmento 2 de 2)

```
  <tr>
    <td><img src="<%=prod.getImagen()%>" /></td>
    <td>
      <a href="showProd.jsp?prodId=<%=prod.getId()%>">
      <%=prod.getId()%></a>
    </td>
    <td><%=prod.getNombre()%></td>
    <td><%=prod.getDescripcion()%></td>
    <td><%=prod.getPrecio()%></td>
  </tr>

  <%
  }
  %>
</table>

</body>
```

JSP que muestra todos los productos

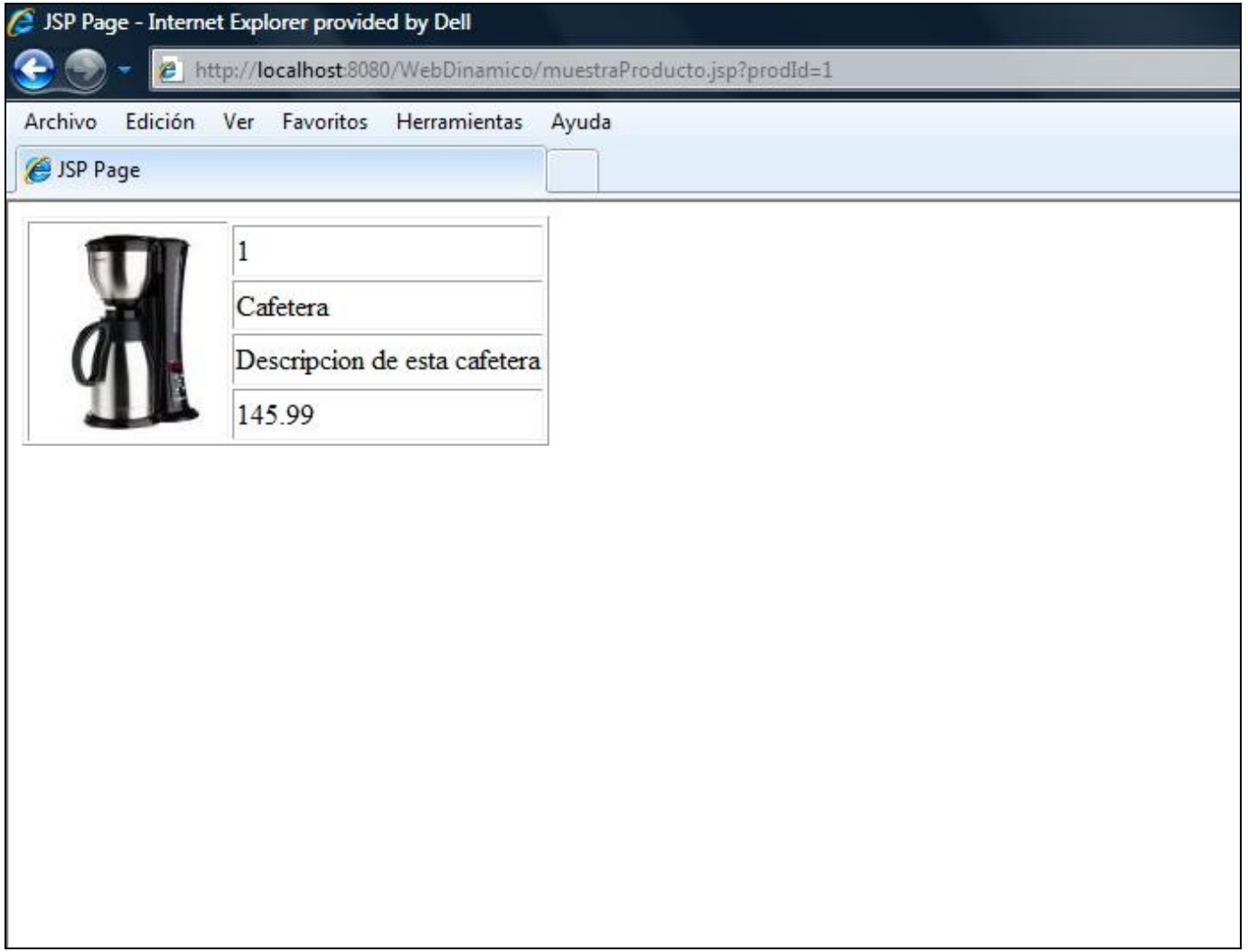


JSP que muestra un producto particular

```
<body>
   <table border="1">
   <%
   String prodId = request.getParameter("prodId");
   int id = Integer.parseInt(prodId);

   DBInterface db = new DBObject(ruta);
   Producto prod = db.recuperaProducto(id);
   %>
      <tr><td rowspan="5">
        <img src="<%=prod.getImagen()%>"/></td>
      </tr>
      <tr><td><%=prod.getId()%></td></tr>
      <tr><td><%=prod.getNombre()%></td></tr>
      <tr><td><%=prod.getDescripcion()%></td></tr>
      <tr><td><%=prod.getPrecio()%></td></tr>
   </table>
</body>
```

JSP que muestra un producto particular



Resumen

Hemos cubierto lo siguiente:

- Diferentes mecanismos que existen para almacenamiento de objetos: Sistema de Archivos, BD Relacional, DB Orientada a Objetos, BD XML.

- Principales ideas detrás de la DBOO db4o; principales ventajas y desventajas.

Resumen

- Como guardar, actualizar, borrar y recuperar objetos de diferentes formas (QBE, Predicados, etc.).

- Ejemplo de tienda virtual de productos electrodomésticos; con código de clases y Java Server Pages.