Universidad Politécnica de Tecámac

Programación para móviles II

1922IS

9no Cuatrimestre

Alvarez Esperon Neri Alejandro

López Bautista Alfredo

Pérez Ponce Anahí

Emmanuel Torres Servin

Contenido

COMPRENSIÓN DE LA INFORMACIÓN, PROBLEMATICA Y COMPONENTES DEL CASO, EL ALUMNO: (ED)

Explica que es el modelado de acceso a datos

Un Objeto de Acceso a Datos o Data Access Object (DAO) son una serie de objetos que le permiten tener acceso y manipular datos mediante programación en bases de datos locales o remotos. Puede utilizar DAO para administrar bases de datos, así como sus objetos y su estructura.

Es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo. El término se aplica frecuentemente al Patrón de diseño Object.

Los Objetos de Acceso a Datos son un Patrón de Diseño Core J2EE y considerados una buena práctica. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

Los Objetos de Acceso a Datos pueden usarse en Java para aislar a una

Un modelo de acceso es un perfil de los usuarios que necesitan acceder al sistema Netezza y los permisos o tareas que necesitan.

Por lo general, un modelo de acceso comienza de forma modesta con pocos usuarios o grupos, pero a menudo crece y evoluciona a medida que que se añaden usuarios nuevos al sistema. El modelo define los usuarios, sus roles y los tipos de tarea que ejecutan o las bases de datos a las que requieren acceso.

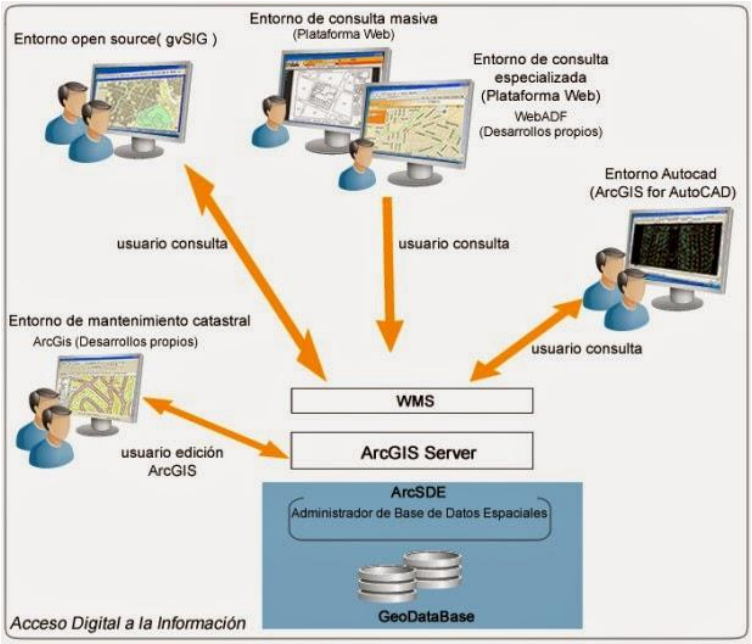
Los modelos de acceso pueden variar significativamente para cada compañía y entorno. Como ejemplo básico, puede desarrollar un modelo de acceso que defina tres grupos iniciales de usuarios de base de datos:

- Administradores: Usuarios que se encargan de gestionar diversas tareas y servicios. Pueden gestionar bases de datos concretas, gestionar accesos de usuario, crear bases de datos, cargar datos o realizar copias de seguridad y restaurar bases de datos.
- Usuarios generales de base de datos: Usuarios que tienen permiso para realizar consultas a una o varias bases de datos y que pueden tener o no acceso a la gestión de objetos en la base de datos. Estos usuarios pueden tener también un nivel de prioridad más bajo para su trabajo.
- Usuarios de bases de datos avanzados: Usuarios que requieren acceso a bases de datos críticas y que pueden utilizar consultas SQL más complejas que los usuarios generales. Estos usuarios pueden necesitar un nivel de prioridad más alto para su trabajo. Pueden tener también permisos para tareas como la creación de objetos de bases de datos, ejecución de objetos definidos por el usuario (UDXs) o la carga de datos.

El modelo de acceso sirve como plantilla para los usuarios y grupos que cree y ofrece también un mapa de necesidades de permisos de acceso. Si crea grupos de bases de datos de Netezza para representar estos roles o conjuntos de permisos, podrá asignar de forma fácil usuarios a los grupos para que hereden los distintos permisos, y podrá cambiar todos los usuarios con un rol cambiando únicamente los permisos de grupo o mover usuarios de un rol a otro cambiando sus grupos y añadiéndolos a los grupos que controlan esos permisos.

Explica que es y cómo funciona la manipulación de datos en dispositivos móviles

En software de computadores, un Data Access Object (DAO, Objeto de Acceso a Datos) es un componente de software que suministra una interfaz común entre la aplicación y uno o más dispositivos de almacenamiento de datos, tales como una Base de datos o un archivo.



VENTAJAS:

Los Objetos de Acceso a Datos son un Patrón de los subordinados de Diseño Core J2EE y considerados una buena práctica. La ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

Los Objetos de Acceso a Datos pueden usarse en Java para aislar a una aplicación de la tecnología de persistencia Java subyacente (API de Persistencia Java), la cual podría ser JDBC, JDO, Enterprise JavaBeans, TopLink, EclipseLink, Hibernate, iBATIS, o cualquier otra tecnología de persistencia.

Usando Objetos de Acceso de Datos significa que la tecnología subyacente puede ser actualizada o cambiada sin cambiar otras partes de la aplicación.

JDBC: Java Database Connectivity, más conocida, es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Otra ventaja de usar objetos de acceso a datos es que cualquier objeto de negocio (aquel que contiene detalles específicos de operación o aplicación) no requiere conocimiento directo del destino final de la información que manipula.

DESVENTAJAS:

La flexibilidad tiene un precio. Cuando se añaden DAOs a una aplicación, la complejidad adicional de usar otra capa de persistencia incrementa la cantidad de código ejecutado durante tiempo de ejecución. La configuración de las capas de persistencia requiere en la mayoría de los casos mucho trabajo.

Transact-SQL proporciona sintaxis no especializada para las instrucciones INSERT, UPDATE o DELETE cuando se modifican datos de columnas de tipos definidos por el usuario (UDT). Las funciones CAST o CONVERT de Transact-SQL se utilizan para convertir tipos de datos nativos al tipo definido por el usuario.

Describe lo que es y conlleva la persistencia de datos en los dispositivos móviles

Persistencia en Android: Preferencias Compartidas o Shared Preferences

Con Shared Preferences podemos almacenar y recuperar en el formato clave-valor información como texto, booleanos y números; lo que lo convierte en potencial para almacenar configuraciones del usuario como: estilos, preferencias, etc.

Los modos de acceso posibles son:

- MODE_PRIVATE: Sólo nuestra aplicación tiene acceso a estas preferencias.
- MODE_WORLD_READABLE: Todas las aplicaciones pueden leer estas preferencias, pero sólo la nuestra puede modificarlas (deprecated desde el API 17).
- MODE_WORLD_WRITEABLE: Todas las aplicaciones pueden leer y modificar estas preferencias (deprecated desde el API 17).

Su uso es sencillo; si queremos recuperar una "preferencia" que hemos llamado "MiPreferencia" basta con emplear el siguiente código:

SharedPreferences preferencia =

getSharedPreferences("MiPreferencia",Context.MODE_PRIVATE);

Para obtener la información de la preferencia:

SharedPreferences preferencia =

boolean = preferencia.getBoolean("habilitar_imagenes", true);

Los parámetros consisten en:

- El nombre de la preferencia que queremos recuperar.
- Valor por defecto que será utilizado si la preferencia no existe en el sistema.

Para guardar o modificar información de la preferencia:

SharedPreferences preferencia =

boolean = preferencia.putBoolean("habilitar_imagenes", false);

Explica qué y cuáles son los mecanismos de tolerancia a fallos

La tolerancia a fallos es la propiedad que le permite a un sistema seguir funcionando correctamente en caso de fallo de uno o varios de sus componentes. Si disminuye su calidad de funcionamiento, la disminución es proporcional a la gravedad de la avería, en comparación con un sistema diseñado ingenuamente de forma que hasta un pequeño fallo puede causar el colapso total del sistema. Tolerancia a fallos es particularmente buscado en sistemas de alta disponibilidad.

Un diseño tolerante a fallos es un sistema que está capacitado para continuar su funcionamiento cuando algún componente del sistema falla.,1 posiblemente a un nivel más reducido, lo que es mejor a que el sistema falle completamente. El término es comúnmente usado para describir sistemas basados en computadoras diseñados para continuar en mayor o menor medida las operaciones que realiza con, a lo mejor, una reducción de su rendimiento o un incremento de los tiempos de respuesta en las componentes que fallan. Esto significa que el sistema, dada una falla de software o de hardware no se detiene. Un ejemplo en otra rama es el de un

automóvil diseñado para continuar su funcionamiento si uno de sus neumáticos recibe un pinchazo.

Tolerancia a fallos es sólo una propiedad de cada una de las máquinas, sino que también puede caracterizar las reglas según las cuales interactúan. Por ejemplo, el protocolo TCP está diseñado para permitir una comunicación fiable de dos sentidos en una red de conmutación de paquetes, incluso en la presencia de enlaces de comunicaciones que son imperfectos o sobrecargados. Esto es así debido a que en los extremos de la comunicación se puede esperar pérdida de paquetes, la duplicación, la reordenación y la corrupción, a fin de que estas condiciones no dañen la integridad de los datos, y sólo reduzcan la capacidad de una cantidad proporcional.

Codificación

Clase palabras

```java
public class palabras implements Serializable {
    private String cadena;
    private boolean existe = false; //comprobación si palabra tiene caracter introducido
    public boolean palabraValida = false; // validación si la palabra se ha validado
    public int contador = 0,contador2; // contador de letras acertadas
    public List<String> posiciones = new ArrayList<>();
// convierte la cadena a letras mayusculas
    public palabras(String cadena) { this.cadena = cadena.toUpperCase(); }
//proceso para hacer lista auxiliar donde se convierte la cadena en arreglo char y se agregan guiones a la lista posiciones por cada caracter existente
    public void llenarList() {
        char[] aux = cadena.toCharArray();
        for (int x = 0; x < aux.length; x++)
            posiciones.add("_");
    }
    public String validar(String intento) {
        // lista de retorno, donde se almacenan los indices donde se encontraron caracteres iguales
        char[] aux = cadena.toCharArray(); // variable auxiliar se convierte la cadena en un arreglo char
        // inicio expresión regular comparación
        Pattern pat = Pattern.compile(intento);
        String input = cadena;
        Matcher mat = pat.matcher(input);
        if (mat.find()) {
            existe = true;
        } else {
            existe = false;
        }
```

```
try {
    if (existe == true) { // Si coincide con caracter se inicia proceso para buscar su posición en la frase
        for (int i = 0; i < aux.length; i++) {
            char auxiliar = intento.charAt(0);
            if (aux[i] == auxiliar) {
                posiciones.set(i, aux[i] + "");
                contador++;
            }
        }
    }

    if (contador == aux.length) { // cuando se hayan encontrado todas las letras se convierte en true
        for (int i = 0; i < aux.length; i++) {
            String temp = posiciones.get(i);
            String temp2=aux[i]+"";
            if (temp.equals(temp2)) {
                contador2++; // contador para letras encontradas
            }else{
                contador--;
            }
        }
    }
```

```
      if(contador2==aux.length){
          /* Si contador2 es igual al numero de caracteres de la palabra
          entonces se da por entendido que el jugador ganó, se redeclaran las variables y se valida palabraValidada*/
          palabraValida=true;
          contador2=0;
          contador=0;
      }else{
          /*Caso contrario se reinicia contador 2*/
          contador2=0;

      }
    } catch (Exception e) {
        palabraValida = false;
    }
    String auxCadena = "";
    // foreach para concatenar posiciones y retornar auxCadena
    for (String pos : posiciones) {
        auxCadena += " " + pos + " ";
    }
    return auxCadena;
}
```

```
// getter & setters
    public String getCadena() {
        return cadena;
    }

    public void setCadena(String cadena) {
        this.cadena = cadena;
    }

    public boolean isExiste() {
        return existe;
    }

    public void setExiste(boolean existe) {
        this.existe = existe;
    }

    public boolean isPalabraValida() {
        return palabraValida;
    }

    public void setPalabraValida(boolean palabraValida) {
        this.palabraValida = palabraValida;
    }
}
```

MainActivity

```java
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // variables de interfaz
    siguiente = (Button) findViewById(R.id.btnSiguiente);
    palabra = (EditText) findViewById(R.id.etPalabra);
    siguiente.setOnClickListener(new View.OnClickListener() { // función OnClick para botón siguiente
        @Override
        public void onClick(View view) {
            if (palabra.getText().toString().equals(null) || palabra.getText().toString().equals("")) { // valida si se introdujo alguna palabra
                Toast toast = Toast.makeText(getApplicationContext(), text: "introduzca una palabra!!!", Toast.LENGTH_SHORT);
                toast.show();
                return;
            } else if(palabra.getText().toString().length()>1){
                //envia a MainActivity2 y se envia un extra "dato" del valor de palabras2
                palabras palabras2 = new palabras(palabra.getText().toString());
                Intent i = new Intent( packageContext: MainActivity.this, MainActivity2.class);
                i.putExtra( name: "dato", palabras2);
                startActivity(i);
            }else {
                // valida que la palabra tenga minimo 2 caracteres
                Toast toast = Toast.makeText(getApplicationContext(), text: "introduzca una palabra con mas de 1 caracter!!!", Toast.LENGTH_SHORT);
                toast.show();
                return;
            }

        }
```

MainActivity2

```java
public class MainActivity2 extends AppCompatActivity {
    public palabras palabra;
    public TextView txtPalabra;
    public Button btnComprobar;
    public EditText edtCaracter;
    public ImageView image; // imagen para mostrar ahorcado
    public int Intentos = 0; // contador de intentos para validar palabras

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        // inicialización de variables para interfaz
        txtPalabra = (TextView) findViewById(R.id.tvPalabra);
        palabra = (palabras) getIntent().getSerializableExtra( name: "dato");
        btnComprobar = (Button) findViewById(R.id.btComprobar);
        edtCaracter = (EditText) findViewById(R.id.etCaracter);
        image = (ImageView) findViewById(R.id.tvImagen);

        String palabras = "";
        // se convierte la cadena de texto en un arrayChar
        char[] palabraMostrar = palabra.getCadena().toCharArray();
        // variable palabras concatena guiones por cada letra de la palabra
        for (int i = 0; i < palabraMostrar.length; i++) {
            palabras += " _";
        }
```

```
        // objeto palabras realiza proceso llenarList
        palabra.llenarList();
        // Se coloca en txt la concatenación de guiones para palabra
        txtPalabra.setText(palabras);
        // proceso onClick para botón comprobar
        btnComprobar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                procesoComprobar();
            } // llama función local procesoComprobar
        });
        // proceso onKey para aceptar tecla enter como submit
        edtCaracter.setOnKeyListener(new OnKeyListener() {
            @Override
            public boolean onKey(View view, int i, KeyEvent keyEvent) {
                if ((keyEvent.getAction() == KeyEvent.ACTION_DOWN) && (i == KeyEvent.KEYCODE_ENTER)) {
                    procesoComprobar(); // llama a función local procesoComprobar
                    return true;
                }
                return false;
            }
        });
    }
```

```
// función para comprobar aciertos y fallas
    public void procesoComprobar() {
        if (edtCaracter.getText().toString().length() < 2) { // valida que se ingrese solo 1 caracter
            String aux = palabra.validar(edtCaracter.getText().toString().toUpperCase()); // convierte caracter a mayuscula
            if (palabra.isExiste()) { // valida que exista caracter
                txtPalabra.setText(aux);
                edtCaracter.setText("");
                if (palabra.isPalabraValida()) { // valida que esté completa la palabra ingresada
                    Toast toast = Toast.makeText(getApplicationContext(), text: "Ganaste", Toast.LENGTH_LONG); // retorna aviso
                    toast.show();
                    Intent i = new Intent( packageContext: MainActivity2.this, MainActivity.class); // devuelve a MainActivity para reiniciar juego
                    startActivity(i);
                    return;
                }
            } else {
                Intentos++; // aumenta el numero de intentos
                cambiarImg(); // cambia a la siguiente imagen de error
                edtCaracter.setText("");
                if (Intentos > 6) {// valida si ahorcado aún no termina
                    // retorna aviso de perdia
                    Toast toast = Toast.makeText(getApplicationContext(), text: "Game Over!! \n Correcto: " + palabra.getCadena(), Toast.LENGTH_LONG);
                    toast.show();
                    Intent i = new Intent( packageContext: MainActivity2.this, MainActivity.class);
                    startActivity(i); // devuelve a MainActivity para reiniciar
                    return;
                }
            }
```

```
        } else {
            // mensaje de aviso si no se introdujo ningun caracter
            Toast toast = Toast.makeText(getApplicationContext(), text: "introduzca una letra con mas de 1 caracter!!!", Toast.LENGTH_SHORT);
            toast.show();
            return;
        }
    }
// función para cambiar imagen dependiendo del numero de intentos realizados
    public void cambiarImg() {
        if (Intentos == 0)
            image.setImageResource(R.drawable.img0);
        if (Intentos == 1)
            image.setImageResource(R.drawable.img1);
        if (Intentos == 2)
            image.setImageResource(R.drawable.img2);
        if (Intentos == 3)
            image.setImageResource(R.drawable.img3);
        if (Intentos == 4)
            image.setImageResource(R.drawable.img4);
        if (Intentos == 5)
            image.setImageResource(R.drawable.img5);
        if (Intentos == 6)
            image.setImageResource(R.drawable.img6);
        if (Intentos == 7)
            image.setImageResource(R.drawable.img7);

    }
```