Universidad Pública de Navarra
Curso 2021/2022

Gestión de Sistemas de Información

Guión de la Práctica 03

Marilin Vega León, Dpto. de Automática y Computación

1. Contenidos del documento

Este documento recoge la Práctica 03 de la asignatura Gestión de Sistemas de Información correspondiente al curso 2021/2022.

2. Desarrollo de la práctica

Cada grupo de alumnos debe completar los ejercicios de la Sección 5, tal y como se describe en la Sección 6. Si bien los ejercicios a realizar en esta práctica no son necesarios para completar la siguiente, es conveniente tenerla completada a tiempo.

Es necesario que los alumnos dispongan de estos conocimientos antes de comenzar la práctica:

- Gestión de árboles para el almacenamiento de información. Estructura (ámbito estático) y recorrido (ámbito dinámico);
- Modelado de información bajo las restricciones de un formato físico (o lógico) determinado (adquirida en la práctica anterior);
- Representación de información usando XML;
- Gestión de cadenas de texto en Java.

Es recomendable, por otro lado, que los alumnos tengan ciertos conocimientos en diseño y uso de parsers para gramáticas bien definidas.

Los alumnos deben adquirir los siguientes conocimientos en el transcurso de la práctica:

- Conocimientos teóricos:
  - Representación de un modelo de objetos usando un formato jerárquico. Gestión de referencias y cardinalidad.
  - Desarrollo de políticas de consistencia de información en formatos que no las soporten de manera nativa. Diseño de políticas específicas al modelo de negocio, referentes al almacenamiento de información.
- Conocimientos técnicos:
  - Análisis de librerías de *third parties*. Decisión acerca de su descarte o uso;
  - Almacenamiento automatizado y lectura de información en formato XML. Desarrollo de parsers y validadores que se adecúen[1] a políticas de negocio. Generación de documentación adecuada para los mismos;
  - Desarrollo de librerías de traducción para el *input/output* de un sistema de información;

1. Esta tilde la he mirado en la RAE- NdA.

3. Fechas de interés

Las fechas de interés para el desarrollo de la práctica son las que siguen:

- 21 y 28 de Octubre, 4 de Noviembre: Días de laboratorio asignados para completar la prácticas;
- 9 de Noviembre (14h): Fecha de entrega límite;
-

Nótese que la asistencia al laboratorio es únicamente obligatoria el primero de los días asignados para esta práctica.

4. Introducción a la práctica

La integración con formatos ofimáticos es una de las maneras que tiene un sistema de información para alcanzar la persistencia en el tiempo. De hecho, es una forma de almacenamiento que tiene varias ventajas, siendo la más evidente el hecho de que la información puede ser revisada y modificada con software especializado. Sin embargo, la información de un sistema medianamente complejo no tiene buen encaje en una malla multidimensional, como es el caso de una hoja de cálculo. Incluso en un sistema de información relativamente sencillo, como el de la Práctica 01, encontramos problemas para almacenar datos multivalorados y referencias a objetos complejos.

En esta práctica tratamos la representación de objetos en formatos textuales (es decir, basados en caracteres y legibles). En concreto, vamos a trabajar con XML, un lenguaje muy flexible que permitirá que almacenemos diferentes tipos de datos a partir de su representación textual.

Es una buena praxis el generar un documento específico con la definición de tipos para cada dominio de negocio (un *Data Type Document*, o DTD), pero en esta práctica omitiremos este paso. El motivo es, simple y llanamente, la falta de tiempo para cubrir todos los posibles objetivos de aprendizaje. De cualquier manera, esta práctica multiplica su utilidad cuando los datos (y las operaciones sobre ellos) se publican para su consumo remoto. En esta práctica usaremos XML como un convenio local de almacenamiento y representación, de manera que no se hace muy necesario el generar un DTD.

5. Lista de ejercicios

Ejercicio 1. Descargue de MiAulario el *interface* `GSILabs.Serializable.XMLRepresentable`. Haga que todas las clases del paquete `GSILabs.BModel` lo implementen.

El ejercicio anterior parece sencillo en un primer término, pero tiene un matiz interesante, como son las relaciones entre objetos. Si la relación es inclusiva (es decir, *es parte de*), parece claro que el componente debe estar integrado en el árbol XML. Sin embargo, en ocasiones tenemos relaciones que no responden a esta semántica, como puede ser la existente entre las *review*, los locales a los que se refieren y los propios autores del *review*. ¿Son dichas relaciones de igual naturaleza? ¿Pertenecen al local, o al autor? ¿A ambos, o a ninguno? ¿Son algo inherente al sistema, y por tanto no deben aparecer en la representación de los objetos (Review, Local y Cliente) en XML?

Este problema nace del hecho de que los objetos se relacionan sin restricciones (formando un grafo), mientras que la representación XML establece una jerarquía de inclusiones. Debe decidir cuál de las alternativas va a implementar (obviar otros objetos, incluir aquellos que semánticamente estén *contenidos*, etc.), y justificarlo en un informe de no más de 400 palabras. Este informe se llamará

P03Ej01GrXX.pdf, donde XX se refiere al número de grupo, y se incluirá dentro de la carpeta comprimida a entregar.

Ejercicio 2. Haciendo uso de los métodos implementados en el ejercicio anterior, haga que la clase `BusinessSystem` también implemente el interfaz `XMLRepresentable`.

Como se ha visto en los ejercicio anteriores, la serialización de una instancia en XML es relativamente sencilla, siendo lo más problemático la relación con otros objetos. Hay que tener en cuenta que esto depende mucho de la lógica con la que se haya implementado en sistema. Si ésta se hace en términos de objetos, las relaciones entre objetos se hacen difíciles de representar, ya que estas no tienen una entidad propia en un árbol inclusivo como el que se construye con XML. Sin embargo, si hubiéramos optado por una representación basada en tablas, en las cuales cada relación entre objetos se materializa como una instancia de una clase (o fila de una tabla), esto no habría supuesto problema alguno. Es relevante reflexionar acerca de la idoneidad de un diseño ajustado a la realidad, si esto implica dificultades en la representación física/persistente del sistema. El objetivo no es tanto priorizar la interpretabilidad o la representación física, sino comprender las alternativas y encontrar un compromiso aceptable entre ellas.

A pesar de la problemática de las relaciones, la complejidad técnica de los ejercicios anteriores es bastante reducida. Sin embargo, a la hora de crear objetos a partir de su representación XML todo se vuelve, a nivel de programación, más complicado. Por ejemplo, tenemos que considerar el hecho de que un árbol XML no está ordenado, por lo que la información (los miembros) de cada objeto puede aparecer en cualquier orden. Además, hay que encontrar una manera elegante de normalizar la lectura/escritura de objetos en ficheros o representaciones XML.

Ejercicio 3. Elija una de estas tres alternativas para leer objetos desde representaciones XML.

a) Cree dentro del paquete `GSILabs.persistence` una clase que implemente métodos estáticos de creación para cada una de las clases instanciables del paquete `GSILabs.BModel`. Por cada clase instanciable debe haber un método que cree un objeto desde un `String`, y otro que lo haga desde un fichero. Por ejemplo, para la clase `Bar` deberá haber un método estático con cabecera

```
Bar parseBar(String str)
```

y otro con cabecera

```
Bar parseBar(File f).
```

b) Añada a cada una de las clases instanciables del paquete `GSILabs.BModel` un constructor que admita como único parámetro una representación XML del objeto en un `String`.

c) Añada a cada una de las clases instanciables del paquete `GSILabs.BModel` un método estático que cree una instancia del objeto en que se encuentre a partir de (a) un `String` y (b) un `File`. El nombrado de los mismos debe ser equivalente al mencionado en la alternativa a).

Uno de los problemas que tiene el uso de XML es la potencial malformación de la representación de cada objeto. Hasta el momento del parseado, no existe manera de saber si el objeto es *correcto*. Además, dado que XML no ofrece maneras nativas de comprobar el contenido de un objeto[2], resulta imposible saber si este contiene toda la información que necesita, etc. Por ello, debe implementarse una excepción que sea lanzada en caso de encontrar un XML inadecuado.

2. De hecho sí existen validadores que analizan si un determinado objeto se ajusta a un DTD.

Ejercicio 4. Implemente una clase `GSILabs.persistence.XMLParsingException` que extienda `Exception` y haga que todos los métodos que analicen un XML la lancen. En caso de usar dichos métodos dentro del código del paquete, encárguese de manejar la excepción con los `try-catch` correspondientes.

Existen muchos motivos por los cuales es interesante serializar objetos en XML, incluyendo posibilidad de enviarlos por canales de comunicación o interactuar con plataformas programadas en otros lenguajes. Sin embargo, la más evidente es ser capaz de generar copias persistentes de la información en forma legible y editable. El siguiente ejercicio permite que toda la información del sistema se almacene en un fichero.

Ejercicio 5. Dentro de la clase `GSILabs.BSystem.BusinessSytem` implemente dos métodos para la lectura de ficheros.

Estos métodos tienen que responder a las cabeceras

```
public static BusinessSystem parseXMLFile(File f) throws XMLParsingException
```

y

```
public boolean loadXMLFile(File f),
```

respectivamente. Cree la clase P03Tester en Test Packages / Pruebas , para hacer pruebas de las funcionalidades

6. Metodología de entrega

Cada grupo debe realizar una única entrega a través de la sección Tareas de MiAulario. Cada grupo de alumnos debe entregar en una carpeta comprimida el proyecto, incluyendo:

a) El proyecto, bien sea de Eclipse o NetBeans;

b) Un breve informe en formato *.pdf con un máximo de 400 palabras en el que justifique la elección de la alternativa en el Ejercicio 1;

c) Un breve informe en formato *.pdf con un máximo de 400 palabras en el que justifique la elección de la alternativa en el Ejercicio 3.

Dicha carpeta se llamará `GrXX.zip` o `GrXX.7z`, donde `XX` debe reemplazarse por el número del grupo. El proyecto debe ser totalmente autocontenido, no haciendo uso de rutas absolutas o dependencias de archivos no incluidos. Cualquier violación de estas normas tendrá una penalización de $-20\,\%$ en la nota del grupo.

La nota se asignará, sobre un máximo de 14 puntos, en función de los siguientes criterios:

e1.1 Se ha completado con éxito el Ejercicio 1 (2 puntos);

e1.2 Se ha justificado la decisión tomada en el Ejercicio 1 en lo referente a la inclusión de objetos relacionados en la representación XML de una instancia dada (2 puntos);

e1.3 Se ha completado con éxito el Ejercicio 2 (3 puntos);

e1.4 Se ha completado con éxito el Ejercicio 3 (2 puntos);

e1.5 Se ha completado con éxito el Ejercicio 4 (2 puntos);

e1.6 Se ha completado con éxito el Ejercicio 5 (3 puntos);

Cada uno de los criterios podrá evaluarse como *fallido* ($0\,\%$), *incompleto* ($25\,\%$, $50\,\%$, $75\,\%$) o *completo* ($100\,\%$). Téngase en cuenta que, dentro de la completitud de los mismos, se consideran

cosas como la adecuada documentación, la limpieza del código, etc. Además, como en prácticas anteriores, el informe deberá estar escrito siguiendo ciertas convenciones y criterios de calidad, y recibirá automáticamente una puntuación de 0 si contiene faltas de ortografía.

5

Control de versiones

Este documento no ha sufrido modificaciones desde su publicación (XX/X/2021).