Objetos en la Web

Diego de la Riva[1], Patricia Miguel[2]

diego.delariva@nexo.unnoba.edu.ar, contacto@patriciaemiguel.com

Escuela de Tecnología – UNNOBA
Calle Jorge Newbery y Sarmiento – (6000) Junín, Bs As., Argentina

Abstract. El presente trabajo consiste en un sitio web destinado a la enseñanza del paradigma de Programación Orientada a Objetos (POO). El sitio incluye conceptos teóricos, acompañados de ejercicios de diseño y ejercicios de programación, estos últimos con la particularidad de permitir la autoevaluación por parte del propio usuario. El sitio se utiliza actualmente como complemento teórico-práctico para la enseñanza del paradigma en las carreras de Informática de la Universidad.

Keywords: POO, programación orientada a objetos, programación, enseñanza autodidacta.

Introducción

El trabajo objeto del presente informe se desarrolló como proyecto de fin de la carrera "Programador Universitario" de la Universidad Nacional del Noroeste de la Provincia de Buenos Aires (UNNOBA). El producto apunta a complementar la enseñanza del paradigma de Programación Orientada a Objetos con la utilización de una aplicación web simple y didáctica. La aplicación web cuenta con una sección de "lecciones" que explican conceptos teóricos, presentados de manera ordenada. A la vez, plantea dos secciones de ejercicios: de diseño y de programación, que permiten al usuario corregir de manera automática sus resoluciones. A los efectos de la práctica de programación se seleccionó el lenguaje Java principalmente por la implementación que dicho lenguaje posee de los conceptos de programación orientada a objetos. Este complemento teórico-práctico junto con la elección del lenguaje, facilita al usuario la comprensión del paradigma.

---

[1] Profesor Adjunto, Escuela de Tecnología, UNNOBA
[2] Ayudante Diplomado, Escuela de Tecnología, UNNOBA

Fundamentación

La programación orientada a objetos permite introducir al usuario principiante en la programación de sistemas a través de un proceso intuitivo, que posibilita la incorporación de conocimientos teóricos y cuestiones específicas de este paradigma durante el desarrollo mismo de análisis del problema a resolver. Esto significa que, alcanzado cierto nivel básico de conocimientos, el programador será capaz de comprender y aprender rápidamente otros lenguajes dentro de este paradigma, contando así con un amplio abanico de posibilidades, tanto educativas como laborales. Si bien dentro de las asignaturas de objetos se brindan explicaciones y consultas de los trabajos prácticos, no se dispone de una herramienta de software que pueda asistir al alumno en la comprensión del tema. El propósito de este producto es proporcionar al alumno un tutor virtual en el aprendizaje del paradigma, con el marco conceptual establecido en las materias.

Objetos en la Web

Objetos en la web es una aplicación web que complementa el aprendizaje del paradigma de Programación Orientada a Objetos.

Objetivo

Facilitar y complementar el aprendizaje de conceptos básicos sobre programación orientada a objetos a través de una aplicación web que incluya lecciones teóricas, ejercicios de diseño y ejercicios prácticos.

Desarrollo

Corresponde al desarrollo de una aplicación web con lecciones teóricas, ejercicios de diseño y de programación del paradigma de programación orientada a objetos.
Entre los conceptos teóricos se incluyen aspectos de diseño, de modelado (introduciendo conceptos iniciales sobre diagramas de clases con UML) y de programación en lenguaje Java.
La sección práctica cuenta con ejercicios de modelado UML y de programación. En los primeros, se enuncia un problema a modelar y se ofrecen alternativas, entre las cuales el usuario puede elegir la que cree más viable. Para los ejercicios de programación se cuenta con un sistema de autocorrección mediante compilación, de

forma que el usuario pueda evaluar sus conocimientos y conocer los resultados inmediatamente.
Debido a que los ejercicios prácticos deben desarrollarse utilizando algún lenguaje de programación orientado a objetos, se escogió a esos fines Java, hoy día uno de los lenguajes de mayor utilización en ámbitos académicos y profesionales. Sin embargo, el trabajo apunta a proveer al usuario de conocimientos básicos extensibles a cualquier otro lenguaje del paradigma de objetos.
Dentro de la estructura general de la aplicación, se destacan las tres secciones principales: "Lecciones", "Ejercicios de diseño" y "Ejercicios de programación".
La sección "Lecciones" contiene los conceptos teóricos que se desea transmitir y se encuentra dividida en títulos que cubren temas centrales de la Programación Orientada a Objetos. Los títulos se muestran en forma de lista, manteniendo oculto su contenido. El usuario, al hacer click sobre uno de estos títulos, despliega el contenido oculto (que en algunos casos incluye ejemplos e imágenes). Los conceptos teóricos se focalizan en las nociones centrales de la Programación Orientada a Objetos y el correcto modelado, pero también se proveen elementos sintácticos y semánticos de Java, ya que es el lenguaje escogido para desarrollar los ejercicios de programación.
La sección de "lecciones" finaliza con un último título que provee un ejemplo de diagrama UML y su interpretación (posicionando, esta vez, al usuario en el rol del programador que recibe un diagrama a implementar), acompañado de un proyecto de código de ejemplo basado en dicho diagrama.
Al plantear los ejercicios de diseño y programación, se tuvieron en cuenta los conceptos vertidos en las lecciones teóricas, de modo que el usuario pueda encontrar toda la información para resolverlos dentro del propio sitio, sin necesidad de recurrir a fuentes externas (aunque se sugiere la lectura de la documentación de cátedra para una mayor profundización y especificidad en los conceptos).
La sección "Ejercicios de diseño" contiene cinco ejercicios de modelado, planteándose por cada uno una situación a resolver, abstraída y enunciada en un lenguaje coloquial pero simulando un caso real, y se ofrecen tres alternativas de diseño plasmadas en diagramas de clases UML. Se aclara al usuario que el modelado de una solución no es único pero que, de las alternativas ofrecidas, dos contienen errores que en la alternativa "correcta" se encuentran subsanados. Al final de cada ejercicio se ofrece un menú que permite seleccionar una de las tres alternativas, permitiendo así al usuario indicar la que cree más viable. A continuación de este menú se encuentra un botón con la leyenda "verificar", que compara la opción elegida por el usuario con la opción planteada como correcta, e informa el resultado: "Respuesta correcta" o "Respuesta incorrecta", según sea el caso. Además de informar al usuario del resultado, hace visibles ciertos componentes de texto que se encuentran ocultos, debajo de cada opción. Para las alternativas de diseño incorrectas, este texto detalla los errores introducidos y sugiere los títulos de la sección de Lecciones que refieren a los conceptos relacionados.
Para la sección de "Ejercicios de programación" se desarrolló un servlet Java que efectúa la compilación del código introducido por el usuario como solución a los

ejercicios. Para esta compilación se utilizó el compilador interno de la máquina virtual de java (JVM), de manera de simular un entorno de compilación real, apuntando a una revisión sintáctica muy similar a la provista por cualquier compilador. Se intentó así mejorar el sistema utilizado en algunos sitios educativos, en que se guía al usuario para que estructure su código de una manera particular, para luego compararlo con un texto predefinido, quitando dinamismo al proceso de aprendizaje.

Se presentan veinte ejercicios de programación, divididos en tres áreas. Cada área de ejercicios contiene enunciados de temática similar, generando una división conceptual de los mismos. Las áreas son: "tipos de datos", "métodos" y "estructuras de control" y cumplen una importante función en el procesamiento realizado por el servlet ya que, dependiendo del área de que se trate, el servlet identificará el tipo de código que recibirá, para determinar el tratamiento que debe darle.

Por cada área se presenta al usuario con un listado de ejercicios numerados y enunciados en pocas líneas. Al final del listado se encuentra un formulario que permite al usuario ingresar código java para luego presionar un botón con la leyenda "corregir", obteniendo a continuación el resultado de la compilación de su código. Además, se ofrece un enlace con la leyenda "ver respuestas" que muestra, debajo de cada ejercicio, la respuesta correcta. El resultado de la corrección de los ejercicios resueltos por el usuario se muestra sin navegar a una nueva página, para lo cual se utilizó Ajax.

El servlet recibe como parámetro el código escrito por el usuario, lo procesa y genera la respuesta. Para la tarea de procesamiento del código fuente se debe determinar la estructura que ha de tener la clase a los efectos de generar una sintaxis bien formada. De esta manera también se evita la necesidad de escribir todo el código necesario para realizar una compilación exitosa, permitiendo al usuario focalizarse en resolver el problema que se le plantea. A modo de ejemplo: si en un ejercicio se solicita la escritura de un atributo para guardar determinado dato inicializándolo con un valor, solo se espera que el usuario escriba una sentencia y no toda la estructura de la clase necesaria para una compilación exitosa. En este caso se completará el código del usuario con código prestablecido para realizar la compilación:

"class Test {" + codigoDeUsuario + " }"

Luego de completado el código necesario se genera una clase en memoria objeto de compilación y una colección donde se almacenarán los posibles errores de compilación.

Con toda la información y objetos necesarios, se crea y ejecuta una tarea de compilación que retornará un valor booleano con el resultado de la misma. Si la compilación falla significa que hubieron errores en la compilación, entonces se itera por la colección para revisar y retornar al usuario los errores encontrados. Si la compilación es exitosa se retornará "Compilación exitosa".

El navegador imprime directamente el valor retornado por el servlet en la pantalla del usuario mediante Ajax para evitar la recarga o navegación hacia una nueva página.

Beneficiarios

Los principales beneficiarios con el uso de la aplicación son los alumnos de las materias referidas a programación orientada a objetos, pero también puede ser utilizada por personas que deseen iniciarse en el aprendizaje del paradigma de forma autodidacta.

Arquitectura

La aplicación web está desarrollada utilizando HTML, CSS, Javascript y Ajax ya que, por los objetivos y metodología propuestos, se consideraron suficientes estas herramientas. El diseño gráfico del sitio web responde a una plantilla de uso libre.
El proyecto fue creado utilizando Eclipse Java EE IDE, versión Kepler Service Release 2.
El código de los ejercicios de programación es recibido y procesado mediante el servlet Java. Un servlet es una abstracción de un servicio web provisto por el lenguaje Java, que funciona bajo una arquitectura cliente-servidor y corre del lado del servidor únicamente. Se trata de una clase java con capacidad para procesar peticiones y devolver una respuesta. El servlet debe ejecutarse dentro del contexto de un "contenedor web". A esos fines se utilizó Apache Tomcat versión 7.
El servlet en este trabajo tiene la función de compilar código de manera dinámica, abstrayendo la noción de archivos de código fuente, ya que el código es pasado al servlet como parámetro.

Resultados obtenidos

El producto ha logrado los siguientes resultados:

- Colabora con la tarea educativa realizada en las aulas de la universidad, en particular las realizadas en las materias de programación orientada a objetos de las carreras de informática.
- Brinda una herramienta dinámica de autoaprendizaje tanto para alumnos como para cualquier persona interesada en aprender el paradigma.
- Facilita al educador la tarea de transmitir conocimientos y al estudiante la de internalizarlos.
- Introduce al usuario en un paradigma de programación vigente y robusto, de amplia utilización.
- Brinda al usuario conocimientos en el lenguaje Java, de alta adopción tanto en ámbitos académicos como laborales.

Trabajos futuros

Luego de la implementación de la primera versión del sitio surgieron nuevas funcionalidades o mejoras al mismo: implementar concurrencia, incorporar verificación semántica del código compilado, implementar seguridad y proveer dinamismo en la presentación de los contenidos.
Concurrencia: El resultado de la compilación se almacena en un archivo que se crea localmente en el disco del servidor. En el caso que varios usuarios estén accediendo a la vez, intentarán crear una y otra vez el archivo, sobrescribiendo el código anterior pudiendo ocasionar inconsistencias.
Verificación semántica: El código del usuario es sólo analizado en cuanto a sintaxis, con lo cual existe la posibilidad de errores semánticos y lógicos, en los que un código es sintácticamente correcto pero su funcionalidad no es la esperada. Para aquellos ejercicios que consisten en declaraciones y asignaciones de variables debe comprobarse el valor y tipo de estas variables, lo cual podría lograrse utilizando la API "Reflection" de Java. Para los ejercicios que requieran definición e implementación de métodos debería crearse una interfaz con un método que permita ejecutar el código de usuario.
Seguridad: Al ejecutarse en el servidor código escrito por un usuario, se abre la aplicación a potenciales riesgos de seguridad, ante la posibilidad de usuarios malintencionados. Para esto, deberían implementarse políticas de seguridad que restrinjan o controlen la ejecución de código.
Contenidos dinámicos: El proyecto fue pensado para ser gestionado por un único administrador, que podría ser de acceso público para cualquier usuario interesado en aprender a programar. Pero dada la potencial utilidad que podría encontrársele en ámbitos académicos, podría convertirse en una plataforma que permita al docente crear sus propios contenidos, completando las secciones de "lecciones", "ejercicios de diseño" y "ejercicios de programación" de manera personalizada de acuerdo a sus necesidades. Debería proveerse también una sección de acceso privado para el docente, desde donde éste pueda administrar los contenidos a mostrarse en la aplicación.

Conclusiones

El producto desarrollado corresponde a un sitio web que permite complementar la enseñanza del paradigma de Programación Orientada a Objetos. Su estructuración en lecciones teóricas, ejercicios de diseño y ejercicios prácticos brindan al usuario la posibilidad de mejorar la compresión del paradigma en sus aspectos principales. La elección del lenguaje Java responde no solo a la masiva utilización del mismo sino a sus características de completitud, simplicidad, multiplataforma y robustez. Estas cualidades del sitio han hecho que sea elegido como herramienta complementaria para

el dictado de la materia “Introducción a la Programación Orientada a Objetos” de la Escuela de Tecnología de la Universidad Nacional del Noroeste de la Provincia de Buenos Aires.

Referencias

1. Lott, S.; "Building Skills in Object-Oriented Design"; 2009.
2. Fowler, M; "UML Distilled"; segunda edición; Addison-Wesley; Massachusetts, 1999; ISBN 020165783X.
3. Rumbaugh, J.; Jacobson, I.; Booch, G.; "The Unified Modeling Language Reference Manual"; Addison-Wesley; Massachusetts, 1999; ISBN 020130998X.
4. IBM Rational Software; "UML Notation Guide"; versión 1.1; 1997.
5. Rumbaugh, J.; Jacobson, I.; Booch, G.; "The Unified Modeling Language User Guide"; Addison-Wesley; Massachusetts, 1998; ISBN 0201571684.
6. http://download.oracle.com/javase/7/docs/api/
7. http://www.cs.bsu.edu/homepages/pvg/misc/uml/
8. http://www.objectmentor.com/resources/articles/umlClassDiagrams.pdf
9. http://javascript-reference.info/ajax-getting-started.htm
10. http://www.javaranch.com/journal/200711/Journal200711.jsp#a4
11. http://www.javaranch.com/journal/200607/Journal200607.jsp#a1
12. http://www.accordess.com/wpblog/an-overview-of-java-compilation-api-jsr-199/
13. http://jexp.ru/index.php/Java_Tutorial/Development/Java_Compiler
14. http://www.roseindia.net/java/example/java/applet/WriteFile.shtml
15. http://www.javabeat.net/articles/73-the-java-60-compiler-api-4.html
16. http://tutorials.jenkov.com/java-reflection/dynamic-class-loading-reloading.html
17. http://www.javaworld.com/article/2077260/learn-java/the-basics-of-java-class-loaders.html