Universidad de San Carlos de Guatemala
Centro Universitario de Occidente División de Ciencias de la Ingeniería
Compiladores 1
Ing. Daniel Gonzalez
Sección: A

Manual Técnico

Carlos Raúl Alberto López Peláez 202031871

Quetzaltenango,22 de Abril del 2024

Servidor Manejador de Contenido

Clase ServidorP

Clase ServidorP:

- static Selector selector: Se declara un selector estático que se utilizará para monitorear múltiples canales de entrada/salida.
- interpretar(String path): Método privado que interpreta la entrada recibida del cliente utilizando un parser generado por JFlex y JCup.
- iniciarServidor(): Método público que inicializa y ejecuta el servidor.
- sendErrorMessage(String errorMessage): Método estático que envía un mensaje de error a todos los clientes conectados.

Método iniciarServidor():

1. Se abre un selector y un canal de servidor (ServerSocketChannel) en el puerto 8888 y se registra con el selector para aceptar conexiones entrantes.
2. Se entra en un bucle infinito que escucha las operaciones de entrada/salida.
3. Cuando una conexión nueva es aceptada (OP_ACCEPT), se configura un canal de socket no bloqueante y se registra con el selector para leer datos entrantes.
4. Cuando hay datos disponibles para leer (OP_READ), se leen en un ByteBuffer.
5. Se interpreta el mensaje recibido utilizando el método interpretar(message).
6. Si se produce un error al leer o interpretar el mensaje, se maneja la excepción correspondiente.
7. Se procesa el mensaje según sea necesario.
8. El servidor continúa esperando nuevas operaciones de entrada/salida.

Método sendErrorMessage(String errorMessage):

- Este método envía un mensaje de error a todos los clientes conectados. Recorre todos los SelectionKey registrados y envía el mensaje de error a cada cliente a través de su canal de socket.

Servidor del lado del cliente

Clase Cliente:

- ByteBuffer buffer: Se declara un buffer para leer y escribir datos.
- startClient(String mensaje): Método público que inicia el cliente y envía un mensaje al servidor.
- respuestaServidor(SocketChannel socketChannel): Método privado que lee la respuesta del servidor y actualiza la interfaz de usuario.

Método startClient(String mensaje):

1. Se crea un canal de socket y se conecta al servidor en el puerto 8888.
2. El socket se configura como bloqueante.
3. Se envía el mensaje al servidor. El mensaje se convierte en bytes y se envía al servidor utilizando el buffer.
4. Se llama al método respuestaServidor() para esperar y procesar la respuesta del servidor.
5. Se cierra la conexión después de recibir la respuesta.

Método respuestaServidor(SocketChannel socketChannel):

1. Se limpia el buffer para prepararlo para recibir datos.
2. Se lee la respuesta del servidor en el buffer.
3. Si se reciben bytes, se convierten en una cadena y se actualiza la interfaz de usuario con el mensaje recibido.
4. Si no se recibe ninguna respuesta del servidor, se muestra un mensaje de error.

Jcup

Terminales:

- Define los tokens terminales que serán reconocidos por el analizador.

No Terminales:

- Define los símbolos no terminales de la gramática, que representan las reglas de producción de la gramática.

Reglas de Producción:

- Especifica las reglas de producción de la gramática. Por ejemplo, la regla ini ::= instrucciones; indica que un programa puede comenzar con un conjunto de instrucciones.
- Cada regla de producción consta de un no terminal seguido de ::= y una secuencia de terminales y/o no terminales que representan cómo se puede construir ese no terminal.

Jflex

Secciones principales:

1. Definiciones:

- %line %char %cup %unicode %caseless %ignorecase: Define opciones para el analizador:
- %line: Mantiene registro del número de línea actual.
- %char: Mantiene registro del número de columna actual.
- %cup: Genera código compatible con CUP (Constructor Universal de Parsers).
- %unicode: Habilita el soporte para caracteres Unicode.
- %caseless %ignorecase: Ignora la diferencia entre mayúsculas y minúsculas.
- %init{ ... }: Código que se ejecuta al iniciar el analizador, inicializando las variables yyline (línea actual) y yychar (columna actual).
- %{ ... }: Código auxiliar que se puede utilizar dentro de las reglas léxicas.
- %state: Define estados para el analizador. En este caso, solo se define el estado STRINGBLANCOS para manejar cadenas de texto.
- IDS, LETRAS: Define expresiones regulares para identificar tokens específicos.

2. Reglas Léxicas:

- <YYINITIAL>: Indica que las siguientes reglas se aplican en el estado inicial del analizador.
- "acciones" { ... }: Define la regla para el token acciones. Si encuentra la palabra "acciones", devuelve un objeto Symbol con el tipo sym.ACCIONES, la línea actual, la columna actual y el texto del token.
- Las demás reglas siguen el mismo formato, definiendo patrones para identificar diferentes tipos de tokens, como palabras reservadas, identificadores, operadores, etc.
- <STRING>: Define reglas para el estado STRING, que se encarga de leer cadenas de texto entre comillas.

3. Funcionamiento general:

- El analizador léxico lee el código fuente caracter por caracter.
- Compara cada caracter con las reglas léxicas definidas.
- Si encuentra una coincidencia, devuelve un objeto Symbol que representa el token encontrado.
- Si no encuentra una coincidencia, se produce un error léxico.

Clase Ejecucion

Método valor:

Este método es el núcleo de la funcionalidad del archivo. Recibe como parámetros un valor (que es el tipo acción que se va a realizar), un objeto de tipo Atributos y un objeto de tipo Parámetros.

- El método comienza con la impresión del valor recibido.
- Luego, se comprueba si el archivo XML de la página existe y si la acción es "MENU". Si se cumplen estas condiciones, se realizan diversas operaciones para obtener información sobre las páginas y los sitios.
- Después, hay una serie de condicionales que parecen manejar diferentes acciones, como modificar componentes, páginas, borrar componentes, sitios web o páginas, o agregar nuevos componentes o páginas.
- Finalmente, se retorna un valor booleano que indica si la operación fue exitosa o no.

Método generadorXml:

Este método genera elementos XML a partir de los parámetros y atributos recibidos. Crea elementos <accion> con sus correspondientes <parametros> y <atributos>, agregando cada uno de ellos a una lista y luego imprimiendo la información de las acciones agregadas.

Métodos agregarParametro y agregarAtributo:

Estos métodos son utilizados por el método generadorXml para agregar parámetros y atributos a los elementos XML.

Clase Modificar Xml

En general esta clase se encarga de modifcar, crear , buscar , elminar etiquetas en xml.

- **accionInicial():** Esta función devuelve un objeto Document que representa el documento XML que se va a modificar. Lee un archivo XML desde una ubicación específica y lo parsea para obtener un objeto Document.
- **modificarAccion():** Esta función recibe como parámetros el documento doc, un id de acción, el nombre de un atributo y un nuevoValor. Busca una acción dentro del documento XML con el ID proporcionado y modifica el valor del atributo especificado con el nuevo valor.
- **modificarEtiquetas():** Similar a la función anterior, esta función busca una acción por su ID y agrega una nueva etiqueta al elemento de etiquetas de dicha acción. También puede eliminar todas las etiquetas existentes si un contador (contar) es igual a cero.
- **modificarTitulo():** Busca una acción por su ID y modifica el valor del parámetro "TITULO" con el nuevo título proporcionado.
- **eliminarAccion():** Elimina una acción del documento XML según su nombre.
- **eliminarAccionPorId():** Elimina una acción del documento XML según su ID.
- **eliminarAccionPorPagina():** Elimina una acción del documento XML según el nombre de la página.
- **anadirAccion():** Añade una o más acciones al documento XML.
- **anadirPagina():** Crea un nuevo elemento de página en el documento XML con los parámetros y atributos proporcionados.
- **guardarCambios():** Guarda los cambios realizados en el documento XML de vuelta al archivo en el disco.
- **obtenerIds(), obtenerIdsSitios(), obtenerEtiquetas(), obtenerIdsNuevasPaginasConPadre():** Estas funciones son auxiliares para extraer información específica del documento XML, como IDs, etiquetas, etc.

Clase ejecución Html

- **Método valor**: Este método parece ser el núcleo de la funcionalidad del archivo. Recibe como parámetros un valor (posiblemente una acción a realizar), un objeto de tipo Atributos y un objeto de tipo Parametros. Este método procesa diferentes acciones dependiendo del valor recibido, como la creación de un nuevo sitio web o página, la adición de componentes, entre otros. También realiza operaciones de manipulación de archivos XML.
- **Método interpretar**: Este método parece ser responsable de interpretar un archivo XML recibido como cadena de texto. Utiliza un parser para analizar la estructura del archivo XML.
- **Métodos de utilidad removeXMLDeclaration y convertXMLToString**: Estos métodos son utilizados para manipular cadenas de texto que representan archivos XML. removeXMLDeclaration elimina la declaración XML de una cadena de texto XML, mientras que convertXMLToString convierte un archivo XML en una cadena de texto.
- **Método ejecutarHtml**: Este método parece ser responsable de ejecutar la lógica principal del programa. Recibe un mensaje, una página y un valor booleano. Dependiendo del valor booleano y la existencia de ciertos archivos, ejecuta diferentes acciones, como la generación de archivos HTML o la interpretación de archivos XML.

Generar Html

- **Método Generar**: Este método recibe como parámetros el tipo de acción a realizar (como la creación de un nuevo sitio web, una nueva página, o la adición de un componente), los atributos asociados y los parámetros necesarios. Luego, genera el contenido HTML correspondiente a la acción especificada. Dependiendo del tipo de acción, el método construye diferentes elementos HTML, como la estructura básica de una página web, encabezados, párrafos, imágenes, videos o menús de navegación.
- **Manipulación de archivos HTML**: El método utiliza objetos BufferedWriter y FileWriter para escribir el contenido HTML generado en un archivo en el sistema de archivos.