|  |  |  |  |
| --- | --- | --- | --- |
| **CARRERA** | **SEMESTRE** | **CÓDIGO MATERIA** | **NOMBRE DE LA MATERIA** |
| Ingeniería Informática | 8º | 805 | Programación Web |

|  |  |  |
| --- | --- | --- |
| **PRÁCTICA N.º:** | **TEMA:** | Framework (Struts) de desarrollo java. |
| 4  5  6 | **NOMBRES:** | Dayana Diaz |
| **FECHA ENTREGA:** | 24 de mayo del 2018 |

|  |  |
| --- | --- |
| **1.** | **INTRODUCCIÓN** |
| La arquitectura Modelo Vista Controlador constituye como hemos visto una excelente solución a la hora de implementar una aplicación Web en Java, aunque desde luego no está exenta de inconvenientes.  Si observamos detenidamente el ejemplo de código presentado en el Capítulo anterior, comprobaremos la existencia de dos puntos débiles que para grandes aplicaciones podrían traducirse en un desarrollo excesivamente complejo y difícilmente mantenible. Estos puntos débiles son:   * **Extensión del Controlador**. En aplicaciones en donde haya que gestionar varios tipos de peticiones, esto se traducirá en un elevado número de líneas de código en el servlet controlador, haciéndolo demasiado complejo y propenso a errores. * **Código Java en páginas JSP**. La presencia de código Java en una página JSP, aunque proporciona una gran flexibilidad a la hora de generar las vistas, hace difícilmente mantenibles las páginas y obstaculiza la labor del diseñador de las mismas.   Como solución a estos problemas se puede optar por dividir el servlet controlador en varios servlets más pequeños, encargándose cada uno de ellos de gestionar un tipo de petición, así como crear librerías de acciones JSP personalizadas que encapsulen la mayor parte del código Java utilizado en la generación de las vistas.  Otra opción consiste en emplear algún tipo de utilidad que facilite las tareas descritas anteriormente; y es aquí donde entra en juego **Struts**. | |

|  |  |
| --- | --- |
| **2.** | **LOGRO DE APRENDIZAJE** |
| * Conocer las ventajas que nos ofrece el framework Struts al aplicar el modelo vista controlador. * Aplicar Struts para un desarrollo de fácil y eficaz de aplicaciones web con tecnología JAVAEE. | |

|  |  |
| --- | --- |
| **3.** | **MARCO TEÓRICO** |
| **FUNDAMENTOS DE STRUTS**  Struts es un framework o marco de trabajo desarrollado por el grupo Apache, que proporciona un conjunto de utilidades cuyo objetivo es facilitar y optimizar los desarrollos de aplicaciones Web con tecnología JavaEE, siguiendo el patrón MVC.  El empleo de Struts en los desarrollos ofrece numerosos beneficios al programador, entre los que podríamos destacar:   * **Control declarativo de peticiones**. Con Struts el programador no tiene que preocuparse por controlar desde código las distintas acciones a realizar en función del tipo de petición que llega al Controlador. Este proceso es implementado por defecto por uno de los componentes que proporciona Struts, encargándose únicamente el programador de definir en un archivo de configuración XML los mapeos entre tipos de petición y acciones a ejecutar. * **Utilización de direcciones virtuales**. A fin de evitar la inclusión directa dentro del código de la aplicación de las URL de los recursos para hacer referencia a los mismos, Struts proporciona unos tipos de objetos que permiten referirse a estos recursos mediante direcciones virtuales asociadas a los mismos. La asociación entre una dirección virtual y su correspondiente URL o dirección real se realiza en un archivo de configuración, de este modo cualquier cambio en la localización de un recurso no afectará al código de la aplicación. * **Manipulación de datos con JavaBean**. Como hemos podido comprobar, los JavaBeans constituyen una pieza importante dentro de la arquitectura MVC al facilitar el transporte de datos entre las capas de la aplicación. Struts proporciona un sólido soporte para la manipulación de datos mediante JavaBeans, encargándose automáticamente el framework de la instanciación de los mismos, su rellenado con los datos procedentes del cliente y la recuperación y almacenamiento de los objetos en las variables de ámbito; todo esto sin necesidad de que el programador tenga que escribir una sola línea de código. * **Juego de librerías de acciones JSP**. A fin de poder reducir al mínimo la cantidad de instrucciones de código Java dentro de una página JSP, Struts proporciona un amplio conjunto de librerías de acciones predefinidas con las que se pueden realizar la mayor parte de las operaciones que tienen lugar dentro de una página JSP, tales como el acceso a las variables de ámbito y parámetros de petición, la iteración de colecciones, manipulación de JavaBeans, etc.   La aportación de Struts al desarrollo de aplicaciones MVC se dirige básicamente a la construcción del Controlador y la Vista de una aplicación, dejando total libertad al programador en la implementación del Modelo y pueda así elegir la tecnología que crea más conveniente en cada caso.  Esto no significa que los beneficios en la utilización de Struts no se vean reflejados también en esta capa, ya que la manera en la que Struts permite implementar el Controlador ofrece un fuerte desacoplamiento entre éste y el Modelo, al tiempo que facilita una buena cohesión entre ambos. | |

|  |  |  |
| --- | --- | --- |
| **4.** | **PROCEDIMIENTO (DESCRIPCIÓN)** | |
| **A** | **HARDWARE REQUERIDO** | **SOFTWARE REQUERIDO** |
|  | * Procesador x86-x64 o AMD 2.5GHz + * RAM 4GB+(mínimo) * GPU 128MB+ | * Windows 7 o superior * MySQL * Web Browser: Google Chrome 30+, Firefox 24+, Opera 20+, Edge, IE12+ * NetBeans IDE 8.1 * Glassfish |
| **B** | **DESARROLLO DE LA PRÁCTICA** | |
|  | **Práctica 4: Modelo Vista Controlador utilizando Framework Struts(introducción)- Suma de dos números.**   1. Creación de un nuevo proyecto Web Application, adicionalmente escoger Struts con el framework a trabajar.      1. Establecer la siguiente estructura al proyecto.     Modelo  Controlador  Vista  Archivos de configuración  Dicha estructura muestra la división de archivos que corresponden a cada capa del MVC.   1. Referenciando al modelo, el paquete modelo tiene en sí una clase *OperacionesBasicas.java*  la cual básicamente contiene las reglas de negocio de la aplicación , en este caso la suma de dos números.     Esto se ve reflejado en el método *suma(ValidacionForm vf)*, el cual retorno la suma del número 1 y el número 2, mediante los atributos extraídos de la clase ValidacionForm.   1. Dentro del paquete *controladoresform* trae consigo la clase *ValidacionForm la cual se encarga de la encapsulación de los dos atributos definidos, numero1 y numero2.*      1. En el paquete *controladoresaction* se crea la clase que tiene el papel principal para el funcionamiento de la aplicación pues *OperationAction* es quien procesa la captura de los datos y quien devuelve una respuesta dependiendo de si se realizó con éxito la transacción.   Esta clase interactúa con los archivos de configuración struts.     1. Una vez finalizado la parte del modelo y el controlador, se continúa creando las vistas. 2. En *Webingreso.jsp* solo se crea el formulario a mostrarse, el cual consta de dos cuadros de texto para el ingreso de dos números y un botón que permita la suma entre ellos. Esto lo realizamos gracias a etiquetas html.      1. En *resultado.jsp* basicamente se hace referencia al atributo “respuesta” el cual tiene en si , el resultado de la operación.      1. Finalmente se procede a la edición de los archivos de configuración que struts los genera. 2. Empezando por *struts-config.xml*  lo que hay que definir es la clase de la cual vamos a trabajar(esta clase se encuentra dentro del controlador), esto lo hacemos dentro de las etiquetas <form-beans>.   Como segundo a modificar es las acciones a tomar <action-mapping> , es decir de *ValidacionForm* va a capturar los datos que provienen de webingreso mediante la operación /validar quien va controlar esta transacciones es *OperationAction* y va devolver el resultado en *resultado.jsp*     1. En el archivo web.xml simplemente hay que definir que archivo .jsp se ejecutará primero, esto se define en *<welcome-file-list>*     **Práctica 5: API de Struts, Interacción con Mysql.**   1. En primer lugar es necesario crear una base de datos que representa la relación existente entre categoría y productos.(Anexo 1) 2. Creación de un proyecto Web Application escogiendo el framework Struts. 3. Creación de la siguiente estructura en el proyecto.      1. Identificar que archivos pertenecen a cada una de las capas del MVC. 2. Dentro del paquete modelo tenemos dos clases, la primera *ConexionDB.java*  que como su nombre indica esta la codificación que permite la conexión con la BD, cabe recalcar que aparte de esta clase se debe agregar la librería de Mysql.     Se hallan dos métodos imprescindibles *getConexion()*  el cual permite establecer la conexión con la base de datos en base al correcto ingreso de parámetros y el método *cierraConexion()* el cual finaliza la conexión en el momento de finalizar la aplicación.    La clase GestionProductos básicamente consta con la lógica de negocio en la cual existen dos métodos, el uno *consultarCategoria()* el cual hace una consulta a la BD solicitando el id de la categoría de la tabla categoría.    El otro *consultarProducto()* hace un join entre las dos tablas para que según el id de la categoría de la tabla categoría , devuelva todos los productos que le pertenecen.   1. Continuando con la arquitectura MVC , es momento de crear los controladores, los cuales consta de dos paquetes.   El primero *controladoresform* tiene en sí, las clases *CategoriaForm* y *ProductoForm*, cada una encapsulando sus atributos.  Se encuentra también la clase *SeleccionForm* , la cual encapsula los datos del id de la categoría.  En el paquete *controladoresaction* se encuentran las clases principales que van a dirigir a la aplicación.  En *CategoriaAction* se establece la relación entre el modelo y la vista *categoriajsp.*  En *ProductosAction* se establece la relación entre el modelo y la vista *productosjsp.*   1. Finalmente realizamos las vistas de nuestra aplicación, siendo estas:   Ingreso.jsp.- es la pantalla principal de bienvenida  Categoría.jsp.- permite la selección del id de alguna de las categorías.  Producto.jsp.- se encarga de mostrar en una tabla la información obtenida de la selección anterior.   1. Los archivos de configuración a editar.   Struts-config.xml el cual permite el control de la dirección de los resultados. Es decir, *CategoriaForm* va a capturar los datos de */ingresar* y quien va a controlar la transacción es *categoriaAction* y va a ser devuelto en *categoría.jsp.*  Lo mismo sucede con *SelectionForm* captura un código de una categoría de listado lo procesa *productosAction* y se muestra en producto.jsp.    Web.xml declaramos la pantalla con la cual iniciara nuestra aplicación *ingreso.jsp*  **Práctica 6: Formulario de Autenticación utilizando el framework Strut**s   1. Es necesario la creación de una nuevo BD llamada test con la siguiente tabla. (Anexo 2) 2. Creación de la siguiente estructura en el proyecto.      1. Identificar que clases pertenecen a cada capa de la arquitectura MVC. 2. Empezando por el paquete modelo, encontramos la clase **Datos.java** en la cual se establece los parámetros necesarios para la conexión con la BD.   Y en la clase GestionCliente.java se encuentra la lógica de negocio, creando los métodos *validar(),* el cual selecciona de la tabla cliente el usuario y el password.    Mientras que el método *registrar(),* permite el registro de un nuevo cliente, el cual solicita el nombre, apellidos, usuario, password y email.     1. Continuando con la arquitectura MVC , es momento de darle forma al Controlador, el cual esta divido en dos paquetes :   Controladoresform: en el cual se encuentra la clase *RegistroForm* y se encarga de encapsular los atributos del cliente, mientras que la clase *ValidacionForm* permite la encapsulación de los atributos relacionados a la autenticación.  ControladoresAction: son las clases cerebro de la aplicación. Encontramos la clase *RegistroAction*  la cual se encarga del mapeo entre el modelo y la vista en este caso el archivo login.jsp,  Mientras que el ValidationAction, genera como vista dos tipos de respuesta, la una Bienvenidos.jsp en el caso de que la autenticación haya sido correcta caso contrario redirigirá a error.jsp.     1. Los archivos para mostrar los resultados (vista) son:   Login.jsp.- el primer archivo jsp que se abre al ejecutar la aplicación. En el solicita el usuario y el password.  Bienvenido.jsp.- en el caso de que los datos sean correctos mostrara una pantalla de bienvenida.  Error.jsp. En el caso de que sean incorrectos, le aparecerá un mensaje anunciado que no se encuentra registrado, permitiendo rellenar un formulario con sus datos para poder registrarse correctamente.  Registro.jsp: formulario solicitando el nombre, apellidos, usuario, password y email.   1. Finalmente se procede a la configuración de los archivos *struts-config.xml* en el cual se le debe mostrar como esta el funcionamiento de nuestra aplicación. Es decir se toman los datos de *ValidacionForm* que provienen de */validar* , *ValidarAction* procesa la transacción y devuelve los resultados dependiendo de la correcta(*Bienvenido.jsp*) o incorrecta (*Error.jsp*)autenticación.     Finalmente, el archivo *web.xml* permite la configuración de la página que dará inicio a la aplicación (*login.jsp*) | |
|  | **RESULTADOS DE LA PRÁCTICA** | |
|  | **Práctica 4:**    **Práctica 5:**    Figure 1.Pantalla de Bienvenida    Figure 2. Listado de Categorías    Figure 3.Resultado dependiendo el Id de categoría    Figure 4.Categorias existentes    Figure 5. Productos existentes  **Práctica 6:**    Figure 6. Autenticación    Figure 7. Datos correctamente ingresados.    Figure 8. Datos ingresados incorrectos    Figure 9. Formulario de registro | |
|  |  | |

|  |  |
| --- | --- |
| **5.** | **CONCLUSIONES Y RECOMENDACIONES** |
| ***Conclusiones:***   * Realizar una aplicación con la arquitectura MVC y utilizando el framework sin duda alguna simplifica el desarrollo de una aplicación, pues permite la iteración de estos dos en una misma clase. * La seguridad que se obtiene al realizar en este tipo de arquitectura es alta, pues la división de clases permite esta calificación de seguridad.   ***Recomendaciones:***   * El usar las etiquetas html , es muy necesario tomar en cuenta donde inicia y donde cierran si es que se desea tener un correcto funcionamiento. * Tomar en cuenta los nombres de los paquetes, las acciones y las clases que se utilizan en los archivos .jsp puesto que si no están bien escritas no muestra error alguno y se ejecuta el programa. * Tener en claro que tipo de archivo pertenece a cada capa de la arquitectura MVC para conocer su correcto funcionamiento. | |

|  |  |
| --- | --- |
| **6.** | **ANEXOS** |
| **Anexo 1:** Modelo entidad relación para la BD práctica.    Anexo 2: Tabla cliente perteneciente a la BD Test    Se adjuntará un CD con el proyecto generado para la correspondiente revisión por parte del Ingeniero. | |

|  |  |
| --- | --- |
| **7.** | **REFERENCIAS** |

# Bibliografía

Armando, P. (20 de 01 de 2015). *Tópicos en Java Mx*. Obtenido de Introducción a la Arquitectura de JavaEE: http://javaconfigmx.blogspot.com/2015/01/introduccion-la-arquitectura-de-java-ee.html

Martin, A. J. (2007). *Struts.* Madrid: RA-MA.

|  |  |
| --- | --- |
| **DICTADO POR:** | Julio César Mendoza Tello |