Objetivos de Aprendizaje: Examen Teoría 2023

Tema 1 Patrón MVC

¿Qué es el patrón MVC?

MVC (Modelo-Vista-Controlador) es un patrón en el diseño de software comúnmente utilizado para implementar interfaces de usuario, datos y lógica de control. Enfatiza una separación entre la lógica de negocios y su visualización, proporcionando una mejor división del trabajo y una mejora de mantenimiento.

¿Cuál es el objetivo fundamental del patrón MVC?

La separación de los datos de su representación visual y manipulación, facilitando así el desarrollo, mantenimiento y reusabilidad.

¿Qué es el modelo del MVC?

Es la capa donde se tratan los datos. Aquí se gestionan todos los accesos y actualizaciones a los mismos mediante mecanismos que se implementarán en este nivel.

¿Qué es la vista del MVC?

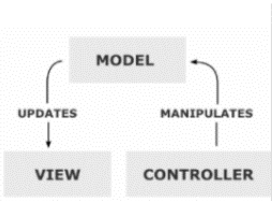
Es la capa donde se representa gráficamente el modelo. Aquí se requerirán los datos del modelo que se quieran presentar al usuario.

¿Qué es el controlador del MVC?

Es la capa que contiene la lógica de nuestra herramienta. Se encarga de atender a las acciones realizadas por el usuario sobre nuestro software. Estas acciones bien pueden requerir la vista o el modelo, por lo que el controlador hace de intermediario entre ambas.

¿Cómo son las dos maneras de interactuar entre si los componentes M, V y C?

1. Controlador-Modelo



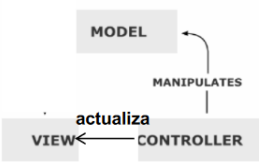
- Controlador modifica el Modelo pertinente.

Vista-Modelo

- Vista no puede cambiar el Modelo.
- Vista puede ser notificada cuando hay un cambio de estado en el Modelo.

2. Controlador-Modelo

- Controlador modifica el Modelo pertinente.



Controlador-Vista

- Controlador actualiza la Vista.

¿Cuáles son las ventajas de la separación de M, V y C?

La implementación es modular. Cualquier cambio en uno de los componentes no afecta a los demás, incluso se pueden implementar múltiples vistas con la misma instancia del modelo y del controlador.

¿Cómo es un modelo del MVC en una aplicación JAVA?

Son estructuras de datos, normalmente clases con diferentes atributos (se pueden generar a partir de un fichero o una BD tanto local como remota, o instanciarlas manualmente). Se le implementan las funciones get() y set() para consultar o modificar sus atributos, pudiendo ser accesibles de esta manera por el controlador.

Un ejemplo podría ser las clases Libro, Carro, BD_Libros o Item_Carro de la aplicación Java/Swing de las transparencias del tema 1.

¿Cómo es una vista del MVC en una aplicación JAVA?

Es la representación del modelo, puede ser una vista swing, estar definida por xml, etc. En resumen, lo forma la interfaz gráfica de la aplicación. También posee métodos para recibir eventos de usuario.

¿Cómo es un controlador del MVC en una aplicación JAVA?

Tiene una referencia a los modelos y a la vista, implementa el interfaz ActionListener y el método actionPerformed(ActionEvent event), encargados de gestionar los eventos de usuario. También se encarga del acceso y la modificación del modelo.

¿Cómo interactúan entre si los componentes M, V y C en una aplicación JAVA?

1. En el main se crean los objetos controlador y algunos modelos.
2. El controlador crea la vista y le asigna el controlador correspondiente.
3. El controlador obtiene los datos del modelo, los escribe en la vista al cual fue asignado e inicia esta.

¿Cómo es un modelo del MVC en una aplicación Android?

Son estructuras de datos, normalmente clases con diferentes atributos (se pueden generar a partir de un fichero o una BD tanto local como remota, o instanciarlas manualmente). Se le implementan las funciones get() y set() para consultar o modificar sus atributos, pudiendo ser accesibles de esta manera por el controlador.

Un ejemplo sería las clases Item, Pedido, LineasPedido y Linea de nuestra aplicación.

¿Cómo es una vista del MVC en una aplicación Android?

La vista de la aplicación la compone los diferentes archivos .xml que se generan para cada actividad creada.

Por ejemplo, para la actividad CatalogoActivity.java de nuestra aplicación se genera un fichero .xml llamado catalogo_view.xml que contiene todos los atributos necesarios para representar gráficamente los Item del modelo.

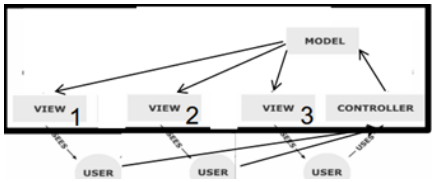
¿Cómo es un controlador del MVC en una aplicación Android?

Todas aquellas clases que hereden de la clase Activity se consideran controladores. Estas contienen todos los métodos necesarios para poder gestionar las acciones del usuario sobre nuestra aplicación.

Por ejemplo, para la actividad CatalogoActivity.java (la cual hereda la clase AppCompatActivity.java) tiene diferentes métodos para atender al item que el usuario seleccione de la vista.

¿Cómo interactúan entre si los componentes M, V y C en una aplicación Android?

El modelo envía actualizaciones a la vista cuando se realizan cambios en los datos. La vista recibe actualizaciones del modelo y envía eventos al controlador cuando el usuario interactúa con la interfaz. El controlador recibe eventos de la vista y realiza operaciones en el modelo.



Tema 2 Aplicaciones Distribuidas: RMI

¿Qué es una aplicación distribuida?

Es una aplicación que ejecuta diferentes partes de código en varios ordenadores. El código en cada ordenador puede invocar código y pasar y recibir datos de otros ordenadores. Este mecanismo se llama “llamada a procedimientos remotos”.

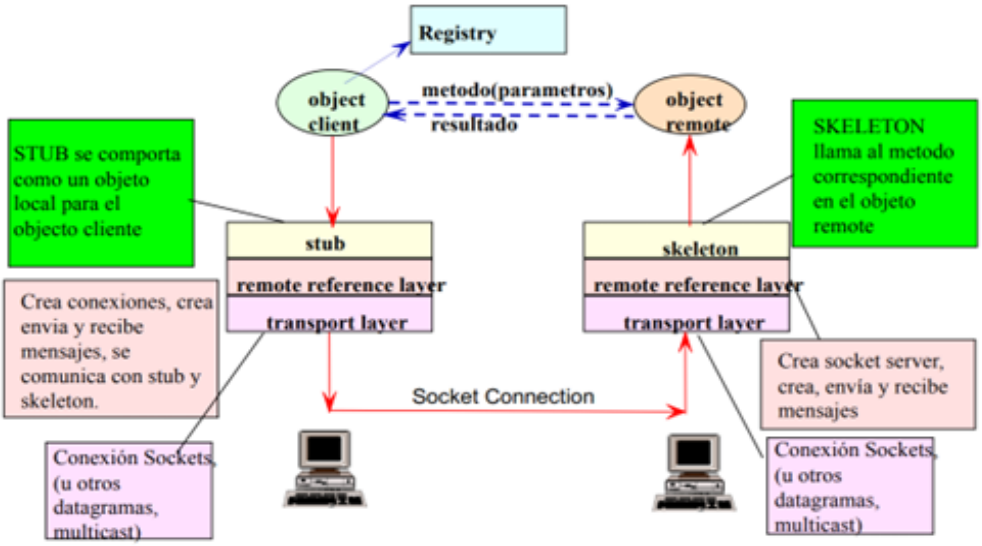
¿Qué es RMI?

Es una implementación orientada a objetos del paradigma de llamadas a procedimientos remotos. Es una implementación solo en java. Los objetos de unas aplicaciones están distribuidos en diferentes ordenadores:

- Objeto remoto: provee métodos remotos que pueden ser invocados por programas clientes

- Objeto cliente: invoca los métodos remotos

¿Cómo es la arquitectura de RMI?



¿Para qué se utiliza la serialización de objetos en RMI?
Se utiliza para convertir objetos en una estructura lineal adecuada para transmitirlos por la red (o para guardarlos en ficheros)

Tema 2 Aplicaciones Distribuidas: WS y REST

¿Qué es un "Servicio Web"?

Una aplicación web identificada por un URI, cuyos interfaces y enlaces pueden ser definidos, descritos y descubiertos mediante artefactos XML.

Soporta la interacción directa con otras aplicaciones web.

El interfaz define una colección de operaciones que son accesibles de forma remota usando mensajes XML.

¿En qué se diferencia un "Servicio Web" de RMI?

Es independiente del lenguaje. Está basado en protocolos web: Http, xml.

¿Cuáles son las ventajas e inconvenientes de un "Servicio Web"?

Ventajas:

- Se puede acceder de diferentes plataformas y lenguajes de programación
- Pueden manejar grandes volúmenes de tráfico y demanda de clientes
- Comunicación en tiempo real
- Pueden ser accedidos desde cualquier lugar

Desventajas:

- Vulnerable a ataques de seguridad
- Latencia
- Requiere de conexión a internet.

¿Cómo es la arquitectura de un "Servicio Web"?

¿Cómo es un servidor de un "Servicio Web"?

Es una aplicación o software que se ejecuta en un servidor web y proporciona funcionalidad a través de WWW. Utiliza el protocolo de comunicación HTTP para recibir y enviar solicitudes y respuestas.

¿Cómo es un cliente de un "Servicio Web"?

Es una aplicación que utiliza el servicio web para realizar operaciones en el servidor remoto. Puede ser cualquier aplicación que pueda realizar solicitudes HTTP y procesar respuestas HTTP. Esto incluye, aplicaciones web, móviles y de escritorio entre otros.

¿Cómo se desarrolla una aplicación de un "Servicio Web"?

Los desarrolladores implementan una aplicación en JAVA. Hacen peticiones GET, POST

Utilizan librerías o SDKs que internamente implementan peticiones web y transforman respuestas XML en objetos locales.

¿Qué es un "Servicio REST"?

Son servicios web más sencillos. Utiliza el protocolo http. No utiliza ni UDDI ni WSDL, solo operaciones básicas CRUD

¿En qué se diferencia un "Servicio REST" de un "Servicio Web"?

Un servicio WEB incluye cualquier servicio que se pueda acceder a través de la web, mientras que un servicio REST es un tipo específico de servicio web que se basa en el protocolo HTTP y los principios de la arquitectura REST.

En un servicio web los recursos se identifican mediante URIs, en un servicio REST mediante URLs.

¿Cuáles son las ventajas e inconvenientes de un "Servicio REST"?

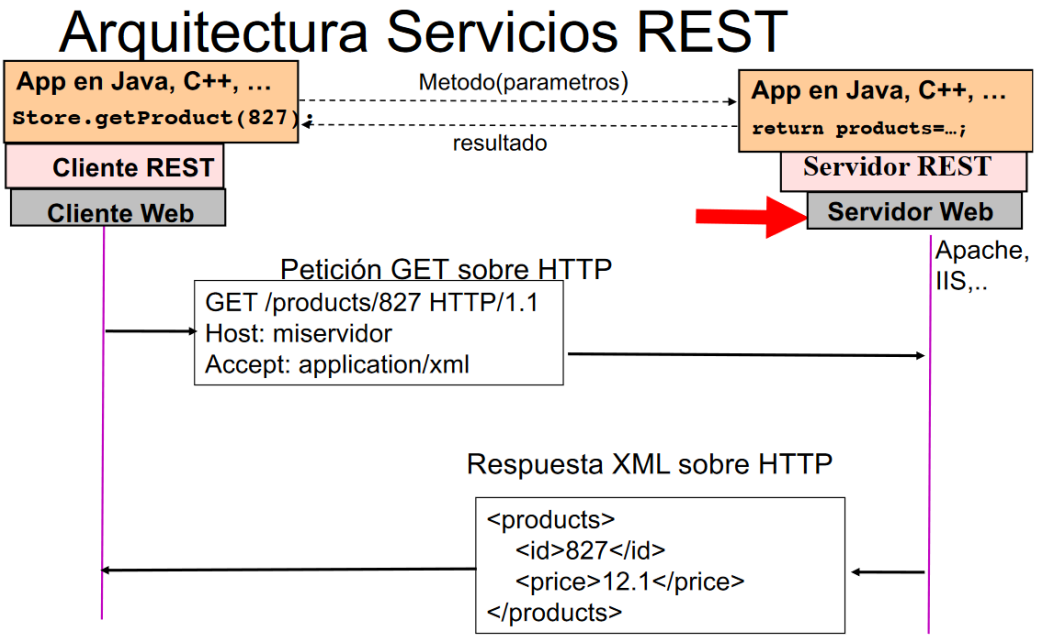
Ventajas:

- Escalabilidad. Se pueden manejar múltiples solicitudes de clientes simultáneamente
- Utiliza formatos de datos livianos (JSON). La transferencia de datos es rápida.
- Se pueden utilizar con cualquier lenguaje de programación y plataforma

Inconvenientes:

- Complejidad
- No proporciona mecanismos de seguridad integrados
- Solo admiten operaciones estándar de HTPP: GET, POST, PUT, DELETE

¿Cómo es la arquitectura de un "Servicio REST"?



La arquitectura de este servicio se basa en la transferencia de recursos (en este caso métodos, objetos o parámetros) sobre HTTP utilizando los diferentes métodos de este (GET, POST, PUT y DELETE). REST recibe las respuestas a sus peticiones en formato XML o JSON entre otros. Suele prevalecer el JSON por ser más sencillo de parsear.

¿Cuáles son las operaciones básicas de un "Servicio REST"?

Las cuatro operaciones CRUD que proporciona REST son los métodos usados en HTTP:

- Create → POST
- Read → GET
- Update → PUT
- Delete → DELETE

De esta manera es muy fácil de depurar y sus librerías son sencillas. Sin embargo, hay que adaptar el diseño de la aplicación a este servicio.

¿Cómo se desarrolla una aplicación de un "Servicio REST"?

En primer lugar, debemos seleccionar un servidor web que provea un servicio REST definido con una API (conjunto de operaciones remotas que provee el servidor. Application Programming Interface para poder manipularlo. A continuación, dependiendo de la tecnología y lenguaje que usemos, deberemos declarar e implementar los métodos, funciones y estructuras de datos que utilizaremos para comunicarnos con el servidor REST y usar su API. Por último, debemos recoger la información y ensamblarla de acuerdo con las estructuras declaradas anteriormente.

¿Qué es el API de un servidor REST?

Una API (Application Programming Interface) es un conjunto de operaciones remotas que provee el servidor para poder comunicarse con él y acceder a sus recursos, enviar solicitudes y recibir respuestas. Todo esto a través de una interfaz de programación.

¿Por qué son tan populares los servicios REST?

La razón con mayor peso es la facilidad y sencillez de uso e implementación frente a los servicios web más tradicionales. Además, genera poco tráfico y la programación es fácil debido a la cantidad de documentación, librerías e implementaciones existentes en los diferentes lenguajes. Esta simpleza en su arquitectura permite la interoperabilidad, es decir, ser consumido por diferentes clientes independientemente de su tecnología. También su bajo acoplamiento les permite no tener un servidor de aplicación asociado, y los cambios que sufren no afectan directamente a los clientes.

¿Cómo es un servicio Agregador de servicios REST?

Este servicio se aloja en un servidor donde la aplicación web, móvil o REST lanza peticiones contra él. El servidor donde se aloja establece conexiones con diferentes servicios REST para recopilar y combinar información presentándola luego a las aplicaciones conectadas a él.

Este tipo de servicio puede ser útil cuando se desea obtener información de diferentes fuentes que ofrecen servicios REST y se quiere presentar la información de forma integrada y organizada.

¿Cómo es una aplicación Agregadora de servicios REST?

Al contrario que en el servicio, aquí no existe un servidor que haga de intermediario. La propia aplicación hace las peticiones a los diferentes servicios REST de los que quiere obtener información. Por tanto, la propia aplicación procesa las peticiones, las integra y presenta los datos de forma ordenada.

Tema 3a Arquitectura J2EE Intro

¿Qué es una arquitectura Frontend-backend?

También se le conoce como arquitectura cliente - servidor. Es un patrón de diseño de aplicaciones web y móviles que divide la aplicación en dos partes fundamentales permitiendo la modularidad, facilitando el mantenimiento y el desarrollo:

- Frontend: Es la parte de la aplicación que se ejecuta en el lado del cliente, es decir, un navegador o aplicación móvil. Esta parte se encarga de la interacción con el usuario y la presentación de los datos. Los datos son pedidos al backend para poder presentarlos y gestionar las interacciones. Estos datos son pedidos a través de un servicio REST.

- Backend: Es la parte de la aplicación que se ejecuta en el lado del servidor. Esta parte se encarga de procesar las peticiones del frontend proveyendo al mismo los datos y servicios necesarios. Normalmente el backend se compone de una base de datos donde se guardan los modelos de datos y un servidor REST que atiende a las peticiones y recoge datos de la base.

¿Cómo es el Backend en una arquitectura Frontend-Backend?

Es la parte de la aplicación que se ejecuta en el lado del servidor. Esta parte se encarga de procesar las peticiones del frontend proveyendo al mismo los datos y servicios necesarios. Normalmente el backend se compone de una base de datos donde se guardan los modelos de datos y un servidor REST que atiende a las peticiones y recoge datos de la base.

¿Cómo es el Frontend en una arquitectura Frontend-Backend?

Es la parte de la aplicación que se ejecuta en el lado del cliente, es decir, un navegador o aplicación móvil. Esta parte se encarga de la interacción con el usuario y la presentación de los datos. Los datos son pedidos al backend para poder presentarlos y gestionar las interacciones. Estos datos son pedidos a través de un servicio REST.

El frontend puede implementarse de muchas maneras, algunos ejemplos son: una aplicación móvil o web, un SPA (Single Page App) o incluso una página web con un servidor al que accede mediante HTTP (el cual accede al backend a través de REST).

¿Qué diferencias hay entre una arquitectura J2EE y una arquitectura Frontend-Backend?

Mientras que en la arquitectura frontend - backend no es necesario atenerse a una tecnología concreta o una implementación específica, en J2EE (Java 2 platform Enterprise Edition) se suele usar mayoritariamente páginas web con un servidor web interconectado con HTTP en el frontend, y un servidor REST conectado a una base de datos en el backend. Tanto el servidor web como el REST deben estar implementados en Java.

¿Qué tecnologías se utilizan en una arquitectura Web J2EE?

Cómo sugiere el nombre, se utilizan tecnologías que implementan Java. En la parte del frontend, en el servidor web suele ser frecuente el uso de Servlets y del patrón de diseño Modelo-Vista-Controlador. Mientras, en el servidor REST del backend, las tecnologías predominantes son JPA (Java Persistence API) y EJB Beans (Enterprise Java Beans).

Tema 3b Servlets

¿Qué es un Servlet?

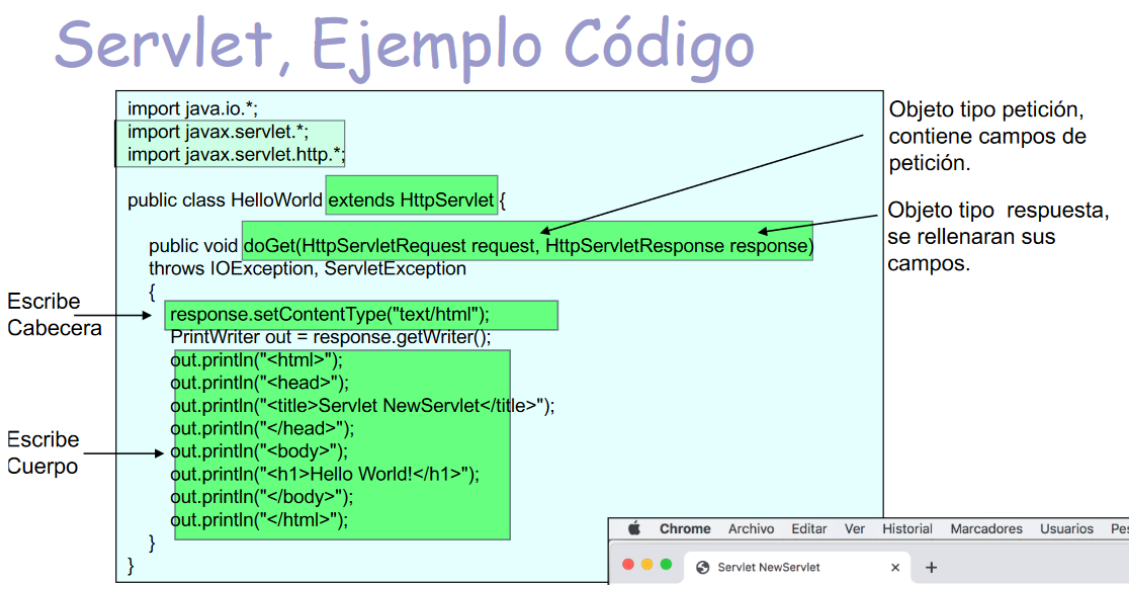
Un servlet es un pequeño programa en Java que se ejecuta dentro de un servidor web. Los servlets reciben y responden peticiones de diferentes clientes web, normalmente a través del protocolo HTTP.

Esencialmente un objeto Java que responde a las solicitudes HTTP de un cliente y puede generar una respuesta dinámica. Los servlets se ejecutan en un contenedor de servlets, que es un entorno de tiempo de ejecución proporcionado por un servidor web, como Apache Tomcat o Jetty. El contenedor de servlets se encarga de la administración del ciclo de vida del servlet, así como de la comunicación entre el servlet y el servidor web.

¿Qué es un contenedor de Servlet?

Los servlets se ejecutan en un contenedor de servlets, que es un entorno de tiempo de ejecución proporcionado por un servidor web, como Apache Tomcat o Jetty. El contenedor de servlets se encarga de la administración del ciclo de vida del servlet, así como de la comunicación entre el servlet y el servidor web.

¿Cómo es el código básico de un Servlet?



¿Para qué se utiliza el método doGet() de un Servlet?

Es un método de la clase HttpServlet que se usa para obtener datos y tratarlos. Este método hace una llamada GET a través del protocolo HTTP y necesita dos parámetros:

- Uno del tipo HttpServletRequest: Contiene todos los campos necesarios de la petición a realizar.
- Otro del tipo HttpServletResponse: Obtiene todos los campos recibidos de la petición anterior y se escriben en el.

Es necesario implementarlo en todas las clases de nuestro proyecto que hereden de HttpServlet y quieran realizar peticiones a algún recurso.

¿Para qué se utiliza un objeto de la clase HttpServletRequest?

Se utiliza para hacer peticiones de tipo HTTP. Proporciona acceso a los datos de las cabeceras HTTP, cookies, parámetros pasados por el usuario, etc, sin tener que parsear nosotros a mano los datos de formulario de la petición.

¿Para qué se utiliza el método getParameter() de HttpServletRequest?

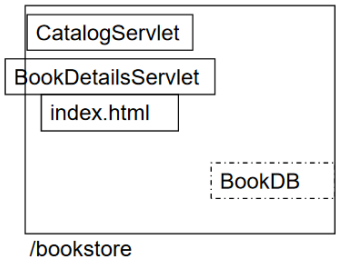
Devuelve el valor de un parámetro de solicitud pasado como argumento como String, o null si el parámetro no existe. Solo se debe usar este método cuando el parámetro tiene solo un valor. Si el parámetro puede tener más de un valor, se debe usar getParameterValues().

¿Para qué se utiliza el descriptor de despliegue web.xml?

Es un componente de aplicaciones J2EE que describe cómo se debe desplegar (o implantar) una aplicación web. Resuelve que clases corresponden a los servlets (<servlet-name>miServlet</servlet-name> <servlet-class>ClaseServlet</servlet-class>) y hace un mapeado (<servlet-mapping>{…}</servlet-mapping>) del servlet a la URL correspondiente de la aplicación web (<url-pattern>/mi_url</url-pattern> esto va dentro del servlet-mapping y después de servlet-name).

¿Qué es un ServletContext en una aplicación Web con Servlets?

El ServletContext es un objeto que permite compartir información entre los distintos componentes objetos entre varios servlets dentro del mismo contexto (Conjunto de servlets, páginas web y otros recursos que forman una aplicación.). Es equivalente a la clase application en Android. Un ejemplo de contexto donde se podría crear varios ServletContexts podría ser el siguiente. Aquí el CatalogServlet, bookDetailsServlet y en index.html formarían un contexto



¿Cómo se utiliza un objeto ServletContext?

Dentro del objeto de contexto de nuestra aplicación podremos establecer una serie de atributos, que serán globales dentro de ella. Estos atributos son un conjunto de pares <*nombre, valor*> que podemos establecer y consultar desde los distintos servlets de nuestra aplicación web. El nombre del atributo será una cadena de texto (String), mientras que el valor podrá ser cualquier objeto java (Object).

Para consultar el valor de un atributo utilizaremos:

```
Object valor = context.getAttribute(nombre);
```

Daremos valor a un atributo con:

```
context.setAttribute(nombre, valor);
```

Podemos también eliminar un atributo:

```
context.removeAttribute(nombre);
```

Lo cual dejará el atributo a null, igual que si nunca le hubiésemos asignado un valor. Por último, con

```
Enumeration enum = context.getAttributeNames();
```

Obtenemos la lista de nombres de atributos definidos en el contexto.

Hay que hacer notar en este punto, que el objeto de contexto aparte de ser propio de cada aplicación web, es propio de cada máquina virtual Java. Cuando trabajemos en un contexto distribuido, cada máquina ejecutará una VM distinta, por lo que tendrán también objetos de contexto diferentes. Esto hará que, si los servlets de una aplicación se alojan en máquinas distintas, tendrán contextos distintos y este objeto ya no nos servirá para comunicarnos entre ellos. Veremos más adelante formas alternativas de comunicación para estos casos.

¿Qué es un HttpSession en una aplicación Web con Servlets?

Es una clase que permite almacenar información de estado sobre la sesión de un cliente. Internamente puede utilizar cookies, o parámetro de sesión en la URL.

Aquí los datos que se guardan solo los verá el usuario con dicha sesión abierta. No se comparten para todos como con ServletContext.

¿Cómo se utiliza un objeto HttpSession?

El funcionamiento del sistema de sesiones es relativamente sencillo. Cada vez que un usuario crea una session accediendo a una página (que la genere) se crea un objeto a nivel de Servidor con un HashMap vacío que nos permite almacenar la información que necesitamos relativa a este usuario. Realizado este primer paso se envía al navegador del usuario una Cookie que sirve para identificar y asociar el HashMap que se acaba de construir para que pueda almacenar información en él. Este HashMap puede ser accedido desde cualquier otra página permitiéndonos compartir información.

Tema 3c Aplicaciones Web MVC con Servlets

¿Cómo se implementa el modelo en una aplicación Web con Servlets?

Se implementa definiendo las clases de los objetos que almacenan y dan acceso a los datos de la aplicación. Existen herramientas que automáticamente generan código de clases a partir de tablas SQL u otras BBDD. El modelo es un objeto java que se puede guardar en la BBDD Y además, no tiene una representación de los datos.

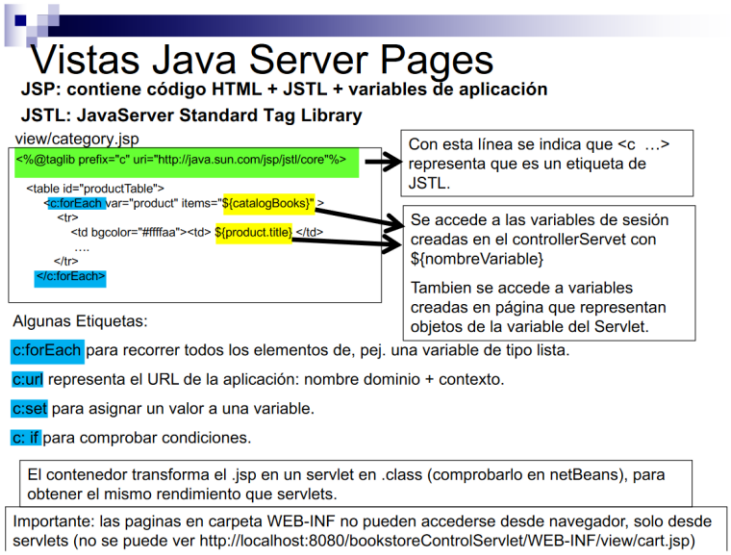
Si se guardan los datos en ficheros implementamos un hashmap, si se guardan en una BD usaremos JDBC en la implementación. Ambos usan el mismo interfaz.

¿Cómo se implementa el controlador en una aplicación Web con Servlets?

Se implementa mediante la clase ControllerServlet, que recibe las peticiones del contexto, configurando adecuadamente el fichero web.xml:

1. En método Init de Servlet se define el modelo. O más de un modelo si lo hubiera
2. Request.getServletPath() obtiene la ruta de la petición de la página Web del usuario.
3. En doGET, doPOST, ... con un SWITCH se comprueba el userPath y se ejecuta un código u otro. (corresponde a las acciones de todos los servlets del ejemplo anterior) Y los datos de los modelos se escriben en variables que pueden ser accedidas por las vistas.
4. Antes de finalizar el controller con esta instrucción se pasa el control al View que corresponda al userPath que escribamos al finalizar el case.

¿Cómo se implementa la vista en una aplicación Web con Servlets?



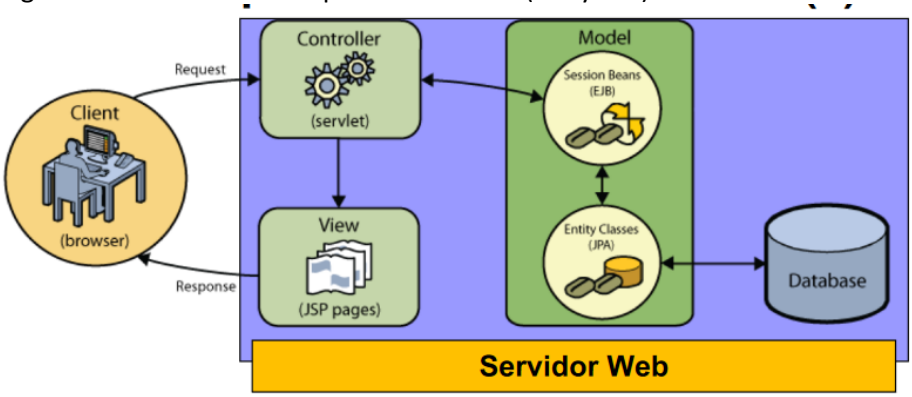
¿Cómo interactúan entre si modelo, vista y controlador en una aplicación Web con Servlets?

La interacción que hacen estos componentes es de la siguiente manera: el cliente hace peticiones REQUEST al controlador que responden a las acciones del usuario, este escribe los datos en el modelo y a su vez manda a la vista que los represente.

- Vista: Para las vistas se requieren de ficheros JSP.

- Controlador: Para los controladores se requieren servlets.

- Modelo: Para el modelo se utilizan clases java creadas por el usuario o por herramientas de generación automática a partir de una BD (EJB y JPA).



Tema 3d ORM, JPA

¿Qué es JDBC Java Database Connectity?

Son librerías java que se utilizan para hacer consultas a bases de datos mediante SQL, sirve para conectar programas java con BBDD, Utiliza clases como:

- DriverManager: Para cargar drivers
- Connection: Para conectar a BBDD
- Statement: Para ejecutar sentencias SQL y enviarlas a la BBDD
- ResultSet: Para almacenar resultados de consultas

¿Qué ventajas y desventajas tiene JDBC Java Database Connectity?

Ventajas: Con JDBC, podemos conectarnos a cualquier base de datos, es el mecanismo de obtención y procesamiento más antiguo y básico.

Desventajas: para poder usar JDBC y realizar las queries que necesitemos, es necesario dominar el SQL de la BBDD. Para obtener y utilizar los objetos de las bases de datos, es que necesitamos mucho código, es decir que no es una forma limpia de obtener todos los datos.

¿Que es ORM Object Relation Mapping?

ORM (Object Relational Mapping) es una técnica de programación que permite mapear objetos de una aplicación a tablas de una base de datos relacional. En lugar de escribir código SQL directamente para interactuar con la base de datos, se utilizan clases Java (objetos) para representar las tablas de la base de datos y se utilizan métodos Java para interactuar con ellos.

El objetivo principal de ORM es simplificar el proceso de interacción con la base de datos, lo que puede ser tedioso y propenso a errores. ORM también puede mejorar el rendimiento y la escalabilidad de una aplicación, ya que proporciona una capa de abstracción que permite a los desarrolladores trabajar con objetos en lugar de tablas.

¿Por qué se usa ORM?

Nos permite abstraernos de la base de datos y hacer el desarrollo más rápido y sencillo trabajando con objetos. Esto mismo nos permite también cambiar entre bases de datos sin tener que alterar el código de nuestra aplicación, y tener un código más sencillo y limpio ya que las consultas SQL están dentro de los objetos ORM.

¿Qué ventajas y desventajas tiene ORM?

Ventajas: Las mencionadas en la pregunta anterior.

Desventajas: Como son las relaciones de herencias que son más difíciles de implementar. En ORM hay atributos que pueden estar en varias clases, lo que hace que la búsqueda de estos atributos sea compleja de realizar.

¿Qué es JPA Java Persistance API?

Son librerías Java diseñadas para hacer ORM es decir, mapeo standard de las tablas de la BBDD en clases java, las clases entity corresponden con las tablas y relaciones de las BBDD, las clases entity manager se encarga de conectar los objetos de la clase entity con la BBDD

¿Qué son las clases Entity?

En JPA, las clases Entity son clases Java que se utilizan para mapear objetos a tablas de base de datos relacionales. Una clase Entity representa una tabla en la base de datos y cada instancia de la clase Entity representa una fila en esa tabla.

Las clases Entity se definen utilizando la anotación @Entity y se pueden mapear a una tabla específica en la base de datos utilizando la anotación @Table. Cada propiedad de la clase Entity representa una columna en la tabla y se puede mapear a una columna específica utilizando las anotaciones @Column, @JoinColumn, o @JoinTable.

¿Qué es un Entity Manager?

En JPA, un Entity Manager es una clase que proporciona métodos para interactuar con las entidades en la base de datos. El Entity Manager es responsable de conectar con la BBDD y guardar las clases Entity en él.

Tema 3e capas J2EE, EJB

¿Cuáles son las capas de una arquitectura Web J2EE?

Las tres capas son: Web/servlets, EJB Beans y Datos. En MVC la primera correspondería con la vista y parte del controlador, la segunda con el controlador y la última con el modelo, ya que los datos suelen alojarse con frecuencia en una BBDD.

El primero contiene un EJB Client para conectarse con el segundo, y este contiene JPA para poder hacer ORM entre el y la BBDD.

¿Qué función realiza la capa Web/Servlets en una arquitectura J2EE?

Aquí se ejecuta la vista y el controlador (el cliente EJB que se comunica con el servidor EJB Beans). Las contiene todos los JSPs de las vistas así como los controladores de los diferentes Servlets.

¿Qué función realiza la capa EJB/Beans en una arquitectura J2EE?

Esta capa se encarga de implementar un EntityManager que gestiona los diferentes objetos Entity para cumplir con el JPA y establece una conexión con la BBDD. Estos EJB Beans son objetos remotos invocados por RMI.

Un EJB también se denomina fachada (Facade) por que implementa un CRUD para acceder a un conjunto de objetos: create() (CREATE); find(); findAll() (2 READ); edit() (UPDATE); remove() (DELETE);

¿Cómo se implementa un EJB/Bean?

Para declarar un EJB Bean se utiliza la anotación @Stateless. Para poder encapsular un EntityManager se debe indicar con @PersistanceContext lo cual indica una conexión con la BBDD. Se deben implementar también los métodos necesarios para poder realizar el CRUD (los de la pregunta anterior).

¿Cómo se comunica la capa Web/Servlets y la capa EJB/Beans?

A través de un cliente EJB que comunica ambos. Internamente este cliente implementa diferentes llamadas remotas usando RMI. Por ejemplo, si queremos referenciar desde web/servlets a través del cliente al servidr EJB/Beans haríamos algo tal que así:

```
@EJB
private ProductFacade bookDB;
```

Esto instancia una referencia al servidor.

Tema 4 SPA

¿Qué es una aplicación Single Page App SPA?

Es una aplicación web o es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio. En un SPA todos los códigos de HTML, JavaScript, y CSS se carga de una vez o los recursos necesarios se cargan dinámicamente como lo requiera la página y se van agregando, normalmente como respuesta de las acciones del usuario. La página no tiene que cargar otra vez en ningún punto del proceso tampoco se transfiere a otra página.

¿Qué ventajas y desventajas tiene una aplicación SPA respecto a una Web Clásica?

Ventajas:

En un SPA, todos los códigos necesarios (HTML, JavaScript y CSS) se recuperan con una única carga de página, o los recursos apropiados se cargan dinámicamente y se agregan a la página según sea necesario, generalmente en respuesta a las acciones del usuario.

La página no se recarga en ningún punto del proceso, ni controla la transferencia a otra página.

SPA mejor que Web Clásica para conexiones en movilidad (Smartphone, 3G), y funciona igual con buena conexión.

Desventajas: Ya que bastante lógica del negocio se implementa en el lado cliente, permitiendo que cualquiera pueda leerla en nuestro código javascript por lo que hay que reforzar la seguridad.

¿Qué ventajas y desventajas tiene una aplicación SPA respecto a una App nativa?

Ventajas:

Se programa en javascript por lo que es multiplataforma.

No hace falta subirla aun store para que se pueda usar.

Desventajas:

Al no presentarse en un store puede presentar menor fiabilidad para el cliente.

Como no está integrada en el SO del móvil no puede utilizar los sensores de este.

¿Cuáles son las ventajas e inconvenientes del lenguaje Javascript?

Ventajas:

- Es multiplataforma
- Tiene una sintaxis muy parecida a java por lo que usuarios de ese lenguaje pueden acceder rápidamente a javascript
- Es interactivo por lo que puede depurar sin necesidad de compilar
- Es soportado por los navegadores más populares (Google, firefox) y es compatible con los dispositivos más modernos.
- Implementación de patrón MVC con librerías como angular.js y react.js
- Se han implementado servidores backend que se programan en Javascript por lo que es muy útil para no tener que cambiar de lenguaje de programación y además soporta peticiones asíncronas en el servidor (por ejemplo, la librería de parse.com utilizada en prácticas).

Desventajas:

- Es difícil de depurar ya que es un lenguaje no tipado.
- No hay clases solo objetos y funciones por lo que se produce un código difícil de mantener y modificar