

# Arquitectura Java para el Cuarto Ejercicio

José Antonio Ruano Ampudia  
Técnico Superior de Proyecto Informático

# Sumario

- Introducción
  - Arquitectura en n-capas
- Arquitectura y el Cuarto Examen
- Java y su modelo empresarial
  - Capa de Presentación
  - Capa de Aplicación
  - Capa de Negocio
  - Capa de Acceso a Datos
  - Capa de Datos
- Arquitectura SOA

# Introducción

- Arquitectura en n-capas ¿Por qué?
  - ¿Por qué no hacerlo todo con JSP?
  - Si funciona, ¿cuál es la motivación de separar y “complicar” toda la solución?
  - ¿Se entiende realmente lo que se pone en los exámenes?

# Introducción

- VENTAJA:
  - Estructura clara y sencilla: facilidad de saber qué tocar en un proyecto complicado.
  - ¿Te imaginas arreglar un error en un proyecto que no es tuyo? ¿Cuánto tarda un nuevo desarrollador en entrar en el proyecto?
  - Otras ventajas:
    - Reutilización.
    - Mayor flexibilidad en la introducción de nuevos módulos.
    - Escalabilidad.

# Arquitectura y el Cuarto Examen

- ¿Cómo influye ya la arquitectura en el cuarto examen?
  - Como tal, el año pasado, no han pedido poner como tal una arquitectura técnica.
    - Aunque alguien la haya puesto.
- ¿Significa esto que no valga para nada esta sesión?
  - No, puede que este año la pidan.
  - Vendrá bien para saber organizar lo que nos piden en el examen → Recordar el apartado de introducción.

# Java y su modelo empresarial

- Java ofrece un montón de posibilidades para cada una de las capas.
  - El modelo Java Enterprise Edition contiene todo lo necesario para trabajar cada una de las diferentes capas.
  - **Existen otros productos que se pueden utilizar en cada una de las capas, las cuales son más usados incluso que algunos de las oficiales de Oracle (Sun).**
- Java es un conjunto de especificaciones:
  - Los famosos *Java Specification Requests* (JSR) → Se utilizan para proponer y especificar cambios en la plataforma Java.

# Capa de Presentación

- Esta capa es la encargada de realizar la comunicación usuario-sistema.
  - Recibe las peticiones del usuario.
  - Muestra los datos al usuario.
- Importante que no se haga nada de lógica de negocio en esta capa.
  - ¿Comprobación de errores?
    - Si por mera eficiencia → En general, las comprobaciones que se hacen de errores, por seguridad, se deben hacer en el servidor.

# Capa de Presentación

- En el mundo java, en general se utiliza para ello jsp:
  - Html con java embebido.
  - AJAX (Wikipedia) → **Ajax**, acrónimo de *Asynchronous JavaScript And XML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*).
    - Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.
    - Es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.
  - AJAX y accesibilidad.
    - Posible pero complejo.
    - En general, la accesibilidad como tal es compleja.

# Capa de Presentación

- Se combina con plantillas (tiles) que te dan la posibilidad de reutilizar código de manera sencilla.
  - Apache Tiles es un framework de plantillas utilizado para simplificar el desarrollo de interfaces de usuario en aplicaciones Web.
    - Permite definir fragmentos de páginas que se pueden ensamblar en páginas completas en tiempo de ejecución.
- Struts también tiene sus propios tiles.

# Capa de Aplicación

- Es la capa que coordina toda la actividad de la aplicación.
- En algún lado se tiene que establecer una separación entre las diferentes capas.
- Modelo vista-controlador:
  - (Wikipedia): es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos.
  - ¿Para qué me vale esto? Con el modelo vista-controlador comunico las diferentes capas de mi sistema: en concreto, voy a comunicar la presentación con la lógica de negocio.

# Capa de Aplicación

- Struts
  - Es, quizá, la implementación del modelo MVC más conocida y utilizada en Java.
  - Actualmente están en la versión 2.
  - A través del fichero de configuración struts-config.xml se establece la comunicación con los diferentes jsp con las acciones que se desencadenan.
  - Obviamente hay que declarar en el web.xml que vas a utilizar struts.

# Capa de Aplicación

- Otra posibilidad es Java Server Faces (JSF)
  - Es una especificación:
    - JSR 127, que definía JSF 1.0 y 1.1.
    - JSR 252 que define JSF 1.2.
    - JSR 314 para JSF 2.0.
    - Es por lo tanto un estándar.
  - Tecnología que simplifica la creación de interfaces de usuario en aplicaciones Web.
  - La programación está más enfocada a componentes que responden a eventos.
  - Diferentes implementaciones para la especificación
    - Apache: MyFaces.
    - Implementación oficial de Sun (Oracle).
    - ADF Oracle la cual fue donada a Apache.

# Capa de Aplicación

- Motor de Workflow.
- Herramienta que nos sirve para modelizar un procedimiento administrativo.
  - Se diseña el flujo con BPMN, generalmente a través de interfaces amigables.
  - Se transforma el flujo a XPDL o BPEL.
  - Decide el flujo de la solicitud.
    - Puede avisar a las personas correspondientes para continuar un trámite.
- Se integrará con el resto de módulos para capturar los eventos que generan las transiciones de estados. Para ello se utilizarán las WAPI del Workflow.
- ◎ JBPM es uno de los productos más utilizados para esta capa.

# Capa de Lógica de Negocio

- Aquí está las entrañas de nuestra aplicación.
- Lo que realmente tiene importancia en nuestro proyecto, lo que le diferencia del resto.
  - Se encuentran los módulos de diseño con los que se trabaja.
- Recibe las peticiones a través del controlador, se comunicará con los datos a través de la capa de acceso a datos.

# Capa de Lógica de Negocio

- ¿Qué utilizamos para programar?
  - ¿Por qué no POJO's?
  - Un POJO es lo que se conoce como Plain Old Java Object.
  - Es decir, las clases java de toda la vida.
  - ¿Para qué complicarnos más la existencia en la programación de nuestra lógica de negocio?

# Capa de Lógica de Negocio

- EJB's: Enterprise Java Bean.
- Hasta la versión 2 de los EJB, según la plataforma JEE era lo que se proponía para programar la lógica de negocio.
  - Sobre todo en entornos distribuidos.
  - Demasiado pesados en cuanto a ejecución.
  - Requerían de un servidor de aplicaciones para su ejecución: ya no nos vale Tomcat.
  - Eran un infierno su programación.
- La especificación EJB 3.0 ha derivado en otra cosa:
  - Session Bean y Message-driven bean para la lógica de negocio
  - Entity Bean se ha reemplazado por Java Persistence API, utilizados para realizar la persistencia de las aplicaciones.

# Capa de Acceso a Datos

- En esta capa se hace el acceso ordenado a la base de datos y demás almacenes de datos.
- Modelo ORM (Object-Relational Mapping).
  - Transformación del modelo orientado a objetos al mundo relacional.
- Existen varios framework que hacen esto:
  - Hibernate.
  - TopLink.
  - OpenJPA.

# Capa de Acceso a Datos

- Hibernate
  - Es seguramente el framework ORM más utilizado.
  - Lo normal es utilizar una clase POJO con anotaciones.
  - También se puede utilizar una clase POJO y configurar XML.
  - Se asocia un POJO con una tabla de la base de datos.
  - Java Persistence API, incluida en EJB 3.0, recogió muchas ideas de Hibernate.
  - <http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=hibernate>.

# Capa de Acceso a Datos

- ¿Es necesario utilizar estos framework?
- Se pueden crear clases java POJO que hagan altas, bajas, modificaciones y consultas a la base de datos.
  - Es otra visión, para que veáis que se pueden utilizar diferentes tecnologías en una arquitectura en capas.
- La lógica de negocio utiliza estas clases para llamar a la base de datos de manera ordenada.
  - En vez de incluir las llamadas al acceso desde la lógica de negocio, llamamos a la capa de acceso de datos que se encarga de ello.

# Capa de Datos

- En esta capa entra claramente la base de datos.
- También pueden entrar los gestores documentales.
  - Debate: ¿Aquí o en la capa de lógica de negocio?
    - Es verdad que hacen otras funciones como flujos de aprobación.
    - Pero, ¿cuál es su función principal?
  - Productos java: Documentum, Alfresco.

# ¿Spring?

- Framework transversal.
- Se puede combinar con Struts o utilizar su propio MVC.
- Tiene muchos módulos, a parte del núcleo de Spring:
  - Capa de presentación: Spring Web Flow (proyecto dentro de Spring que se encarga de proveer una infraestructura para construir y ejecutar aplicaciones web ricas).
  - Capa de aplicación: Spring MVC (Si no quieres usar Struts).
    - Spring se integra también con struts.
  - Capa de negocio: Se aprovecha de
    - Inversión de Control .
    - Programación Orientada a Aspectos.
    - Gestión de Transacciones.
  - Capa de acceso a datos
    - Se integra con la mayoría de los ORM del mercado.

# ¿Spring?

- Inversión de Control.
  - Cambio en el flujo de ejecución.
  - Una clase X depende de otra clase Y
    - Se delega la creación de la clase Y por parte de X a un elemento externo.
    - La Clase X no se preocupa de su creación
      - Se gana en independencia entre las clases.
      - Es más fácil de probar.
- Programación Orientada a Aspectos.
  - Se utiliza para favorecer la modularidad.
  - Se utiliza para insertar funcionalidades adicionales a una clase sin modificar el código de dicha clase.
    - Se hace de manera declarativa.
    - Ejemplo: La escritura de logs en aplicaciones.

# Arquitectura SOA

- SOA = Service-Oriented Architecture.
- En esta arquitectura, más “filosofía”, se trata de que orientamos nuestros sistemas a resolver problemas, dando un servicio.
- Soluciones más comunes para implementar SOA:
  - Servicios Web.
  - ESB.

# Arquitectura SOA

- ¿Por qué SOA?
  - Podemos tener aplicaciones antiguas que queremos poder disponibles.
  - Queremos montar un nuevo servicio utilizando ya servicios que ya tenemos:
    - Se puede componer uno nuevo de manera sencilla utilizando esta filosofía.
  - Mantener una independencia de acceso a todos nuestros servicios
    - Más fácil para el desarrollador
- Una gran frase:
  - “Se van acercando los diferentes elementos que hay en la compañía”.

# Arquitectura SOA

- ¿Cuándo me conviene usar un ESB?
- ¿Cuándo me conviene usar servicios Web?
  - No todo lo que se hace con servicios Web se le puede considerar que aplique SOA.
- Quizá haya que hacerse una serie de preguntas.
  - ¿cuántos servicios voy a tener?
  - ¿tengo mucha orquestación?
  - ¿son los componentes muy diferentes?

# Arquitectura SOA

- Complejidad Alta: ESB.
  - Mucha orquestación.
  - Integración de sistemas heterogéneos.
  - Composición de diferentes posibilidades de servicios de manera rápida.
- Complejidad Baja: Servicios Web.
  - No se orquesta mucho ni componen los servicios.
  - Los servicios Web van a acceder a bases de datos convencionales.

# Arquitectura SOA

- Servicios Web
  - Intercambio de mensajes SOAP.
    - Son XML, complejos, pero XML.
  - Implementación java más conocida Axis2.
  - Con Axis2 podemos crear tanto el servidor como el cliente.
    - Es decir, si queremos poner un servicio nuestro como servicio Web, también utilizamos Axis2.
  - Módulo Rampart para mejorar la seguridad siguiendo estándar WS-Security.

# Arquitectura SOA

- ESB
  - Enrutamiento y redireccionamiento de mensajes.
  - Estilo de comunicación síncrono y asíncrono.
    - En servicios Web también podemos componer estos estilos de comunicación.
  - Multiplicidad de tipos de transporte y protocolos de enlace.
  - Transformación de contenido y traducción de mensajes.
    - Introduce seguridad en las comunicaciones hacia el ESB.
  - Orquestación y coreografía de procesos de negocio.
    - Crear nuevos servicios a partir de otros existentes.
  - Presencia de adaptadores a múltiples plataformas.
    - Posibilidad de exponer servicios de un HOST o un ERP.
  - Gestión y monitorización.
- Ejemplos
  - Comerciales: Oracle Enterprise Service Bus, ActiveMatrix™ Service Bus
  - Open Source: Jboss ESB, Apache ServiceMix, Open ESB

GRACIAS POR SU ATENCIÓN

