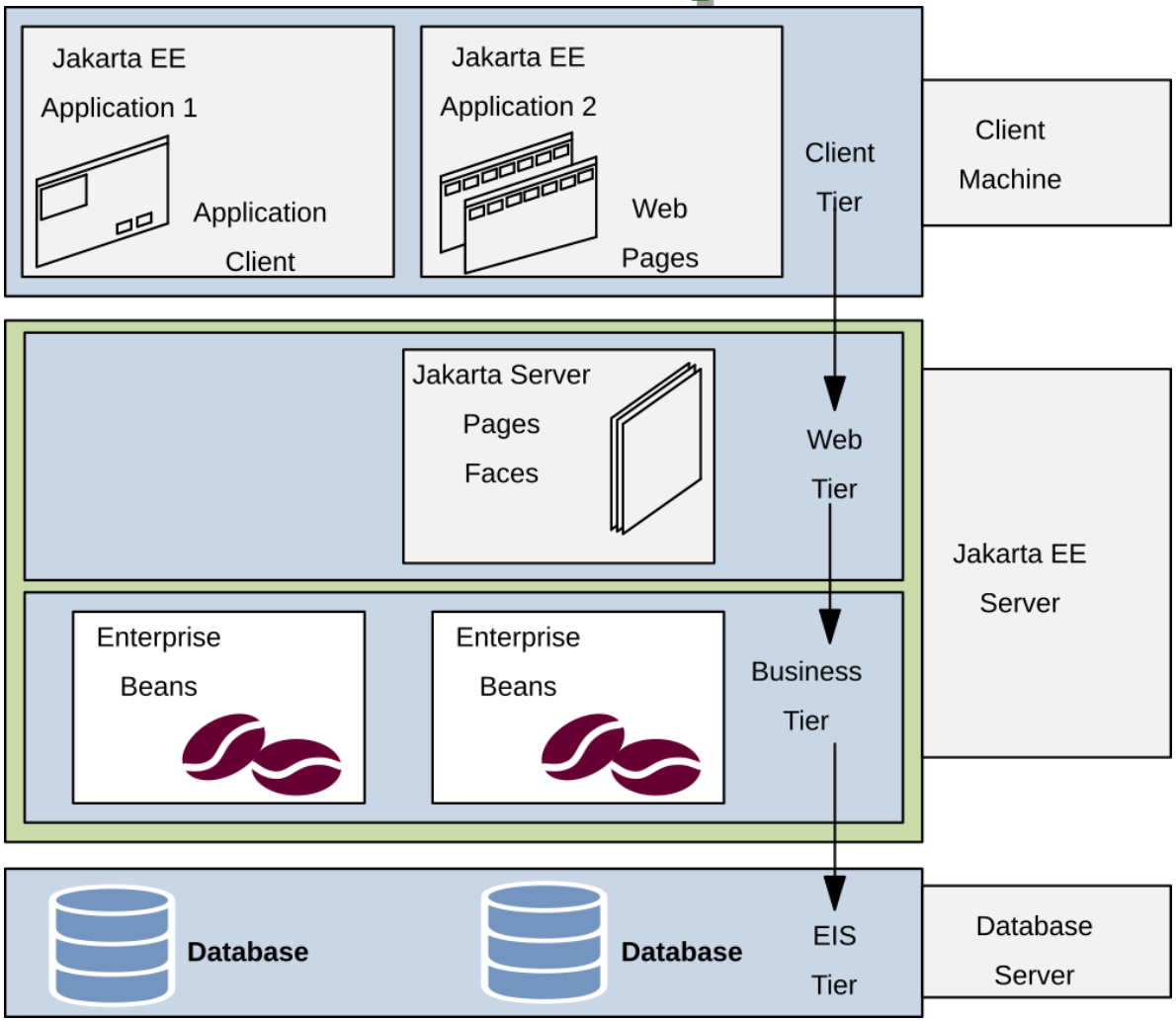
Sistemas Informáticos II
Práctica 1 – Segunda parte
EJB

Prácticas 2023/24

Objetivos y entorno

- Objetivos
  - Conocer y experimentar con la tecnología de ***Enterprise Java Beans (EJBs)***
  - Mensajería **JMS**, gestores de colas
  - **Message-Driven Beans**
- Material entregado
  - **P1-ejb-base.tgz**: Proyecto de ejemplo con la estructura para EJB Session Stateless (con interfaz local)
  - **P1-ejb-transaccional-base.tgz**: Modificaciones para comprobar la transaccionalidad de un EJB
  - **P1-jms-base.tgz**: Proyecto de ejemplo con cliente JMS y servidor MDB para envío de mensajes
- Entorno
  - Glassfish v7
  - Línea de comandos

Aplicaciones web distribuidas multicapa



Fuente: Jakarta Tutorial

Capa Negocio

- Primera parte: **EJB** sesión sin estado
  - **Objetivo:** convertir la clase VotoDAOWS realizada en la P1a en un EJB stateless.
  - **Interfaz remota**: cliente y servidor residirán en distintos servidores.

P1-ejb

- Segunda parte: Transaccionalidad **EJB gestionada por el contenedor**
  - **Objetivo:** comprobar la correcta gestión de transaccionalidad realizada por parte del contenedor del EJB.
  - Modificar *registraVoto* para comprobar su funcionamiento transaccional → campo votos restantes en el censo de la base de datos.

P1-ejb-transaccional

- Tercera parte: **MDBs** y colas de mensajes JMS
  - **Objetivo**: Familiarizarse con el uso de mensajería JMS, gestores de colas, y Message-Driven Beans (MDB).
  - Incluir en la aplicación la posibilidad de que un agente externo realice la cancelación de un voto mediante el envío de un mensaje.

P1-jms

JavaBeans

- **Modelo de programación modular por componentes (Beans)**

- **Conjunto de estándares**
  - Todos los atributos son privados
  - Métodos get/set
  - Constructor público sin argumentos
  - Serializable

Enterprise JavaBeans (EJBs)

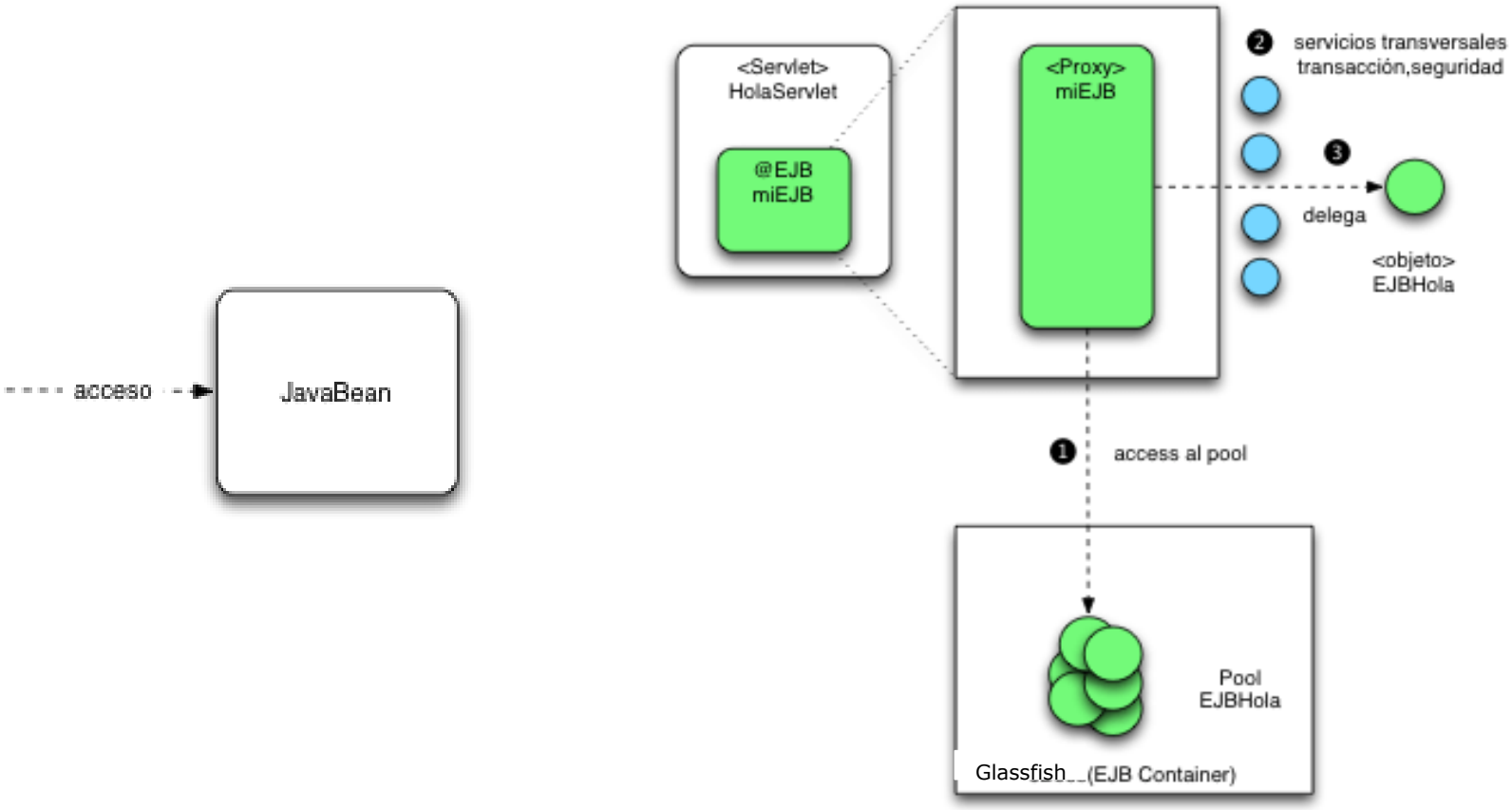
- **Extensión de los estándares JavaBeans para arquitecturas remotas**
- **Permiten la abstracción de detalles de bajo nivel típicos de la programación remota**
  - Manejo de la transaccionalidad
  - Seguridad: comprobación de permisos de acceso a los métodos del bean.
  - Concurrencia: llamada simultánea a un mismo bean desde múltiples clientes.
- **Sistema de EJBs**
  - Componentes: Clases Java (JavaBeans) que contienen la lógica de negocio.
  - Contenedor: Controla el ciclo de vida y los aspectos de bajo nivel del EJB
  - Interfaz/Interfaces: Permiten el acceso de clientes a los métodos de los componentes EJB

EJB vs JavaBeans

- **JavaBean**: objetos POJO (Plain Old Java Object)
  - Modelo más simple y adecuado para componentes sencillos.
  - No se pueden usar en entornos objetos distribuidos ya que no soportan RMI
  - No soportan transaccionalidad

- **EJB**: presentan mayor complejidad, pero a cambio
  - Soportan invocación remota: mayor escalabilidad
  - Proporcionan transaccionalidad abstrayendo al programador de su gestión

EJB vs JavaBeans



Fuente: https://www.arquitecturajava.com/ejb-3-1-ii-despliegue-de-ejb/

EJB vs WS

- **EJB**: Emplea RMI-IIOP
  - Transferencias más ligeras (protocolo binario)
  - Protocolo IIOP actualmente limita la interconectividad a través de firewalls en comparación con HTTP
  - Limitado a Java
  - **Permite interfaz de invocación local más eficiente**
  - Gestión automática de la transaccionalidad

- **WS:** SOAP emplea XML sobre HTTP
  - Transferencias más pesadas (XML vs binario)
  - Mayor interconectividad
  - Soportado por más lenguajes de programación
  - No permite invocación local – la permite, pero es igual de ineficiente
  - Gestión manual de la transaccionalidad

Transaccionalidad

- **Propiedades ACID**
  - **A**tomic, **C**onsistent, **I**solated, **D**urable
- **Gestionada por la aplicación (*Bean Managed*):**
  - La aplicación puede decidir abortar la transacción
  - La "marcha atrás" se hará a mano, deshaciendo todas las operaciones (ROLLBACK sobre la base de datos, modificaciones manuales sobre el modelo)
- **Gestionada por el contenedor (*Container Managed*):**
  - La aplicación lanza **EJBException**
  - Todos los cambios son deshechos automáticamente

Session EJB vs MDBs

- **Session EJB**: Su ejecución viene desencadenada por la invocación de un método (local o remoto) por otra aplicación. Sólo almacena datos locales (de sesión)
  - Stateful (con estado)
  - Stateless (sin estado)
  - Singleton (1 única instancia por aplicación)

- **Message Driven Beans (MDBs)**
  - Reciben mensajes JMS de forma asíncrona
  - Su ejecución viene desencadenada por la llegada de un mensaje (método onMessage())
  - Únicamente accesibles por conexiones JMS. No ofrecen interfaces al cliente.

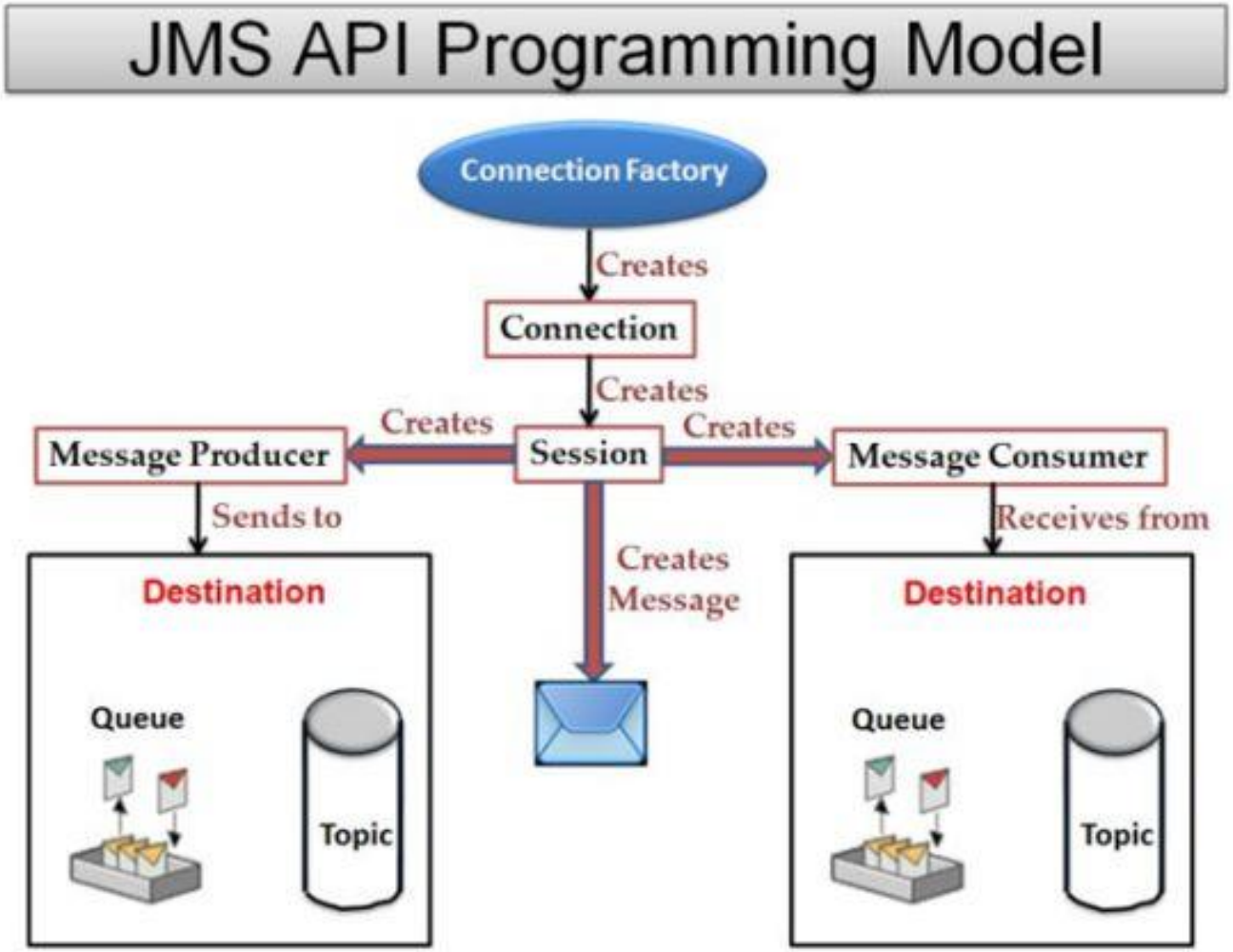
Java Messaging Service (JMS)

- **Transmisión de mensajes *asíncrona, desacoplada, escalable y segura* para aplicaciones distribuidas**

- **Destinos de conexión:**
  - Contenedores de mensajes a los que se conecta la aplicación para enviar/obtener mensajes
  - Dos tipos:
    - ***Message Queues* (MQs):** Un lector, varios escritores
    - ***Message Topics* (MTs):** Varios suscriptores, varios escritores

- **Factorías de conexión**: Permiten la instanciación de destinos de conexión de manera ordenada

Java Messaging Service (JMS)



Fuente: http://theopentutorials.com/tutorials/java-ee/ejb3/mdb/jms-api-programming-model/

Entrega

- La **entrega de los resultados de esta práctica se regirá por las normas expuestas durante la presentación de la asignatura**.
- Nomenclatura del fichero a entregar
  SI2P1B_<grupo>_<pareja>.zip
  (ejemplo: SI2P1B_2311_1.zip)

- Contenido del fichero:
  - Informe técnico siguiendo la plantilla publicada en la página del laboratorio con las respuestas a todas las preguntas
  - **P1-ejb** con las modificaciones que hayan sido necesarias para el EJB.
  - **P1-ejb-transaccional** con las modificaciones que hayan sido necesarias para comprobar la transaccionalidad de los EJB
  - **P1-jms** con las modificaciones que hayan sido necesarias para el MDB.
- Entrega: la fecha de entrega se encuentra en la 'Planificación de Prácticas' en moodle