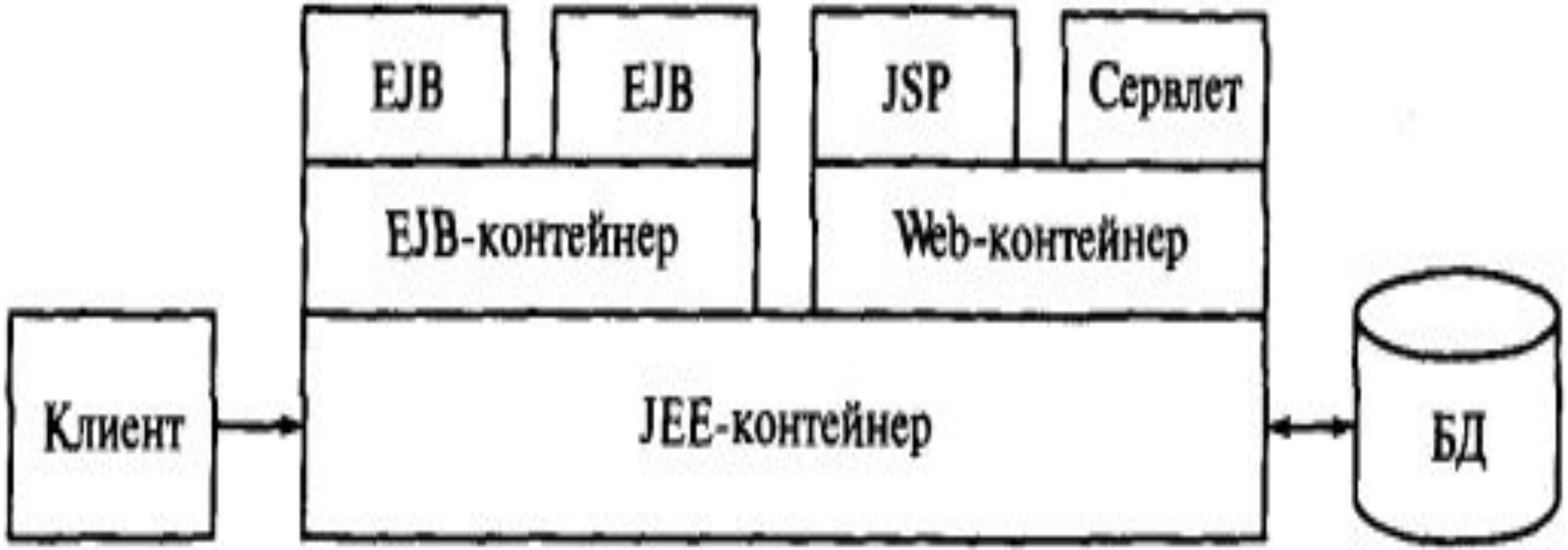
Технология Enterprise Java Beans

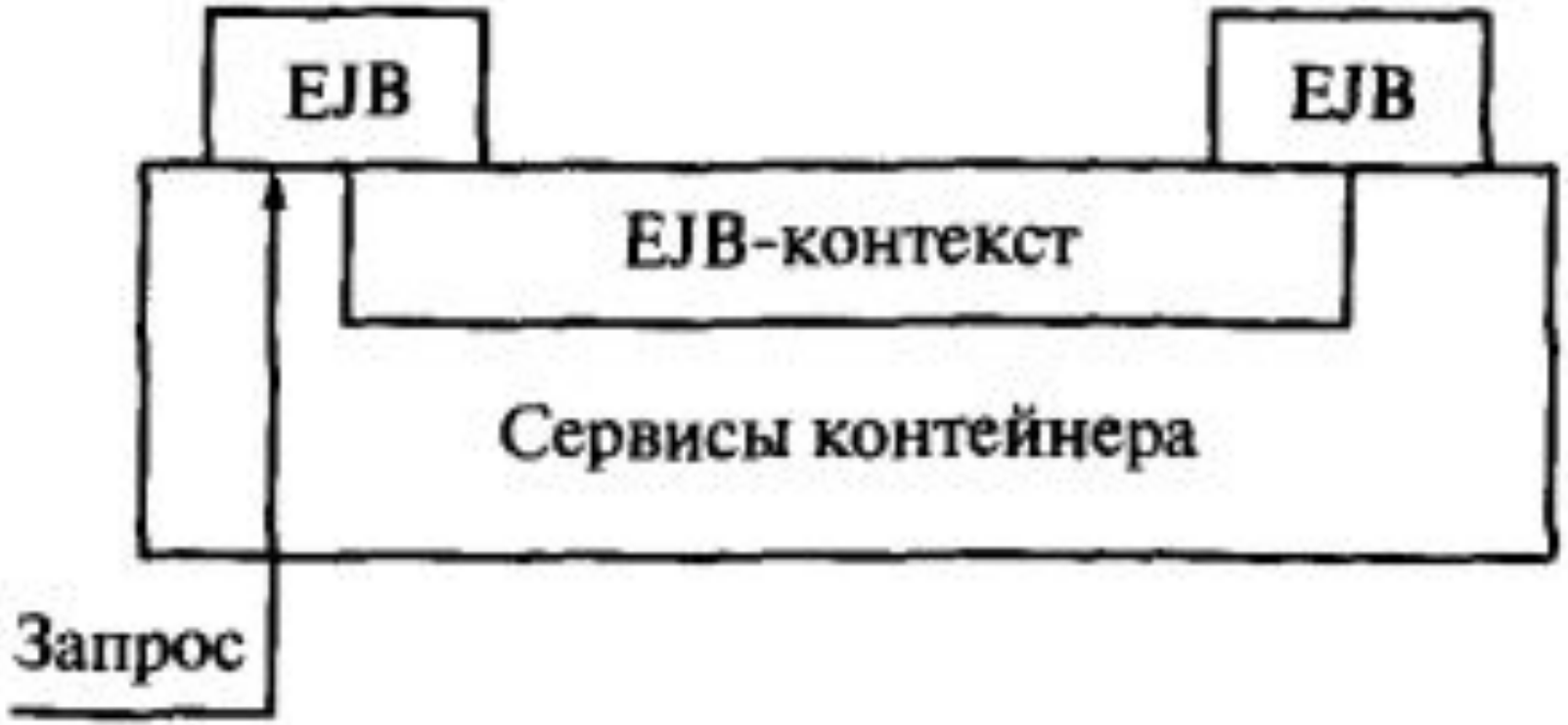
- Технология Enterprise Java Beans (EJB) представляет собой высокоуровневый подход к построению распределенных приложений масштаба предприятия. EJB — это модель серверных компонентов (server component model) для Java.
- Основная идея технологии Enterprise JavaBeans состоит в создании такой инфраструктуры для компонентов, чтобы они могли легко добавляться (plug in) и удаляться из серверов без перекомпиляции кодов приложения, тем самым расширяя или ограничивая функциональность сервера.

Архитектура Enterprise Java Beans



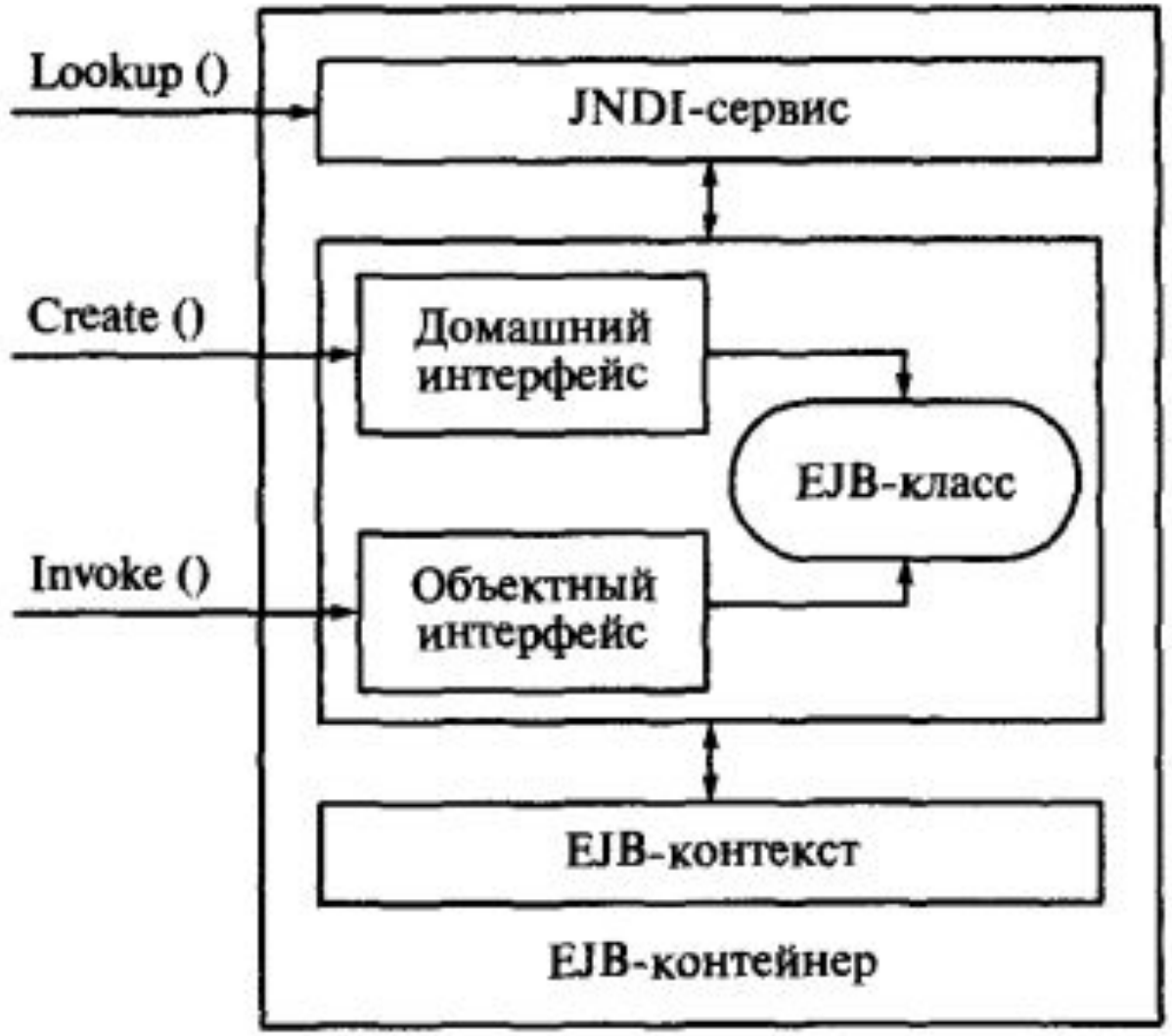
- JEE совместимый сервер приложений обеспечивает следующие основные сервисы: HTTP-сервис, сервисы безопасности, Java Naming and Directory Interface (JND) — сервисы.

Структура EJB-контейнера



- EJB-компонент представляет собой программный компонент, который может быть повторно использован без перекомпиляции в разных приложениях. Для использования EJB-компонента достаточно поместить его в соответствующий каталог и выполнить настройки конфигурационного файла, который называют Deployment Descriptor (DD). DD представляет собой XML-файл.

Структура EJB-компонента



Компонентная модель EJB 2.X

- КаждыйEJB-компонент имеет объектный интерфейс (EJB-Object), через который клиент может обратиться к данному компоненту. При этом клиенту могут быть неизвестны подробности, касающиеся, в частности, местонахождения компонента. Этот интерфейс называют удаленным (remote interface). Конкретный экземпляр EJB-компонента управляется контейнером через домашний интерфейс (home interface). Каждый EJB-компонент должен поддерживать как удаленный, так и домашний интерфейс.
- Конфигурирование EJB-компонента осуществляется посредством редактирования конфигурационного файла.

- Взаимодействие осуществляется следующим образом. Клиент обращается к компоненту по имени (Lookup), которое используется для получения объектной ссылки (Object Reference, OR) через JNDI-сервис. При обращении к контейнеру (Create) он создает экземпляр EJB-компонента и передает ему параметры вызова. Очевидно, что для того чтобы можно было получать ссылки, компоненты должны быть предварительно зарегистрированы с помощью JNDI-сервиса. Затем вызывается требуемый метод (Invoke).
- EJB-компонент представляет собой программный компонент, который может быть повторно использован без перекомпиляции в разных приложениях.

- Для использования EJB-компонента достаточно поместить его в соответствующий каталог и выполнить настройки конфигурационного файла, который называют Deployment Descriptor (DD). DD представляет собой XML-файл.

- Компонентная модель EJB определяет три основных типа компонентов:
- сеансовые компоненты (Session Beans);
- компоненты-сущности (Entity Beans);
- компоненты, ориентированные на сообщения (Message Driven Beans).
- В свою очередь, сеансовые компоненты делятся на две группы: без состояния (Stateless Session Beans) и с состоянием (Stateful Session Beans).

- *Сеансовые компоненты без состояния* предназначены для выполнения бизнес-операций, когда не требуется сохранять внутреннее состояние компонента. К таким операциям можно отнести операции, связанные с поиском слов в словаре, архивирование файлов, калькуляторы. Компоненты данного типа можно реализовать в виде Web-сервисов.
- *Сеансовые компоненты с состоянием* помнят свое состояние только в пределах сеанса. Время жизни таких компонентов соответствует одному сеансу, продолжительность которого может составлять от нескольких секунд до нескольких часов и даже дней.
- Типичный пример использования сеансовых компонентов с состоянием — интернет-магазин. EJB-компонент может описывать, например содержимое корзины покупателя. Тогда сеансом будет считаться время с момента регистрации пользователя на сайте магазина до момента закрытия пользователем браузера или перехода к другому сайту.

- Особенности компонентной модели EJB 3.0. Одним из побудительных мотивов создания EJB-технологии были сложности CORBA. Однако эволюция технологии EJB привела к тому, что эта технология стала казаться разработчикам чрезмерно сложной, в частности:
- тяжеловесная модель программирования, требующая работы с несколькими интерфейсами;
- необходимость непосредственно взаимодействовать с Java Naming Directory Interface (JNDI);
- необходимость работать с непомерно сложным XML DD.
- Эти недостатки устранены в EJB 3.0, где используется только один бизнес-интерфейс вместо интерфейсов home и remote, минимизируется использование DD за счет использования аннотаций. Кроме того, используется новый механизм для сохранения состояния для компонентов-сущностей (Entity Beans). Одна из новых черт EJB 3.0 — использование Java Persistence API (JPA) для реализации сохранения состояния.