**Принципы Ioc и CDI**

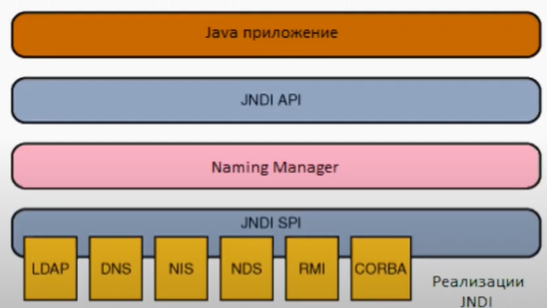
IoC

* Жизненным циклом управляет контейнер
* За взаимодействие между компонентами также отвечает контейнер

Изображение выглядит как текст

Автоматически созданное описаниеСDI – позволяет снизить или убрать зависимость от контейнера. Когда мы пишем компонент, мы не зависим от API которые ему предоставляет контейнер.

* Не требует реализации интерфейсов
* Не нужны прямые вызовы API
* Реализуется через аннотации



**JNDI** - это набор Java API, организованный в виде службы каталогов, который позволяет Java-клиентам открывать и просматривать данные и объекты по их именам. Очень похожа на jdbc

Службы каталогов – иерархическая бд.

Два варианта использования JNDI:

• CDI (аннотации) - работает только в managed компонентах.

• Прямой вызов API - работает везде.

Принцип Location Transparency - Благодаря CDI не важно, где физически расположен вызываемый компонент - за его вызов отвечает контейнер. Также это можно понимать как паттерн – мы пишем компоненты так, чтобы нам было не важно где они находятся физически.

RMI –API, который позволет вызывать методы удаленных объектов

* В общем случае, объекты передаются по значению (копии).
* Передаваемые объекты должны быть Serializable.

**Enterprise Java Bean**

EJB - технология разработки серверных компонентов, реализующих бизнес-логику.

Особенности EJB:

* Возможность локального и удалённого доступа.
* Возможность доступа через JNDI или Dependency Injection.
* Поддержка распределённых транзакций (с помощью JTA).
* Поддержка событий.
* Жизненным циклом управляет EJB-контейнер (в составе сервера приложений).

Виды EJB

Message-Driven Bean

Session Bean

* Предназначены для синхронной обработки вызовов.
* Вызываются посредством обращения через API.
* Могут вызываться локально или удалённо.
* Могут быть endpoint'ами для веб-сервисов.
* CDI - аннотация @EJB.
* Не обладают свойством персистентности.
* Можно формировать пулы бинов (за исключением @Singleton).

|  |  |  |
| --- | --- | --- |
| Stateless | Stateful | Singleton |
| • Не сохраняют состояние между  последовательными обращениями.  • Нет привязки к конкретному клиенту.  • Хорошо масштабируются.  • Объявление - аннотация @Stateless. | • «Привязываются» к конкретному клиенту.  • Можно сохранять контекст в полях класса.  • Масштабируются хуже, чем @Stateless.  • Объявление - аннотация @Stateful. | • Контейнер гарантирует существование строго  одного экземпляра такого бина.  • Объявление - аннотация @Singleton. |

Изображение выглядит как текст

Автоматически созданное описаниеРазработка session bean

1)Создание бизнес-интерфейса компонента (Business Interface).

* Бизнес-интерфейс содержит заголовки всех методов, реализующих бизнес-логику компонента.
* Может быть локальным (@Local) и удалённым (@Remote).

Изображение выглядит как текст

Автоматически созданное описание2)Создание класса компонента, реализующего бизнес-интерфейс.

* Компонент должен реализовывать все методы бизнес- интерфейса.
* Может содержать методы, реагирующие на события жизненного цикла компонента.

3)Конфигурация компонента с помощью аннотаций и/или дескриптора развёртывания.

Модель EJB

* EJB-компоненты инкапсулируются контейнером.
* Контейнер предоставляет клиентам прокси- объекты (ргoxy objects) для доступа к EJB- компонентам.
* Прокси-объекты реализуют бизнес-интерфейсы (Business Interfaces).
* Клиенты вызывают методы бизнес- интерфейсов.

