Сессионные компоненты

Виды сессионных компонентов

- Stateless Session Bean (SSB)
  - Сессионный компонент без состояния
- Stateful Session Bean (SFSB)
  - Сессионный компонент **с** состоянием
- Singleton — *EJB 3.1*
  - Синглтон
- Сильно отличаются по жизненному циклу с точки зрения контейнера
- Похожи по жизненному циклу с точки зрения клиента
- Похожи по требованиям, предъявляемым к разработчику компонента

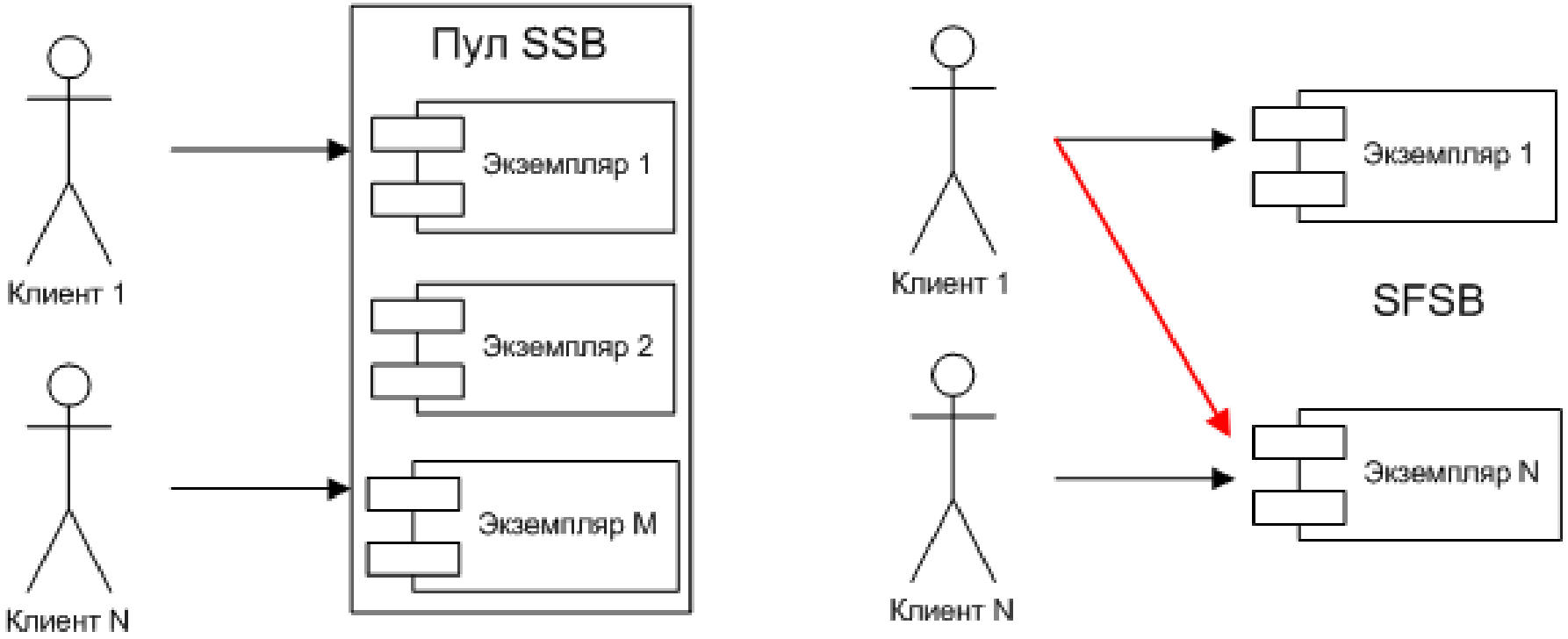
Сессионные компоненты без состояния

- Предназначены для
  - реализации функционального API приложения
  - моделирования процессов или бизнес-операций
- «Фасад» приложения
  - Скрывает от клиентов внутренние слои middleware-приложения
- Между вызовами методов компонента внутри него не должна сохраняться никакая информация о состоянии
  - Для реализации сохраняемого состояния используются другие средства (SFSB, БД, файлы, JNDI-каталог)

Сессионные компоненты с состоянием

- Используются для хранения данных в рамках сеанса работы пользователя с приложением
  - Данные сеанса реализуются в виде полей класса компонента
    - Их не требуется сохранять в постоянном хранилище, так как они актуальны и используются исключительно в рамках сеанса
- Вызовы, сделанные клиентом на одном и том же экземпляре компонента, обязательно будут обработаны одним и тем же экземпляром класса SFSB-компонента

Сессионные компоненты без состояния и с состоянием



Преимущества SFSB перед веб-сессией

- Для SFSB контейнер поддерживает сервис управления памятью
  - Пассивный компонент: при повышенной нагрузке экземпляр SFSB-компонента может удаляться из памяти, его состояние сохраняется в сериализованной форме в постоянном хранилище
    - Тип постоянного хранилища и политика управления памятью не закреплены в спецификации
  - При обращении к пассивному компоненту со стороны клиента, он автоматически восстанавливается в оперативной памяти — становится активным
- SFSB могут быть использованы как веб-, так и GUI-клиентами, совместимыми с платформами Java или CORBA
- Транзакции и безопасность, управляемые контейнером

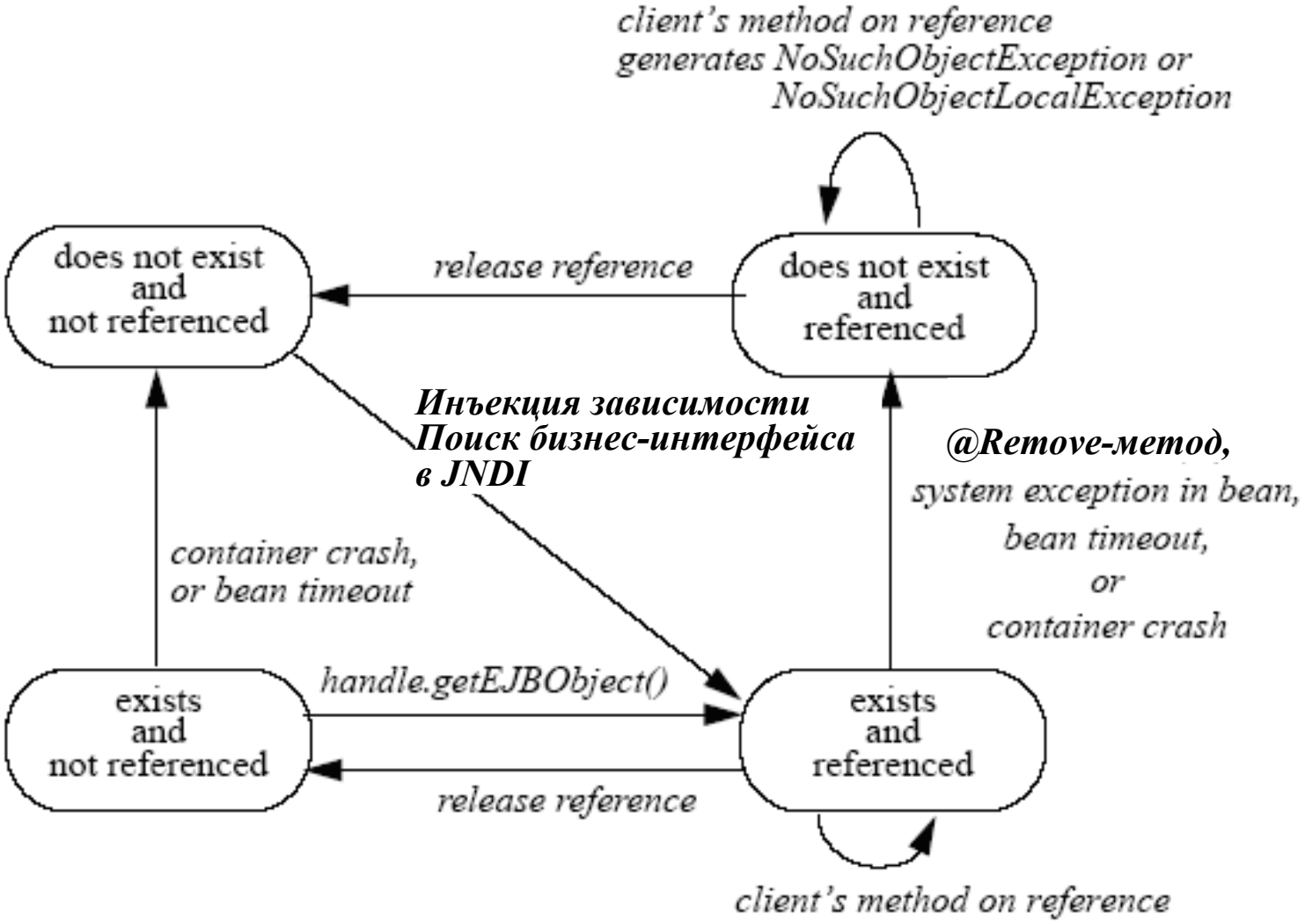
Синглтоны

- Реализуют шаблон проектирования Singleton
  - В любой момент времени в приложении существует не более одного объекта заданного класса
  - Например, для организации кэша
- Аналогичны контексту приложения в веб-приложениях, но имеют преимущества:
  - Могут быть использованы как веб-, так и GUI-клиентами
  - Транзакции и безопасность, управляемые контейнером
  - Container Managed Concurrency — автоматическая синхронизация доступа к синглтону

Хранение данных в веб-приложении и в сессионных компонентах

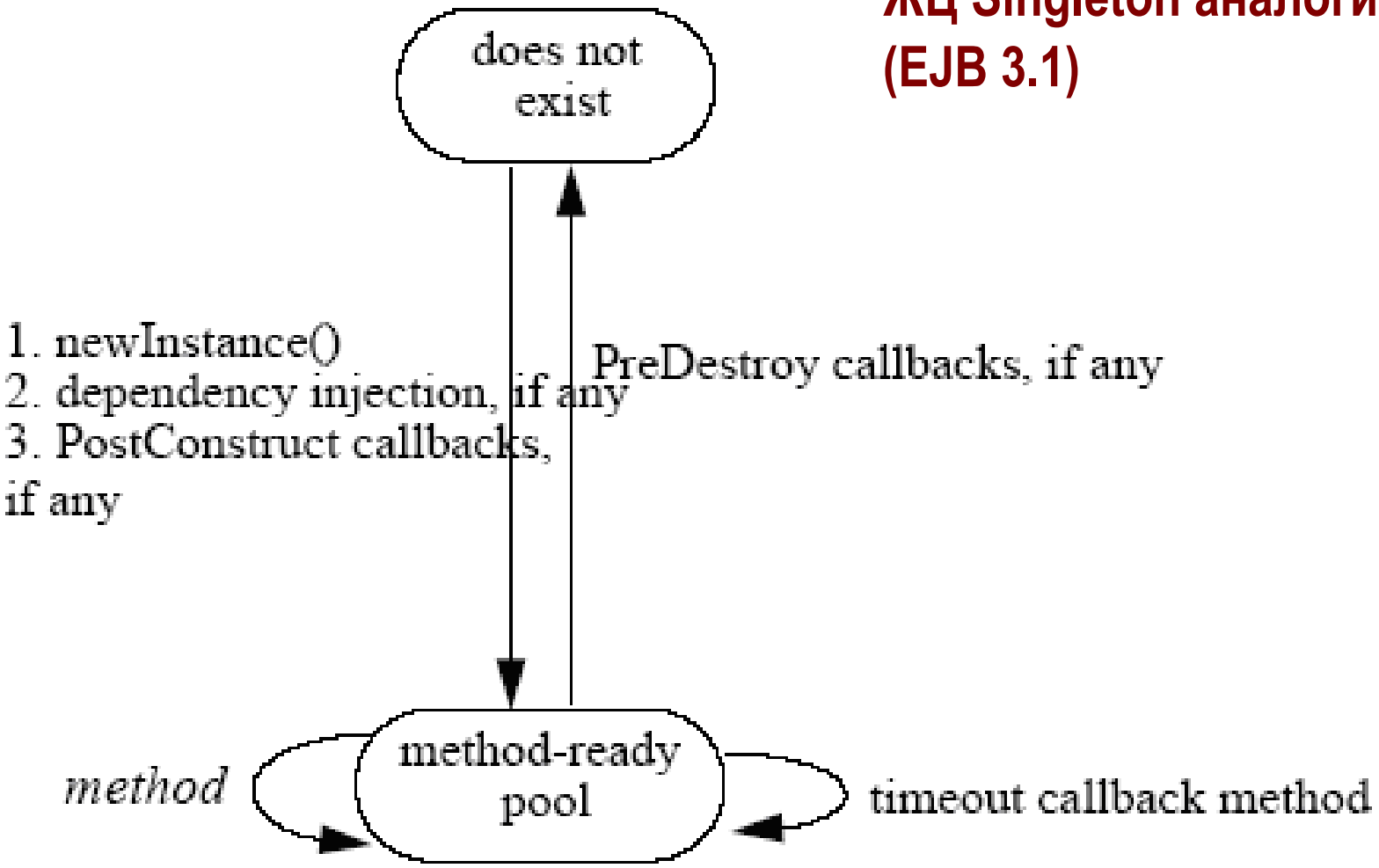
Контекст хранения данных в веб-приложении	Вид сессионного компонента
Запрос	**Без состояния**
Сессия	**С состоянием**
Приложение	**Синглтон**

ЖЦ сессионного компонента с точки зрения клиента

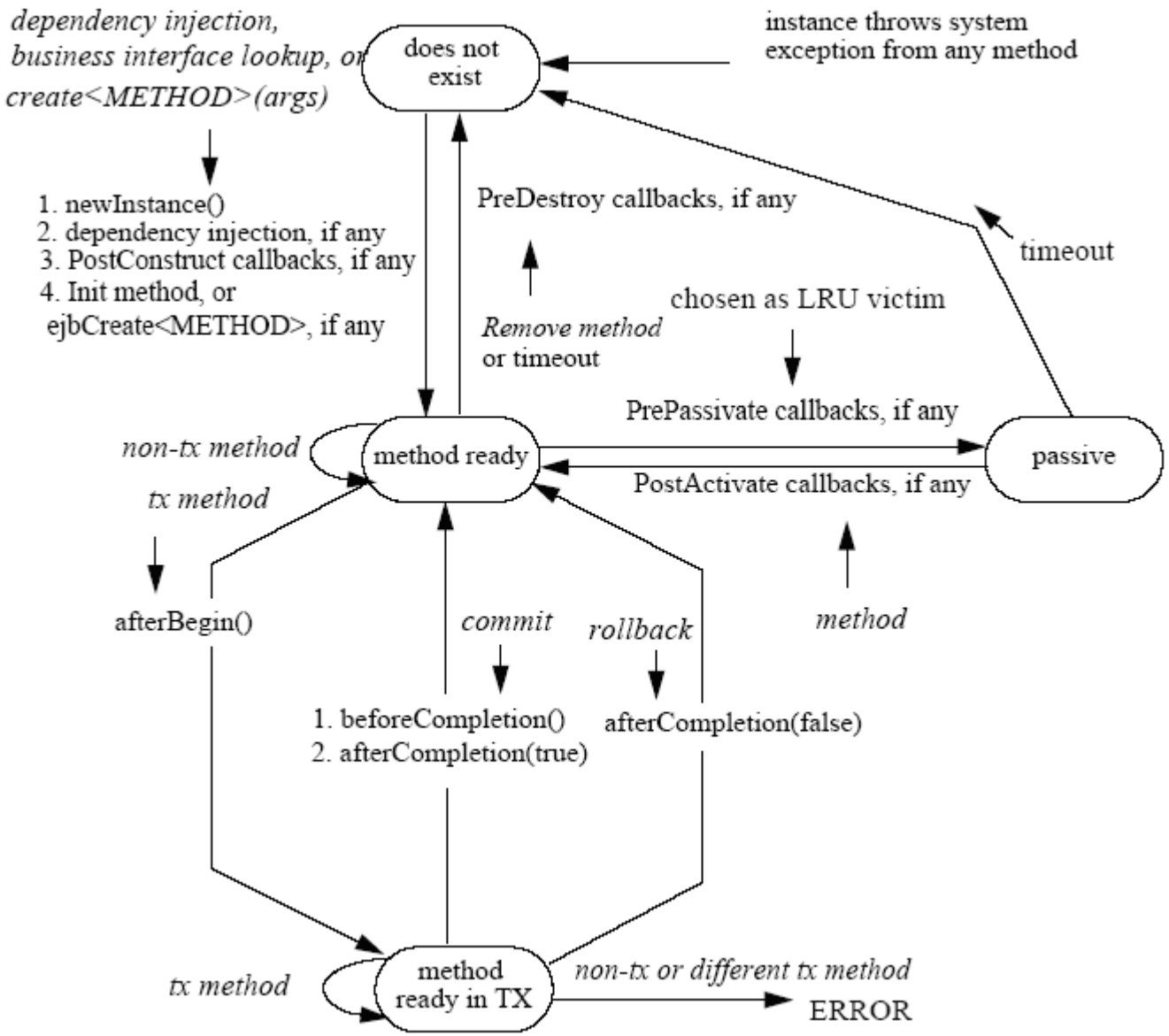


ЖЦ SSB с точки зрения контейнера

ЖЦ Singleton аналогичен (EJB 3.1)



ЖЦ SFSB с точки зрения контейнера



Требования к полям SFSB

- По завершении обработчика @PrePassivate любое не-transient поле класса SFSB должно содержать:
  1) сериализуемый объект, либо значение простого типа;
  2) **null**;
  3) ссылка на объект, управляемый контейнером
  4) ссылка на объект, становящийся сериализуемым при замене содержащихся в нем объектов на сериализуемые (например, коллекция ссылок на компонентные интерфейсы)
- То же относится к перехватчикам, связанным с SFSB
- В @PrePassivate - закрыть все активные соединения с менеджерами ресурсов и присвоить этим полям null
- В @PostActivate - восстановить соединения

Реализация SB - требования к бизнес-интерфейсу

1) Тип представления, обеспечивающего данный интерфейс, можно указать с помощью аннотации @Remote (удаленное представление) или @Local (локальное представление)
2) Интерфейс НЕ должен расширять интерфейс javax.ejb.EJBObject или javax.ejb.EJBLocalObject
3) Если бизнес-интерфейс является удаленным, то он НЕ обязан расширять java.rmi.Remote, и его методы в этом случае НЕ должны выбрасывать исключение java.rmi.RemoteException

Реализация SB - требования к классу компонента

1) Класс должен быть объявлен как public и НЕ может быть final или abstract - для генерации View
2) Публичный конструктор без параметров - для создания экземпляра класса компонента
3) Аннотации для типа компонента: @Stateful, @Stateless, @Singleton
4) Класс компонента должен реализовать бизнес-интерфейс(ы) компонента или его (их) методы
5) Класс компонента может прямо или косвенно реализовать интерфейс javax.ejb.SessionBean
6) Класс компонента может содержать ejbCreate-метод(ы)
7) Класс компонента может иметь родительский класс, который НЕ может быть классом сессионного компонента (отменено в EJB 3.1)
8) Кроме этого класс компонента может содержать любые вспомогательные методы

Реализация SB - требования к бизнес-методам

a) название метода может быть произвольным, но не должно начинаться с ejb;

b) метод должен быть public и не может быть final или static - для генерации View;

c) параметры и возвращаемое значение должны быть сериализуемыми, если метод соответствует бизнес-методу удаленного бизнес-интерфейса;

d) метод может выбрасывать произвольные исключения, определенные в приложении (не системные)

Новое в EJB 3.1

- Синглтоны:
  - Порядок инициализации (@Startup, @DependsOn)
  - Параллелизм, управляемый контейнером
    - Блокировки @Lock(READ) и @Lock(WRITE)
- Асинхронные вызовы сессионных компонентов
- EJB Lite 3.1 - минимальное подмножество EJB API для разработки переносимой транзакционной бизнес-логики
- Встраиваемый EJB-контейнер