1. EJB

ИТМО ВТ

Enterprise Java Beans

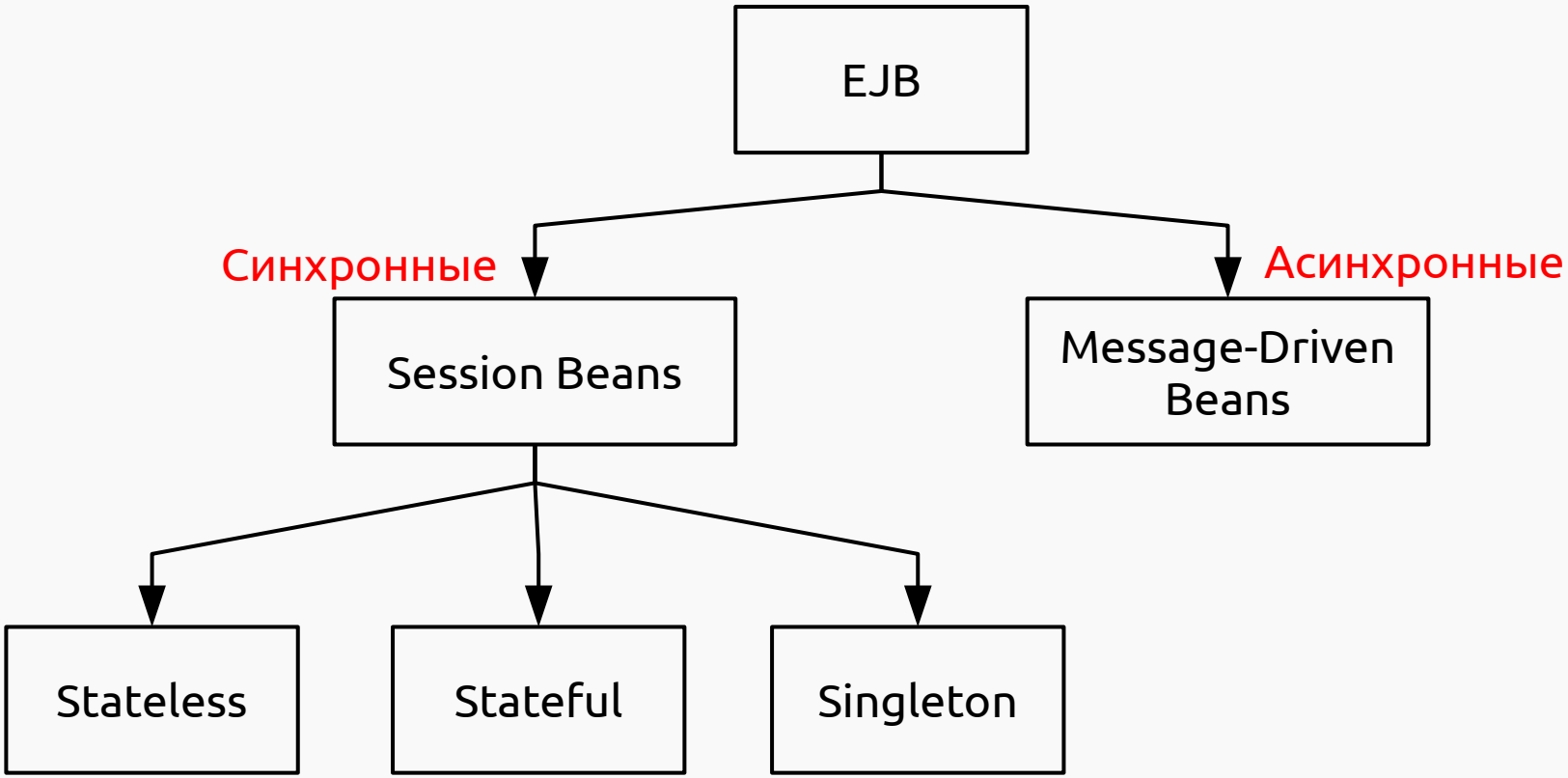
EJB — технология разработки серверных компонентов, реализующих бизнес-логику.

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).

ИТМО ВТ

Виды EJB



ИТМО ВТ

Session Beans

- Реиспользуемый компонент, с помощью которого реализуется бизнес-логика приложения.
- Управляется **EJB-контейнером**:
  - Поддержка транзакций
  - Перехватчики
  - Сервисы управления расписанием (Timer services)
  - Управление безопасностью

ИТМО ВТ

Пример Session Bean (без интерфейса)

```
package converter.ejb;

import java.math.BigDecimal;
import javax.ejb.*;

@Stateless
public class MyConverterBean {
    private BigDecimal yenRate = new BigDecimal("83.0602");
    private BigDecimal euroRate = new BigDecimal("0.0093016");

    public BigDecimal dollarToYen(BigDecimal dollars) {
        BigDecimal result = dollars.multiply(yenRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }

    public BigDecimal yenToEuro(BigDecimal yen) {
        BigDecimal result = yen.multiply(euroRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }
}
```

Пример Session Bean (Remote)

```
package converter.ejb;
//same imports here

public interface MyConverter {
    public BigDecimal dollarToYen(BigDecimal dollars);
    public BigDecimal yenToEuro(BigDecimal yen);
}

@Stateless
@Remote(MyConverter.class)
public class MyConverterBean implements MyConverter {
    private BigDecimal yenRate = new BigDecimal("83.0602");
    private BigDecimal euroRate = new BigDecimal("0.0093016");

    public BigDecimal dollarToYen(BigDecimal dollars) {
        // same implementation
    }

    public BigDecimal yenToEuro(BigDecimal yen) {
        // same implementation
    }
}
```

Stateless Session Bean (1)

- EJB-бин, не хранит состояние между последовательными запросами к нему:

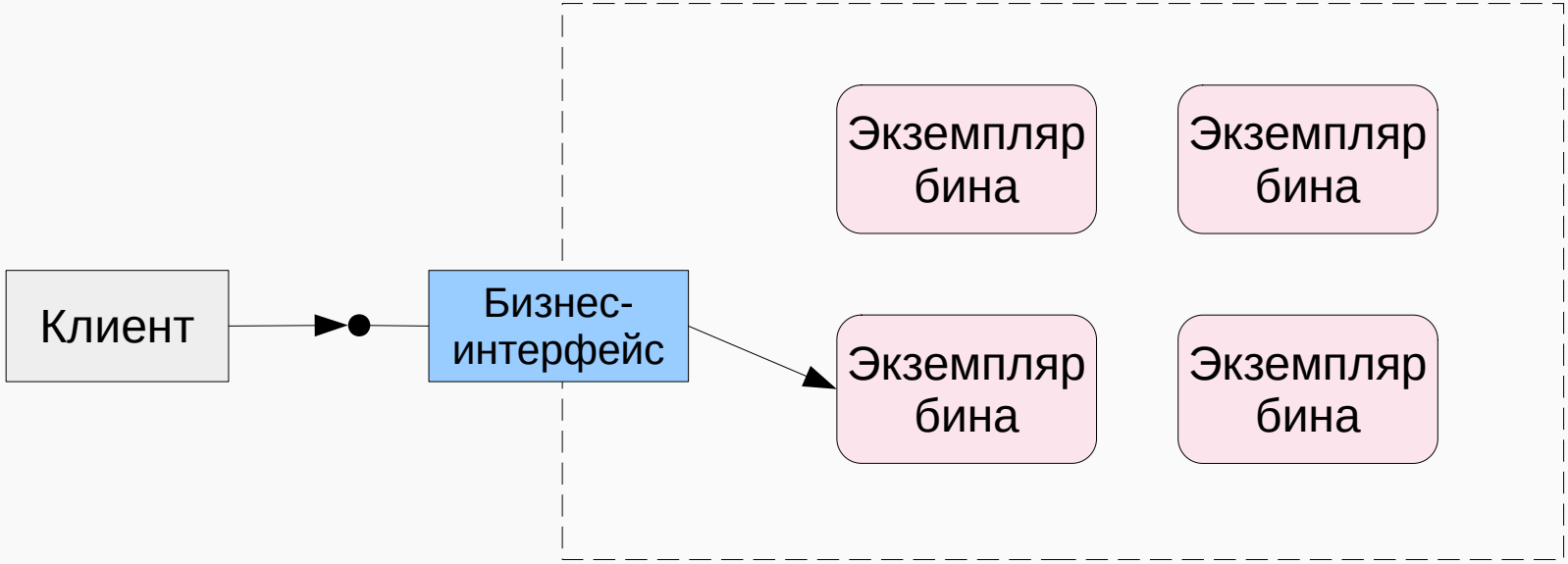
```
@Stateless
public class MyConverterBean {
    private BigDecimal yenRate = new
BigDecimal("83.0602");
    private BigDecimal euroRate = new
BigDecimal("0.0093016");

    public BigDecimal dollarToYen(BigDecimal dollars) {
        BigDecimal result = dollars.multiply(yenRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }

    public BigDecimal yenToEuro(BigDecimal yen) {
        BigDecimal result = yen.multiply(euroRate);
        return result.setScale(2, BigDecimal.ROUND_UP);
    }
}
```

Stateless Session Bean (2)

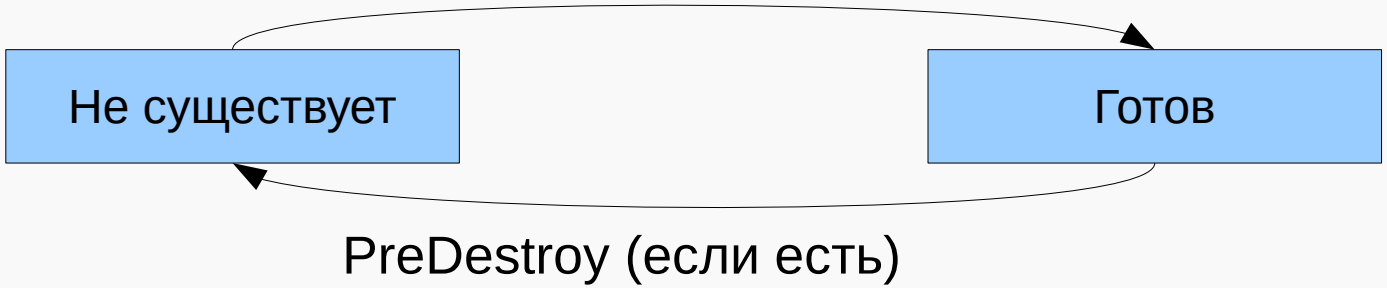
- На сервере — пул таких бинов:



ИТМО ВТ

ЖЦ Stateless Session Bean

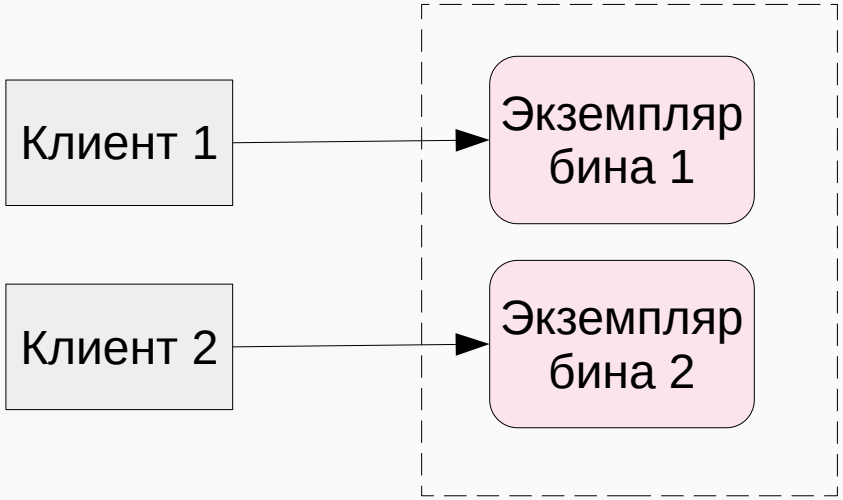
1. Dependency Injection (если есть)
2. PostConstruct (если есть)



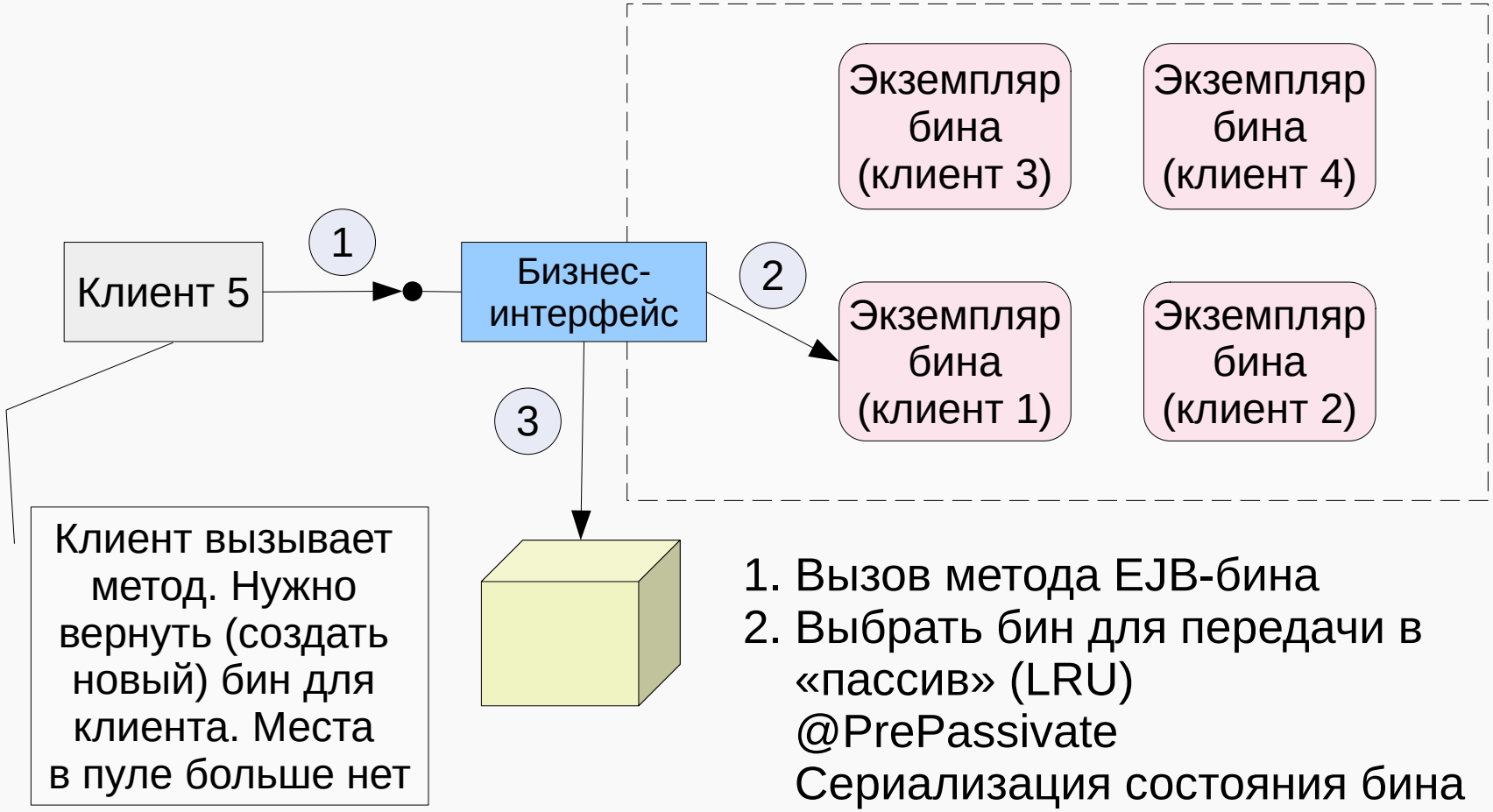
ИТМО ВТ

Stateful Session Bean (1)

- EJB-бин, хранит состояние между последовательными запросами к нему.
- У каждого клиента — свой Stateful Bean.

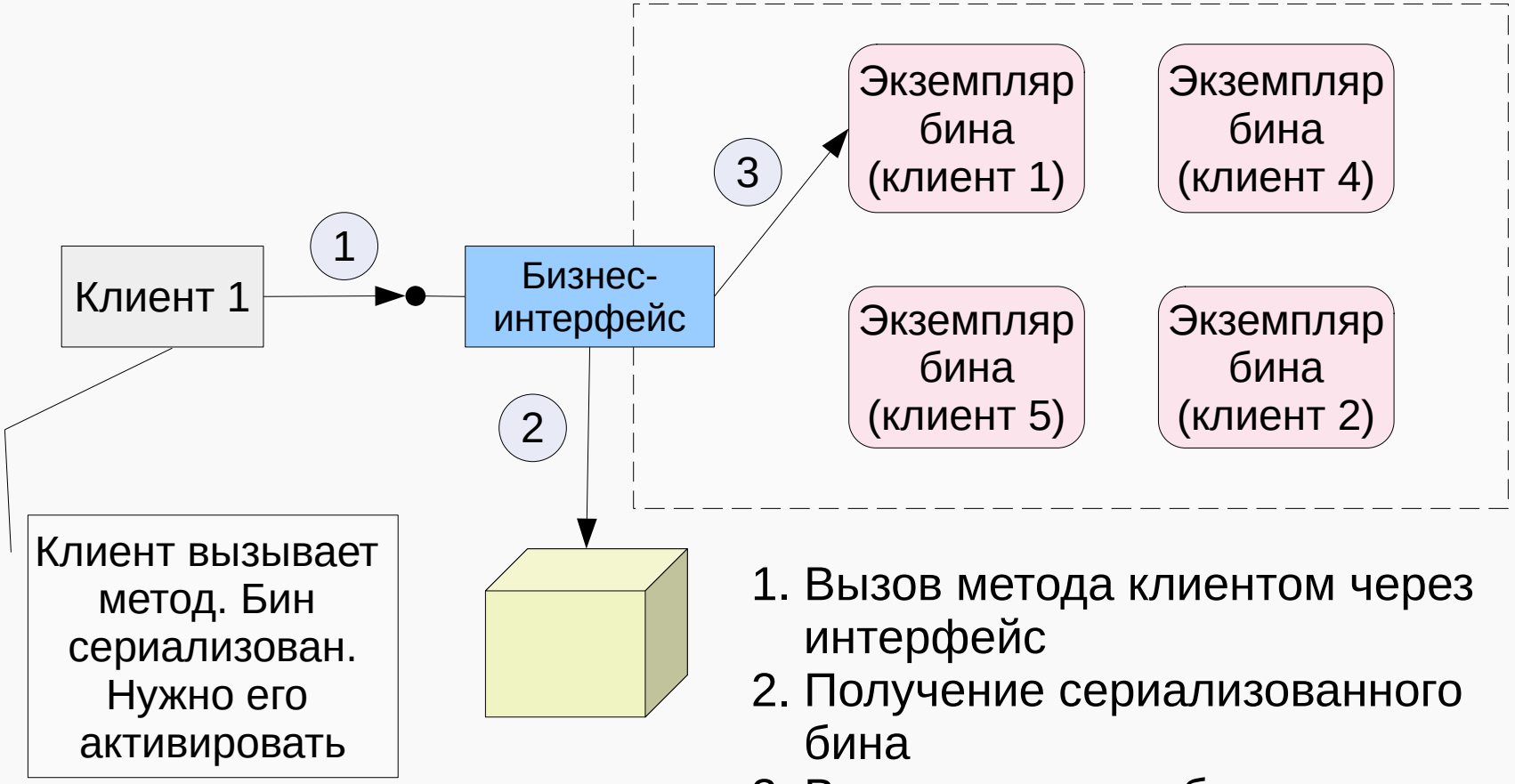


Stateful Session Bean (Passivation)



1. Вызов метода EJB-бина
2. Выбрать бин для передачи в «пассив» (LRU)
   @PrePassivate
   Сериализация состояния бина
3. Сохранение сериализованного состояния

Stateful Session Bean (Activation)

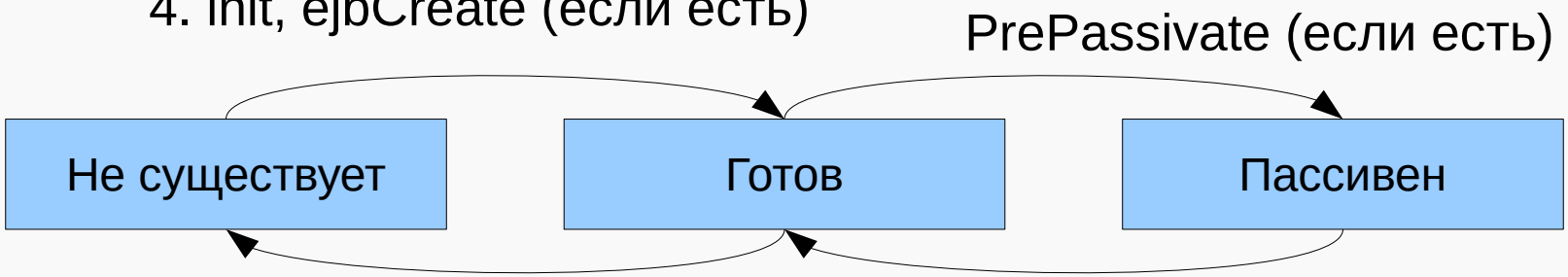


1. Вызов метода клиентом через интерфейс
2. Получение сериализованного бина
3. Восстановление бина
   @PostActivate
   Вызов метода и выполнение

ИТМО ВТ

ЖЦ Stateful Session Bean

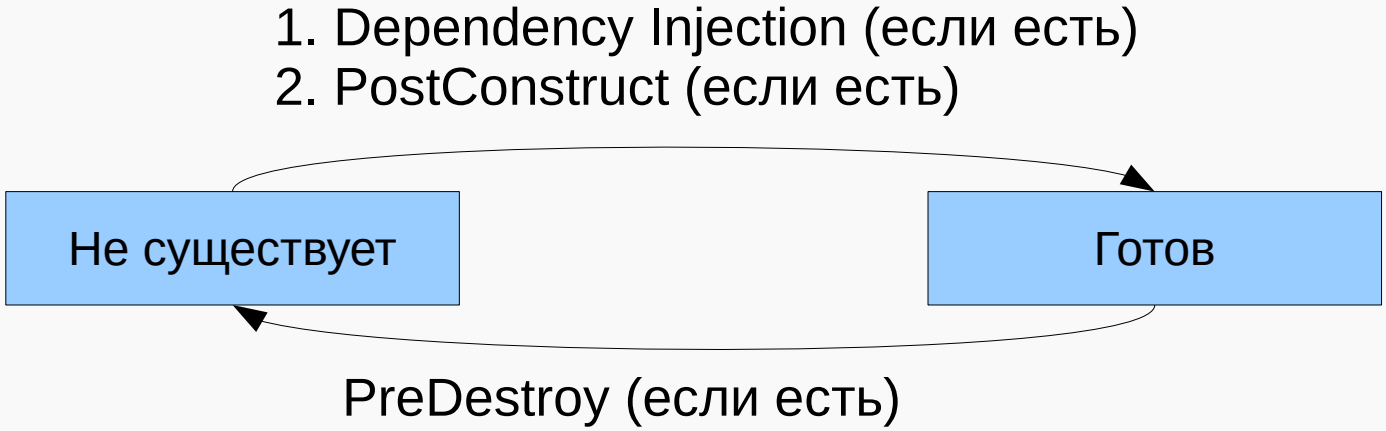
1. Создание
2. Dependency Injection (если есть)
3. PostConstruct (если есть)
4. init, ejbCreate (если есть)



1. Remove
2. PreDestroy (если есть)

Singleton Session Bean

- Singleton — один на приложение.
- Жизненный цикл:



Вызов бина (Local)

```
@Stateless
public class MyConverterBean {
    private BigDecimal yenRate = new
BigDecimal("83.0602");
    private BigDecimal euroRate = new
BigDecimal("0.0093016");

    public BigDecimal dollarToYen(BigDecimal dollars) {
        //implementation
    }

    public BigDecimal yenToEuro(BigDecimal yen) {
        //implementation
    }
}
```

```
public class MyMainBean {
    @EJB
    MyConverterBean myConverterBean;

    public BigDecimal getResult(BigDecimal mv) {
        return myConverterBean.dollarToYen(mv);
    }
}
```

Вызов бина (Local)

```
@Stateless
public class MyConverterBean {
    private BigDecimal yenRate = new
BigDecimal("83.0602");
    private BigDecimal euroRate = new
BigDecimal("0.0093016");

    public BigDecimal dollarToYen(BigDecimal dollars) {
        //implementation
    }

    public BigDecimal yenToEuro(BigDecimal yen) {
        //implementation
    }
}
```

```
public class MyMainBean {
    @EJB
    MyConverterBean myConverterBean;

    public BigDecimal getResult(BigDecimal mv) {
        return myConverterBean.dollarToYen(mv);
    }
}
```

Вызов бина (Remote) (1)

```
package converter.ejb;
//same imports here

public interface MyConverter {
    public BigDecimal dollarToYen(BigDecimal dollars);
    public BigDecimal yenToEuro(BigDecimal yen);
}

@Stateless
@Remote(MyConverter.class)
public class MyConverterBean implements MyConverter {
    //impl

    public BigDecimal dollarToYen(BigDecimal dollars) { // impl }
    public BigDecimal yenToEuro(BigDecimal yen) { //impl }
}
```

```
public class MyMainBean {
    @EJB
    MyConverter myConverterBean;

    public BigDecimal getResult(BigDecimal mv) {
        return myConverterBean.dollarToYen(mv);
    }
}
```

Вызов бина (Remote) (2)

```
package converter.ejb;
//same imports here

public interface MyConverter {
    public BigDecimal dollarToYen(BigDecimal dollars);
    public BigDecimal yenToEuro(BigDecimal yen);
}

@Stateless
@Remote(MyConverter.class)
public class MyConverterBean implements MyConverter {
    //impl

    public BigDecimal dollarToYen(BigDecimal dollars) { // impl }
    public BigDecimal yenToEuro(BigDecimal yen) { //impl }
}
```

```
public class MyMainBean {
        MyConverter myConverterBean;

        public BigDecimal getResult(BigDecimal mv) {
            //add myConverter null checking
            myConverterBean = (MyConverter)InitialContext
                .lookup("java:global/myApp/MyConverter");
            return myConverterBean.dollarToYen(mv);
        }
}
```