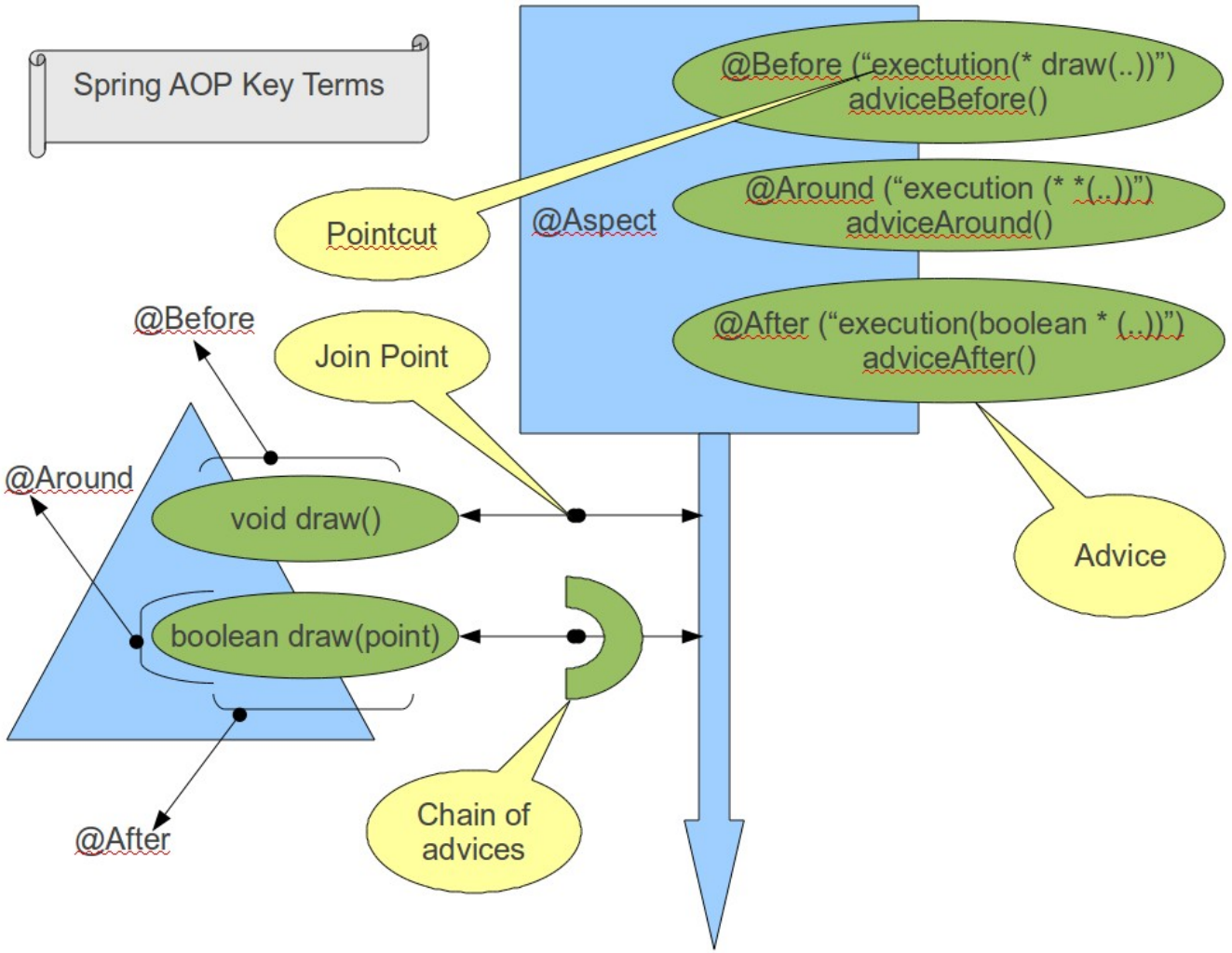
Spring Framework ::
Дополнительные материалы слушателя

Vyacheslav Yakovenko
March, 2012

Spring :: AOP


Spring AOP Key Terms
@Before ("exectution(* draw(..))")
adviceBefore()
@Around ("execution (* *(..))")
adviceAround()
@After ("execution(boolean * (..))")
adviceAfter()
Pointcut
@Aspect
@Before
Join Point
@Around
void draw()
boolean draw(point)
Advice
Chain of advices
@After

Transactions API :: AOP

LUXOFT
Engineering Business Performance

Подробное описание языка Pointcut-ов: http://www.eclipse.org/aspectj/doc/next/progguide/semantics-pointcuts.html (short url: http://goo.gl/Tr42D)

Transactions API :: Isolation

LUXOFT
Engineering Business Performance

При параллельном использовании транзакций могут возникать следующие проблемы:

- потерянное обновление (lost update);
- «грязное» чтение (dirty read) — чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится(откатится);
- неповторяющееся чтение (non-repeatable read);
- фантомное чтение (phantom reads).

Transactions API :: Isolation

LUXOFT
Engineering Business Performance

Потерянное обновление

Предположим, имеются две транзакции, открытые различными приложениями, в которых выполнены следующие SQL-операторы: В транзакции 1 изменяется значение поля f2, а затем в транзакции 2 также изменяется значение этого поля. В результате изменение, выполненное первой транзакцией, будет потеряно.

Транзакция 1	Транзакция 2
`SELECT f2 FROM tbl1 WHERE f1=1;`	`SELECT f2 FROM tbl1 WHERE f1=1;`
`UPDATE tbl1 SET f2=f2+20 WHERE f1=1;`	
	`UPDATE tbl1 SET f2=f2+25 WHERE f1=1;`

Transactions API :: Isolation

LUXOFT
Engineering Business Performance

«Грязное» чтение

Предположим, имеются две транзакции, открытые различными приложениями, в которых выполнены следующие SQL-операторы: В транзакции 1 изменяется значение поля f2, а затем в транзакции 2 выбирается значение этого поля. После этого происходит откат транзакции 1. В результате значение, полученное второй транзакцией, будет отличаться от значения, хранимого в базе данных.

Транзакция 1	Транзакция 2
`SELECT f2 FROM tbl1 WHERE f1=1;`	
`UPDATE tbl1 SET f2=f2+1 WHERE f1=1;`	
	`SELECT f2 FROM tbl1 WHERE f1=1;`
`ROLLBACK WORK;`	

Transactions API :: Isolation

LUXOFT
Engineering Business Performance

Неповторяющееся чтение

Предположим, имеются две транзакции, открытые различными приложениями, в которых выполнены следующие SQL-операторы: В транзакции 2 выбирается значение поля f2, затем в транзакции 1 изменяется значение поля f2. При повторной попытке выбора значения из поля f2 в транзакции 2 будет получен другой результат. Эта ситуация особенно неприемлема, когда данные считываются с целью их частичного изменения и обратной записи в базу данных.

Транзакция 1	Транзакция 2
`SELECT f2 FROM tbl1 WHERE f1=1;`	`SELECT f2 FROM tbl1 WHERE f1=1;`
`UPDATE tbl1 SET f2=f2+1 WHERE f1=1;`	
`COMMIT;`	
	`SELECT f2 FROM tbl1 WHERE f1=1;`

Transactions API :: Isolation

LUXOFT
Engineering Business Performance

Фантомное чтение

Предположим, имеется две транзакции, открытые различными приложениями, в которых выполнены следующие SQL-операторы: В транзакции 2 выполняется SQL-оператор, использующий все значения поля f2. Затем в транзакции 1 выполняется вставка новой строки, приводящая к тому, что повторное выполнение SQL-оператора в транзакции 2 выдаст другой результат. Такая ситуация называется фантомным чтением. От неповторяющегося чтения оно отличается тем, что результат повторного обращения к данным изменился не из-за изменения/удаления самих этих данных, а из-за появления новых (фантомных) данных.

Транзакция 1	Транзакция 2
	`SELECT SUM(f2) FROM tbl1;`
`INSERT INTO tbl1 (f1,f2) VALUES (15,20);`	
	`SELECT SUM(f2) FROM tbl1;`

Transactions API :: Isolation

LUXOFT
Engineering Business Performance

Поведение при различных уровнях изолированности

«+» — предотвращает, «–» — не предотвращает.

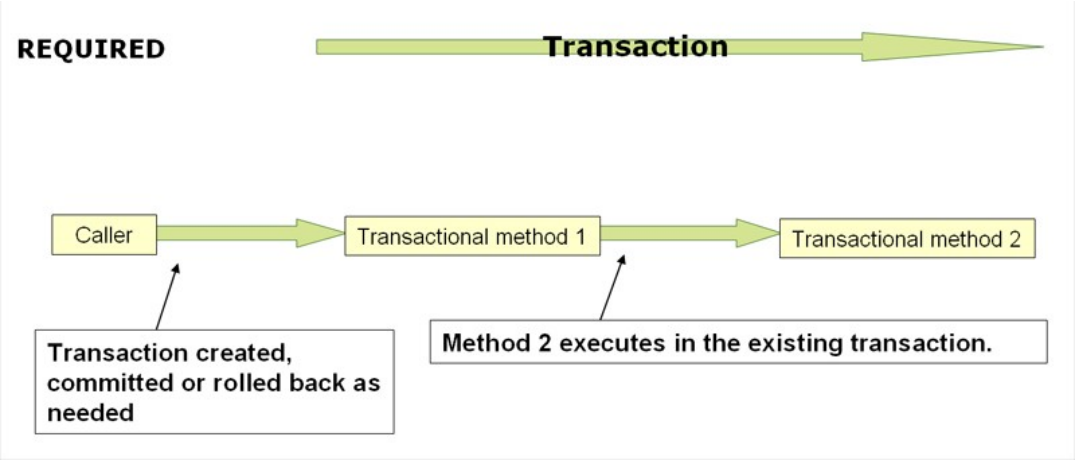
Уровень изоляции	Фантомная вставка	Неповторяющееся чтение	«Грязное» чтение	Потерянное обновление[3]
SERIALIZABLE	+	+	+	+
REPEATABLE READ	–	+	+	+
READ COMMITTED	–	–	+	+
READ UNCOMMITTED	–	–	–	+

Transactions API
Propagation

LUXOFT
Engineering Business Performance

Способы передачи транзакций от метода к методу:

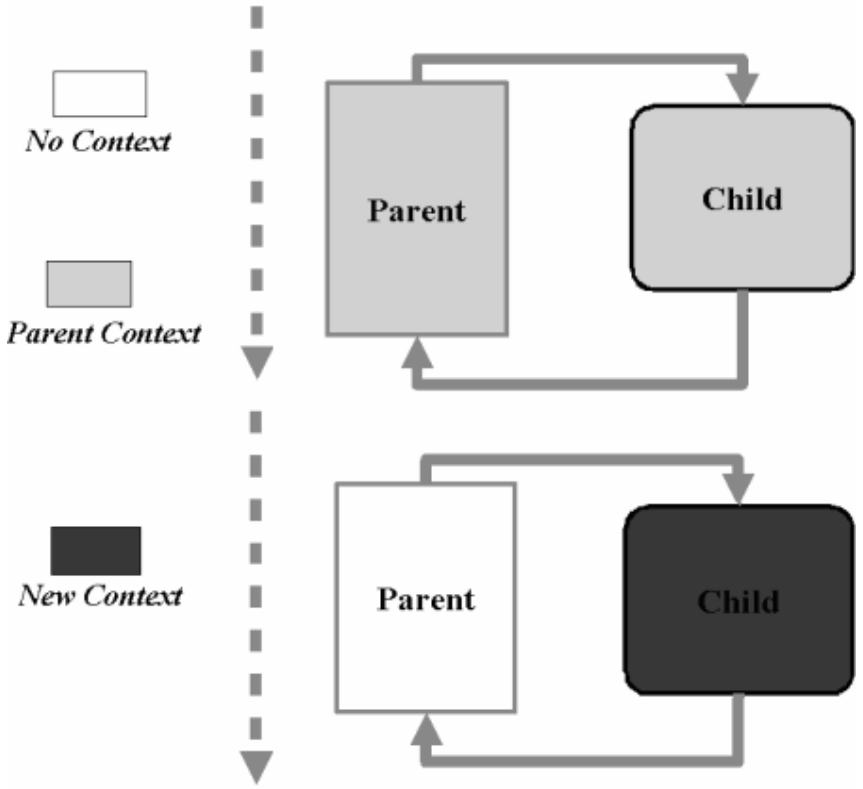
- MANDATORY
- NESTED
- NEVER
- NOT_SUPPORTED
- REQUIRED
- REQUIRES_NEW
- SUPPORTS



LUXOFT
Engineering Business Performance

Propagation. REQUIRED

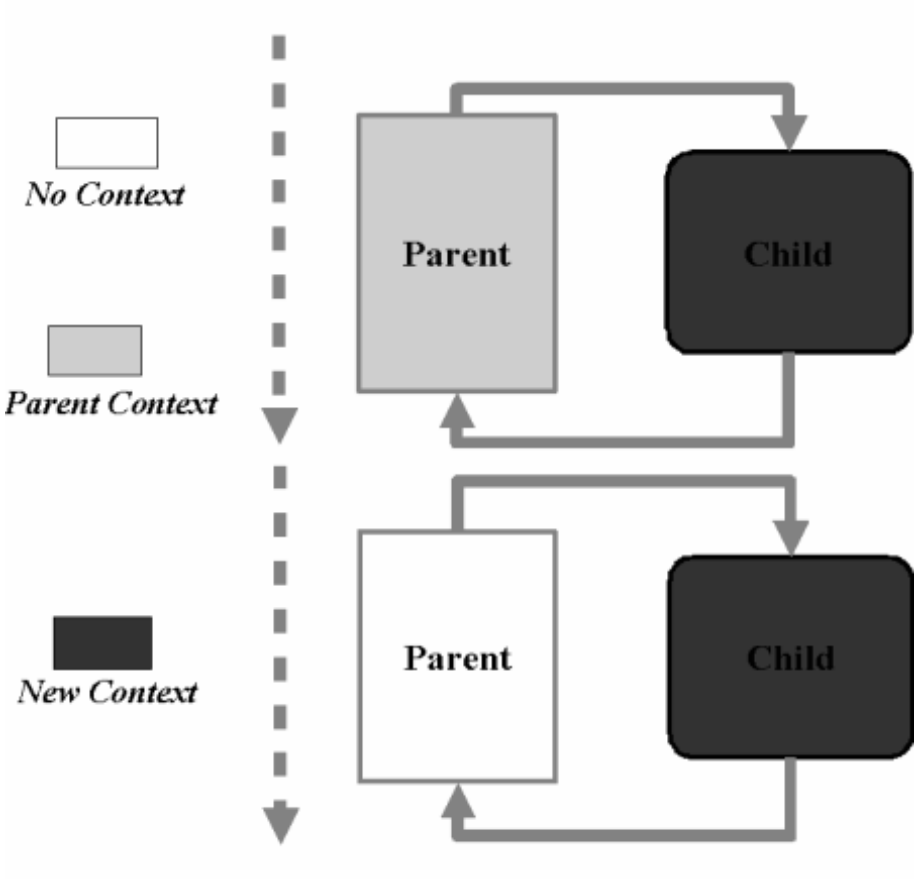
- Метод всегда выполняется внутри транзакции
- Если вызывающий метод выполняется внутри транзакции, то данный метод выполняется в той же транзакции
- Если вызывающий метод не имеет транзакции, то создается новая



LUXOFT
Engineering Business Performance

Propagation. REQUIRES_NEW

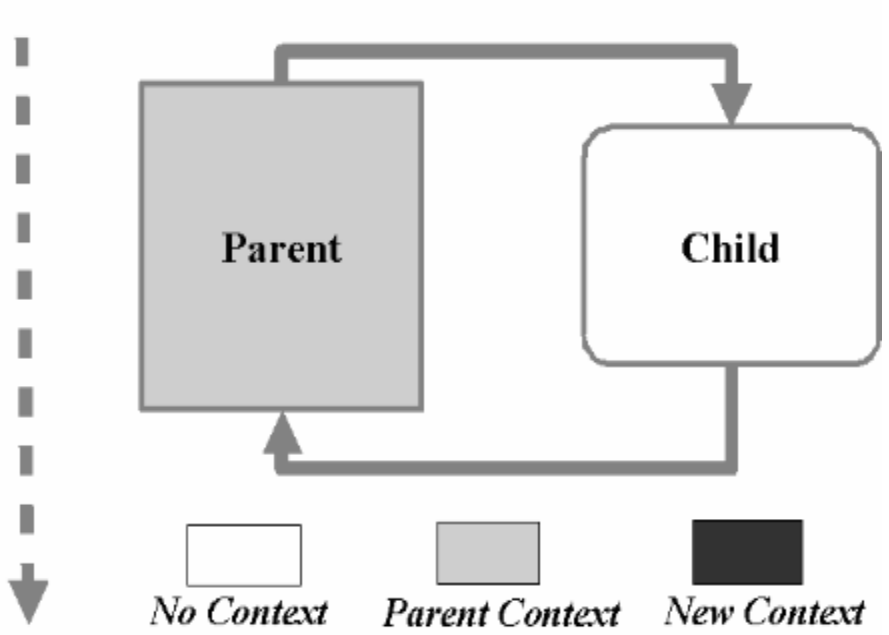
- Всегда создается новая транзакция при вызове метода
- Если вызывающий метод выполняется внутри транзакции, то она приостанавливается, создается новая, в ней выполняется текущий метод, после выполнения метода остановленная транзакция возобновляется



LUXOFT
Engineering Business Performance

Propagation. NOT_SUPPORTED

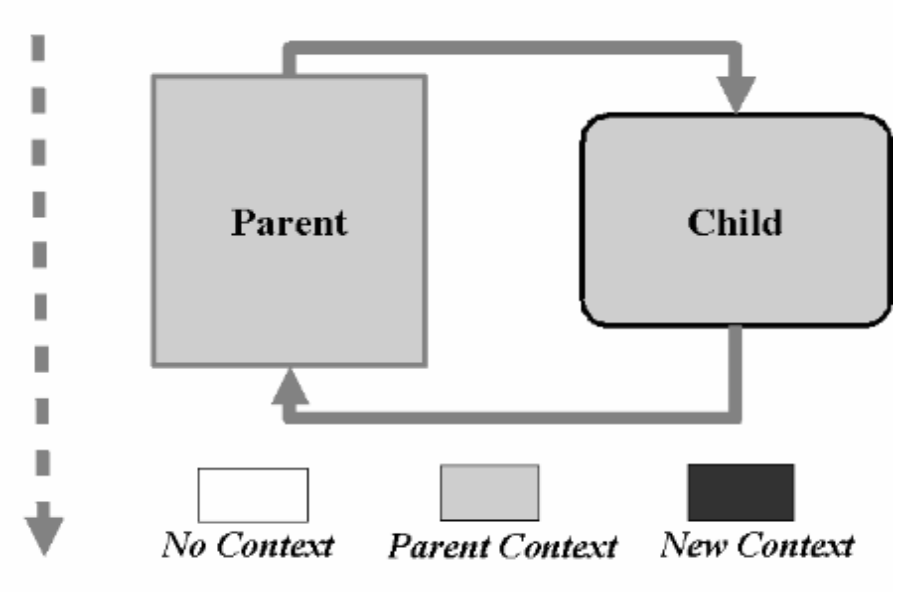
- Если вызвавший метод выполняется внутри транзакции, она приостанавливается, пока выполняется текущий метод, а после завершения метода возобновляется
- Если вызвавший метод вне транзакции, то она не создается



LUXOFT
Engineering Business Performance

Propagation. SUPPORTS

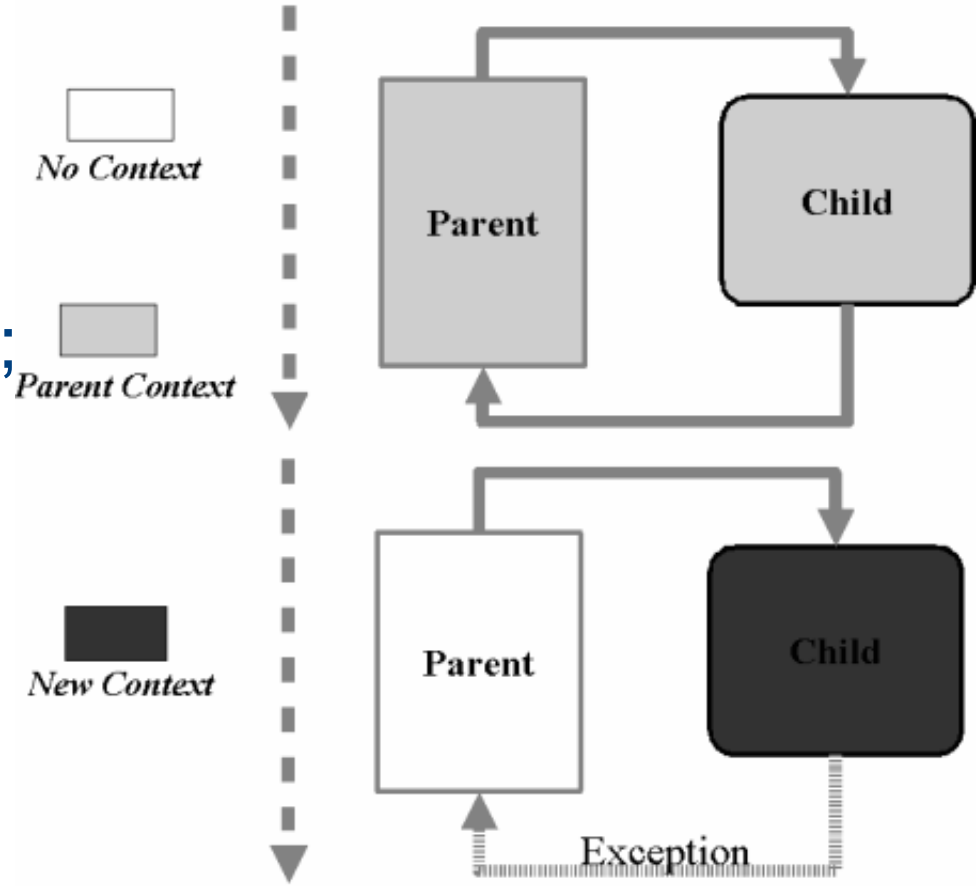
- Если вызвавший метод выполняется внутри транзакции, то текущий метод выполняется в той же транзакции;
- Если вызвавший метод выполняется вне транзакции, то новая транзакция не создается и текущий метод выполняется вне транзакции тоже;



LUXOFT
Engineering Business Performance

Propagation. MANDATORY

- Метод всегда должен выполняться в уже созданной транзакции;
- Если метод вызывается вне транзакции, то выбрасывается исключение.



LUXOFT
Engineering Business Performance

Propagation.NEVER

- Метод всегда выполняется вне транзакции
- Если метод вызывается внутри транзакции, то выбрасывается исключение

LUXOFT
Engineering Business Performance

Propagation.NESTED

- Использует физически одну транзакцию со многими точками (savepoint) к которым можно откатить транзакцию;
- Такая частичная отмена транзакций позволяет внешней транзакции продолжаться, даже если внутренняя транзакция была отменена;

Spring :: Task :: Quartz

LUXOFT
Engineering Business Performance

Схема взаимодействия бинов в общем случае: