Transaction Management

Andrew Morgan
INDEPENDENT CONSULTANT
@mogronalol

Overview

What is a transaction?

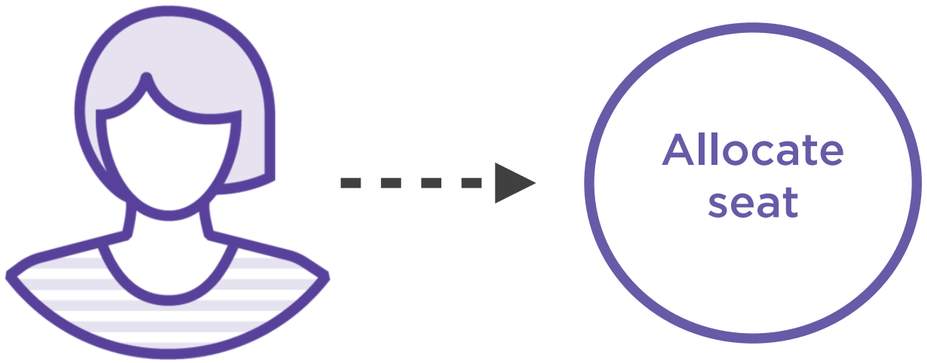
What's the problem with coding our own transactions?

Learning how to use Spring Data to manage our transactions

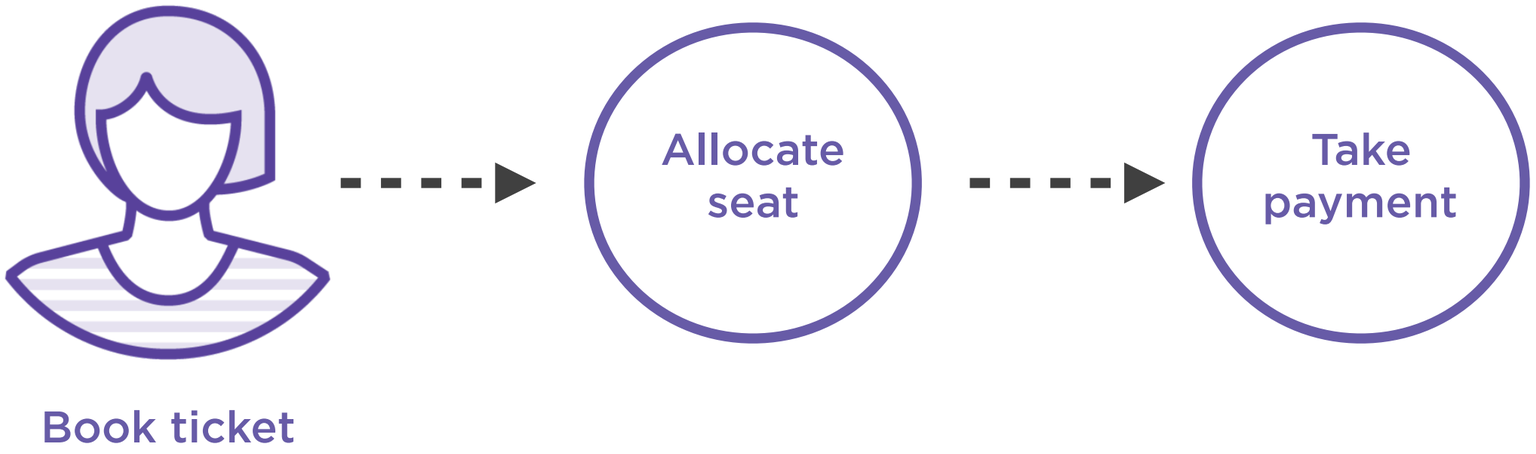
Why We Need Transactions

Book ticket

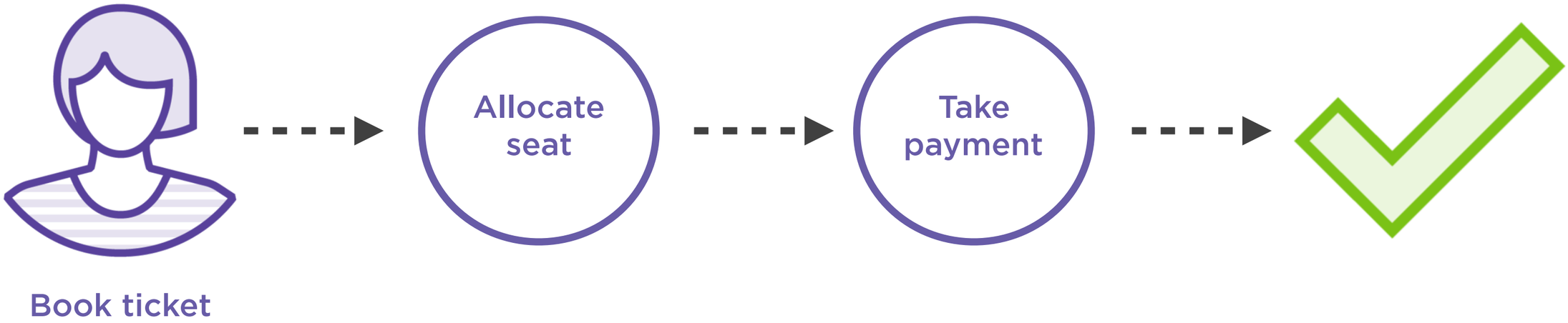
Why We Need Transactions



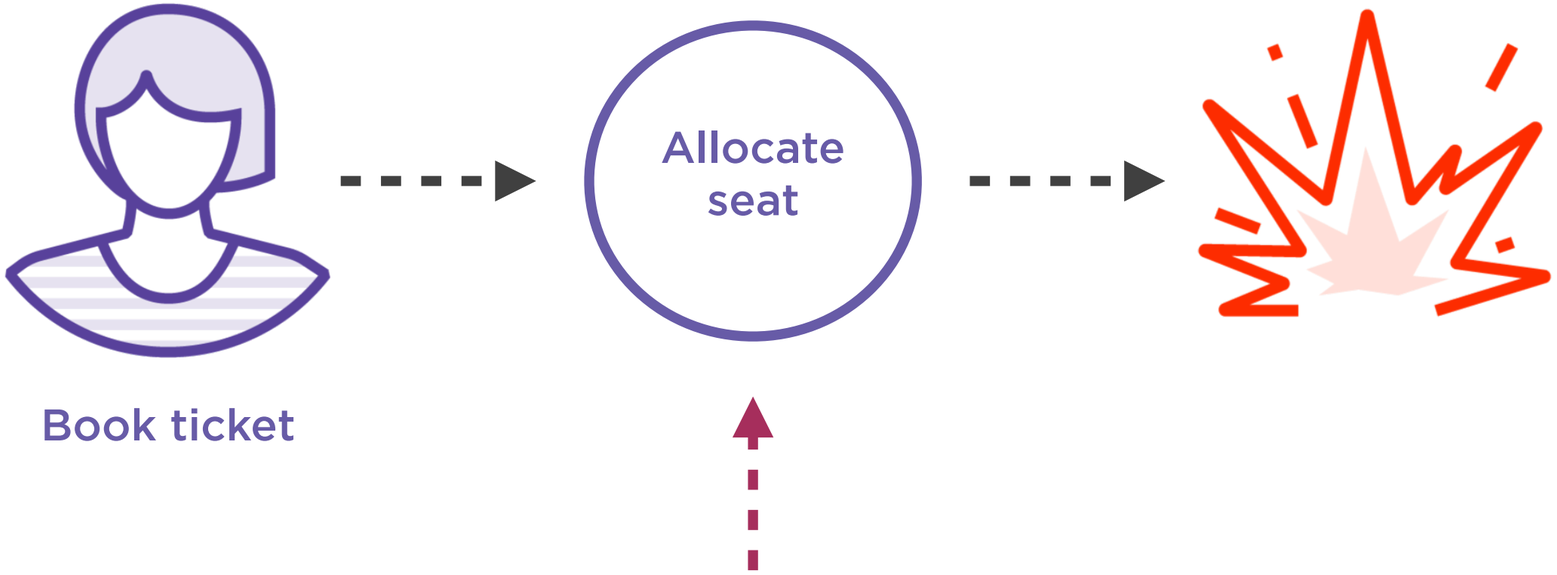
Why We Need Transactions



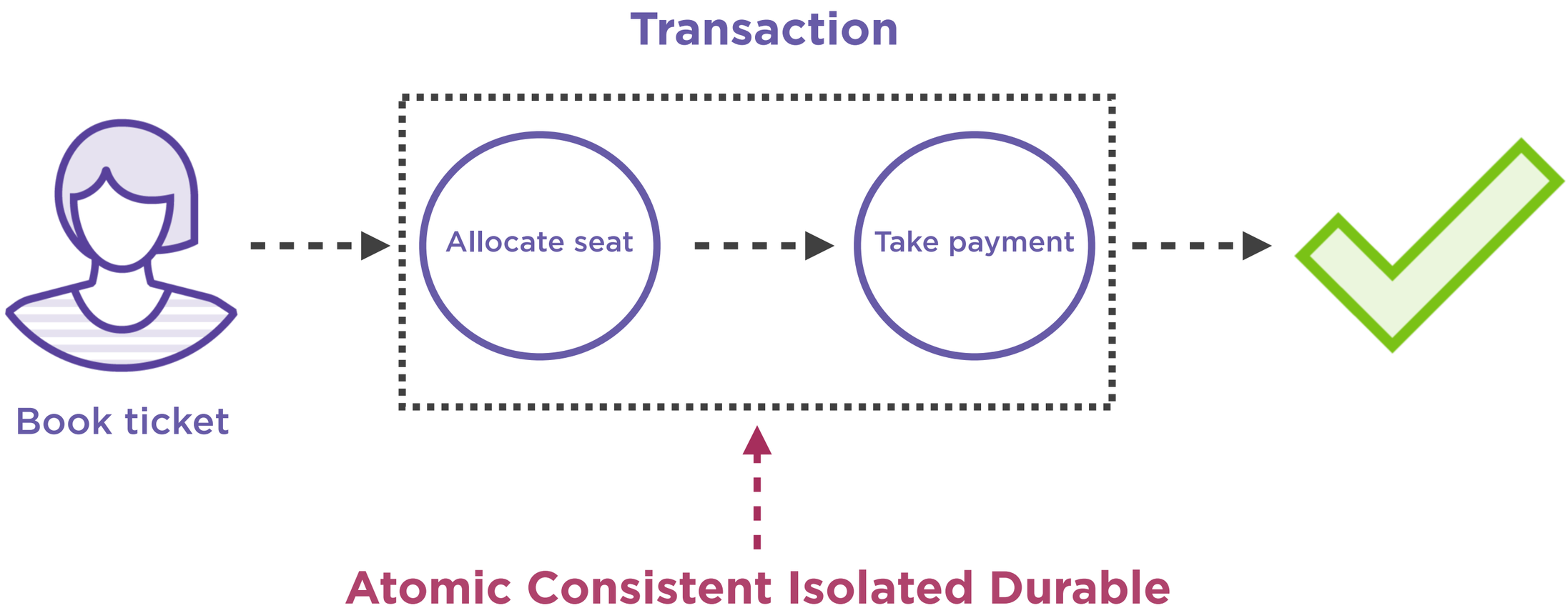
Why We Need Transactions



Why We Need Transactions



Why We Need Transactions



```
bookTicket(ticket, payment);
```

```
Session sess = factory.openSession();
Transaction tx;
try {
    tx = sess.beginTransaction();

    bookTicket(ticket, payment);

    tx.commit();
}
catch (Exception e) {
    if (tx!=null) tx.rollback();
    throw e;
}
finally {
    sess.close();
}
```

```
Session sess = factory.openSession();
Transaction tx;
try {
    tx = sess.beginTransaction();

    bookTicket(ticket, payment);

    tx.commit();
}
catch (Exception e) {
    if (tx!=null) tx.rollback();
    throw e;
}
finally {
    sess.close();
}
```

◀ Coupled to Hibernate

```
Session sess = factory.openSession();
Transaction tx;
try {
    tx = sess.beginTransaction();

    bookTicket(ticket, payment);

    tx.commit();
}
catch (Exception e) {
    if (tx!=null) tx.rollback();
    throw e;
}
finally {
    sess.close();
}
```

- Coupled to Hibernate
- **Gotta start it**

```
Session sess = factory.openSession();
Transaction tx;
try {
    tx = sess.beginTransaction();

    bookTicket(ticket, payment);

    tx.commit();
}
catch (Exception e) {
    if (tx!=null) tx.rollback();
    throw e;
}
finally {
    sess.close();
}
```

- Coupled to Hibernate
- Gotta start it
- **Gotta commit it**

```
Session sess = factory.openSession();
Transaction tx;
try {
    tx = sess.beginTransaction();

    bookTicket(ticket, payment);

    tx.commit();
}
catch (Exception e) {
    if (tx!=null) tx.rollback();
    throw e;
}
finally {
    sess.close();
}
```

- Coupled to Hibernate
- Gotta start it
- Gotta commit it
- **Remember it's in a try-catch so we can rollback!**

```
Session sess = factory.openSession();
Transaction tx;
try {
    tx = sess.beginTransaction();

    bookTicket(ticket, payment);

    tx.commit();
}
catch (Exception e) {
    if (tx!=null) tx.rollback();
    throw e;
}
finally {
    sess.close();
}
```

- Coupled to Hibernate
- Gotta start it
- Gotta commit it
- Remember it's in a try-catch so we can rollback!
- **Must remember the finally block!**

The Transactional Annotation

```
public void bookTicket(Booking booking) {
  allocateSeat(booking.getSeat())
  makePayment(booking.getCardDetails())
}
```

The Transactional Annotation

```
@Transactional
public void bookTicket(Booking booking) {
  allocateSeat(booking.getSeat())
  makePayment(booking.getCardDetails())
}
```

The Transactional Annotation

Does it all for us!

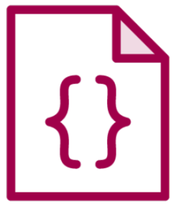
```
@Transactional
public void bookTicket(Booking booking) {
  allocateSeat(booking.getSeat())
  makePayment(booking.getCardDetails())
}
```

Benefits of Transaction Management

No more boilerplate

Benefits of Transaction Management

No more boilerplate

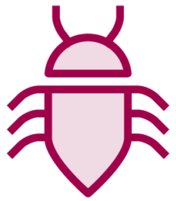


Declarative and non-invasive

Benefits of Transaction Management

No more boilerplate

Declarative and non-invasive



Bugs less likely

Benefits of Transaction Management

No more boilerplate

Declarative and non-invasive

Bugs less likely

Data-store agnostic

Demo

Creating a transactional method

Verify it is transactional by triggering rollback

Summary

Transaction code can produce too much boilerplate

The transactional annotation is a better alternative

We can use it by simply annotating the method that we want to be executed in a transaction