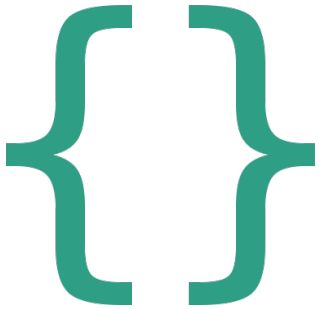
webtide

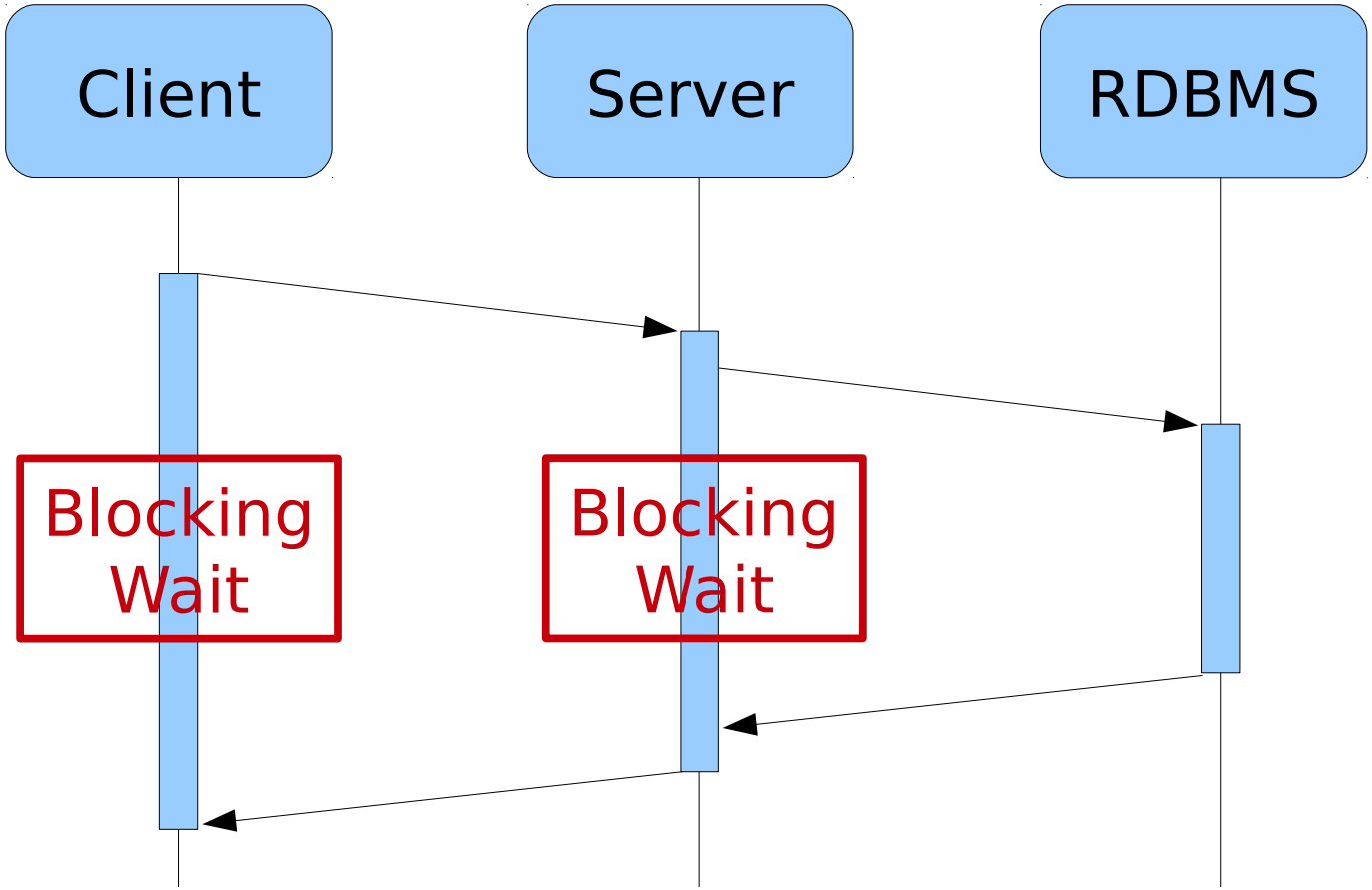


ApiConf

Reactive APIs

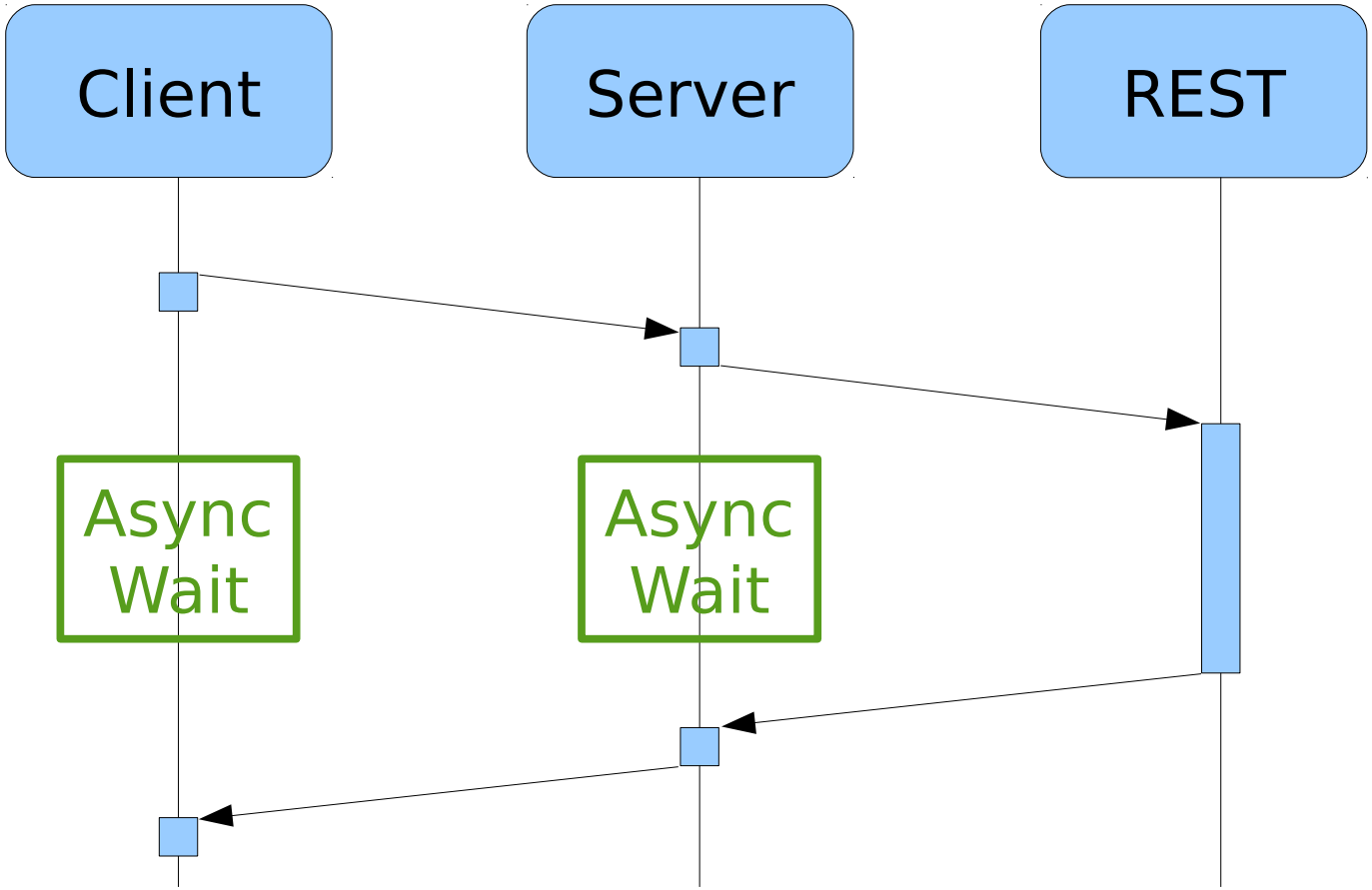
Simone Bordet
sbordet@webtide.com

Why Reactive



Simone Bordet
sbordet@webtide.com

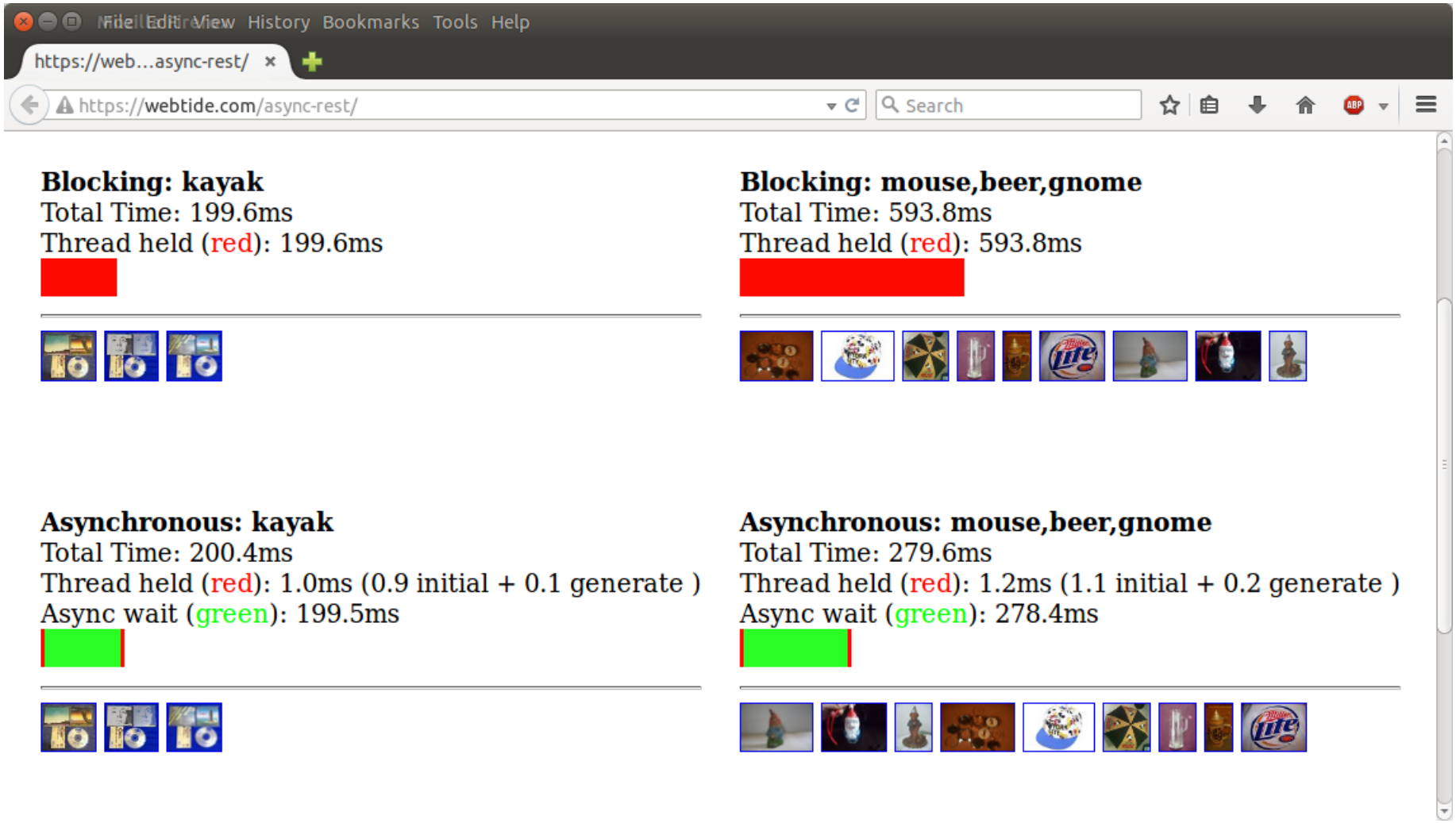
Why Reactive



Simone Bordet
sbordet@webtide.com

Why Reactive

https://webtide.com/async-rest/



Simone Bordet
sbordet@webtide.com

Reactive Streams API

```
public interface Publisher<T> {
    void subscribe(Subscriber<? super T> s);
}
public interface Subscriber<T> {
    void onSubscribe(Subscription s);
    void onNext(T t);
    void onError(Throwable t);
    void onComplete();
}
public interface Subscription {
    void request(long n);
    void cancel();
}
public interface Processor<T, R> extends Subscriber<T>, Publisher<R> {}
```

Simone Bordet
sbordet@webtide.com

Reactive Streams API

- **JDK 9**
  - java.util.concurrent.Flow
  - http://download.java.net/java/jdk9/docs/api/java/util/concurrent/Flow.html

```
public class Flow {
    public interface Publisher<T> { ... }
    public interface Subscriber<T> { ... }
    public interface Subscription { ... }
    public interface Processor<T, R> ...
}
```

Simone Bordet
sbordet@webtide.com

Reactive Streams API

MongoDB Example

```
MongoCollection<Document> collection = ...
Publisher<Success> publisher = collection.insertOne(document);
publisher.subscribe(new Subscriber<Success>() {
    public void onSubscribe(Subscription s) {
        s.request(1);  // <-- Only now insertion will occur
    }
    public void onNext(Success success) { /* Inserted */ }
    public void onError(Throwable t) { /* Failed */ }
    public void onComplete() {}
});
```

Simone Bordet
sbordet@webtide.com

RxJava 2 APIs

```
Publisher<Attendee> attendees = ...;
Flowable.fromPublisher(attendees)
    .filter(attendee -> attendee.birth.getMonth() == JULY)
    .map(attendee -> attendee.addNote("Awesome"))
    .flatMap(attendee -> saveViaREST(attendee))
    .subscribe(new Subscriber<Result>() {
        public void onSubscribe(Subscription s) {
            _subscription = s; s.request(1);
        }
        public void onNext(Document document) {
            System.out.println(document);
            _subscription.request(1);
        }
        ...
    });
```

Simone Bordet
sbordet@webtide.com

Reactor APIs

```
Publisher<Attendee> attendees = ...;
Flux.from(attendees)
    .filter(attendee -> attendee.birth.getMonth() == JULY)
    .map(attendee -> attendee.addNote("Awesome"))
    .flatMap(attendee -> saveViaREST(attendee))
    .subscribe(new Subscriber<Result>() {
        public void onSubscribe(Subscription s) {
            _subscription = s; s.request(1);
        }
        public void onNext(Document document) {
            System.out.println(document);
            _subscription.request(1);
        }
        ...
    });
```

Simone Bordet
sbordet@webtide.com

Conclusions

- **Embrace Reactive Programming**
- **It’s good for you**
  - Functional programming
  - Asynchronous programming
  - Standard APIs (learn one, learnt all)
- **It’s good for your business**
  - More scalable systems
  - Less costs, more profit

Simone Bordet
sbordet@webtide.com