TDD + IoC = LOVE

Антон Белоусов,
основатель http://tai.st
about.me/antonbelousov

TDD — что это?

- Не «сначала тесты, потом код»
- Вообще не про тесты
- TDD — способ проектирования

О докладчике

- 10 лет программирования
- Олимпиадное/Enterprise/Веб-сервисы
- Разработчик/Team Lead/PM
- Core: C#, PHP, Javascript
- Люблю Coffeescript

Сравнение

- Без TDD:
  - Кое-как формулируем задачу словами
  - Фигачим код
  - Пытаемся запустить
- TDD:
  - Четкие спецификации словами
  - Спецификации программным кодом
  - Код, соответствующий спецификациям
  - Сразу работает

Пример: бронирование билетов

- Делаем сервис бронирования авиабилетов.
- Задача: отображать рейсы, доступные на заданную дату
- Для TDD используем Jasmine - http://pivotal.github.com/jasmine/

Спецификация кодом

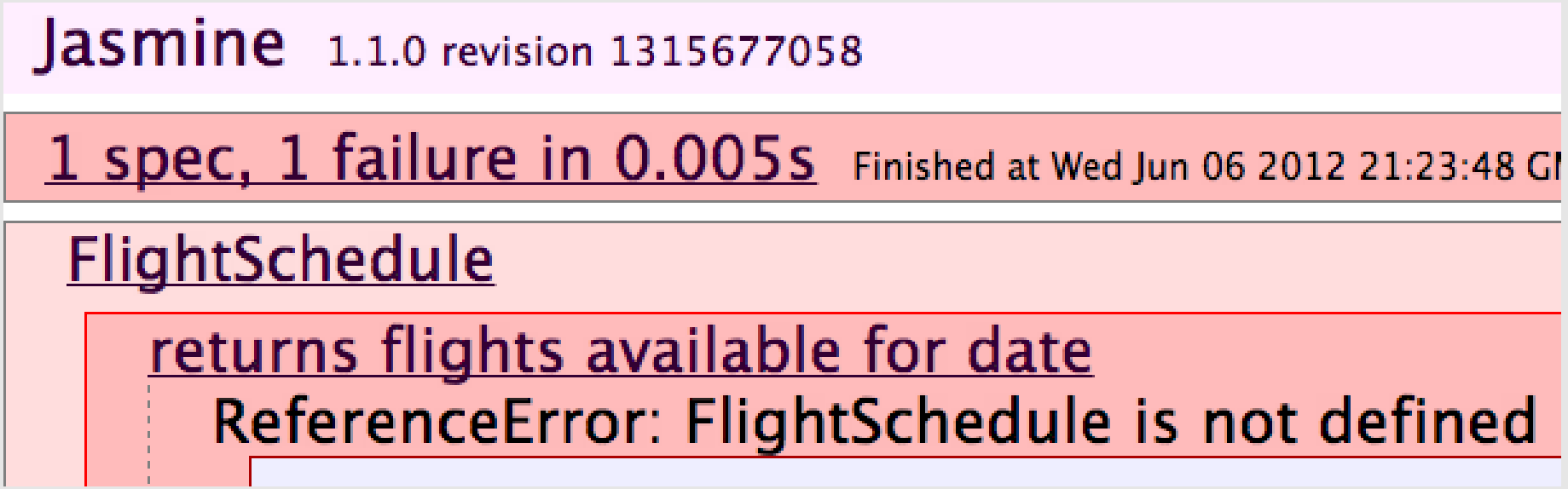
- Пишем до создания кода:

```
describe('FlightSchedule', function(){
    it('returns flights available for date', function(){
        schedule = new FlightSchedule()

        expect(schedule.getAvailableFlightsForDate(
                new Date())).toEqual([1])
    })
})
```

Проверим, что не работает

- Jasmine выдает ошибки:



Реализация - заглушка

- Проще начать с заглушки

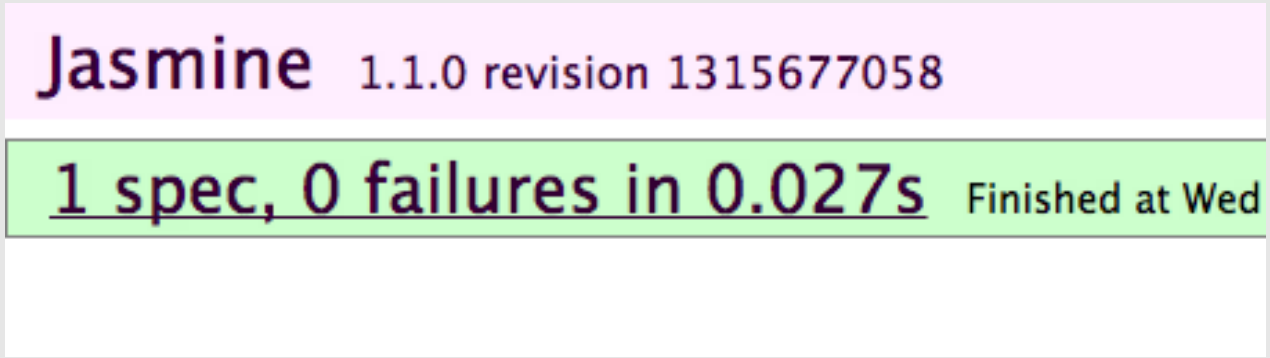
```
function FlightSchedule(){}

FlightSchedule.prototype.
    getAvailableFlightsForDate = function(date){

    return [1];
}
```

Проверяем

- Уже работает



- Надо сделать рабочую
- Нужны зависимости — получать рейсы

Добавляем зависимости

- Сначала меняем спецификацию, затем — реализацию
- Не думаем о природе зависимостей — только об интерфейсе
- Используем заглушки

Меняем спецификацию

```
it('returns flights available for date', function(){
    var mockFlightsCollection = {
        getAllFlights: function() {
            return [{
                    fitsDate: function(date) {return true;},
                    isAvailable: function() {return true;},
                    getNumber: function() {return 1;}
    }]}}

    var schedule = new FlightSchedule()
    schedule._flightsCollection = mockFlightsCollection;
```

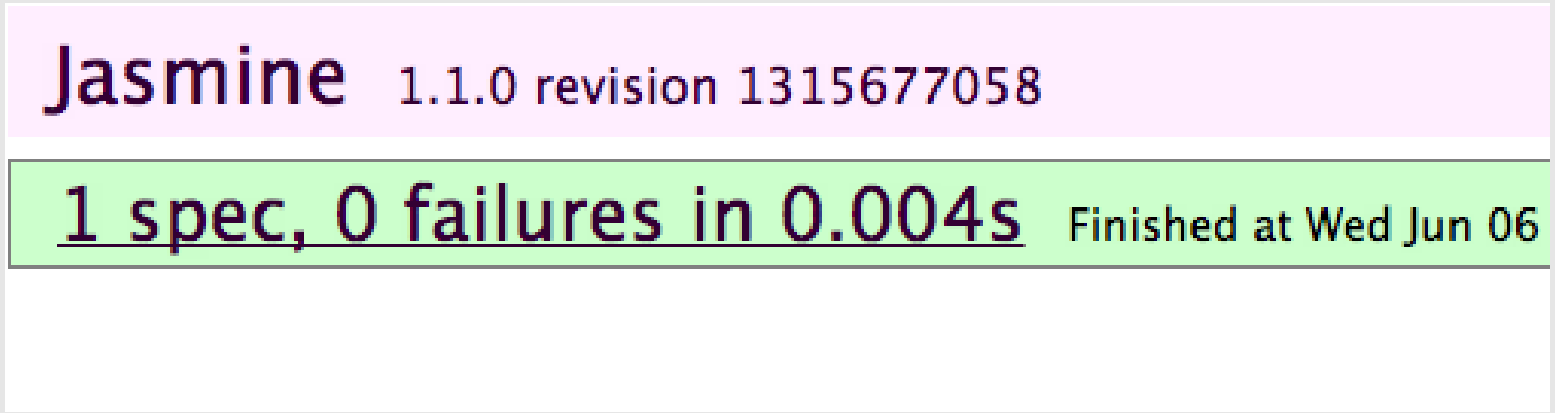
Меняем реализацию

```
getAvailableFlightsForDate = function(date){
var allFlights = this._fliahtsCollection.getAllFlights();
var results = [];

for(var i=0; i < allFlights.length; i++) {
    var flight = allFlights[i];
    if(flight.fitsDate(date)) {
        if(flight.isAvailable())
        results.push(flight.getNumber())
    }
}
return results;}
```

Проверяем

- Опять работает



- Реализации зависимостей нет, интерфейс уже есть

Как проставлять зависимости?

- Вручную
- IocContainer

Вручную: дублирование, ошибки

```
var database = new Database();

var flightSchedule1 = new FlightSchedule();
var flightsCollection1 = new FlightsCollection();

flightSchedule1._flightsCollection = flightsCollection1;
flightsCollection1._database = database;

var flightSchedule2 = new FlightSchedule();
var flightsCollection2 = new FlightsCollection();

flightSchedule2._flightsCollection = flightsCollection2;
flightsCollection2._database = database;
```

Решение: IocContainer

- Используется во всех «Enterprise»-языках
- Spring, PicoContainer, etc.
- Очень сложные, с костылями
- Что в Javascript?

Proton IoC

```
var dependencySchema = {
    schedule: {
        multiple: FlightSchedule,
        deps: {_flightsCollection: 'flightsCollection'}
    },
    flightsCollection: {
        multiple: FlightsCollection,
        deps: {_database: 'database'}
    },
    database: {single: Database }
}
var iocContainer = new IocContainer();
iocContainer.setSchema(dependencySchema);
```

Proton IoC

- **single**: единый экземпляр (аналог singleton в Spring)
- **multiple**: много экземляров (аналог prototype)
- **ref**: прямая ссылка (полезна для библиотек)
- **factoryFunction**: обертка для создания новых элементов с простановкой зависимостей

Proton IoC

- github.com/abelousov/proton
- Работает для браузера
- Код/проверки - 50/50
- Используется в tai.st
- TBD: NodeJS, Контракты, АОП

Спасибо!

Можно задавать вопросы)

Антон Белоусов,
основатель http://tai.st
about.me/antonbelousov