BeanDiffer2

Hamcrest Matcher

А. С. Семёнов
semkagtn@yandex-team.ru

Яндекс

15 октября 2015 г.

Содержание

Содержание

Мотивация

Мотивация

- Хочется иметь универсальный Matcher

Мотивация

- Хочется иметь универсальный Matcher
- Отчёт должен быть информативным

Мотивация

- Хочется иметь универсальный Matcher
- Отчёт должен быть информативным
- Matcher должен быть расширяемым

Мотивация

- Хочется иметь универсальный Matcher
- Отчёт должен быть информативным
- Matcher должен быть расширяемым
- Matcher должен быть гибким

Содержание

BeanEquals

BeanEquals

- Плюсы

BeanEquals

- Плюсы
  - Позволяет сравнивать объекты различного типа

BeanEquals

- Плюсы
  - Позволяет сравнивать объекты различного типа
  - Даёт возможность задавать стратегии сравнения

BeanEquals

- Плюсы
  - Позволяет сравнивать объекты различного типа
  - Даёт возможность задавать стратегии сравнения
- Минусы

BeanEquals

- Плюсы
  - Позволяет сравнивать объекты различного типа
  - Даёт возможность задавать стратегии сравнения
- Минусы
  - Позволяет сравнивать только JavaBean объекты

BeanEquals

- Плюсы
  - Позволяет сравнивать объекты различного типа
  - Даёт возможность задавать стратегии сравнения
- Минусы
  - Позволяет сравнивать только JavaBean объекты
  - Отсутствует возможность сравнивать объекты «вглубь»

BeanEquals

- Плюсы
  - Позволяет сравнивать объекты различного типа
  - Даёт возможность задавать стратегии сравнения
- Минусы
  - Позволяет сравнивать только JavaBean объекты
  - Отсутствует возможность сравнивать объекты «вглубь»
  - Стратегии сравнения не очень гибкие

BeanDiffer

BeanDiffer

- Плюсы

BeanDiffer

- Плюсы
  - Позволяет сравнивать не только JavaBean объекты

BeanDiffer

- Плюсы
  - Позволяет сравнивать не только JavaBean объекты
  - Даёт возможность задавать гибкие стратегии сравнения

BeanDiffer

- Плюсы
  - Позволяет сравнивать не только JavaBean объекты
  - Даёт возможность задавать гибкие стратегии сравнения
- Минусы

BeanDiffer

- Плюсы
  - Позволяет сравнивать не только JavaBean объекты
  - Даёт возможность задавать гибкие стратегии сравнения
- Минусы
  - Баги позволяют использовать весь функционал

BeanDiffer

- Плюсы
  - Позволяет сравнивать не только JavaBean объекты
  - Даёт возможность задавать гибкие стратегии сравнения
- Минусы
  - Баги позволяют использовать весь функционал
  - Присутсвуют также другие мелкие баги

BeanDiffer

- Плюсы
  - Позволяет сравнивать не только JavaBean объекты
  - Даёт возможность задавать гибкие стратегии сравнения
- Минусы
  - Баги позволяют использовать весь функционал
  - Присутсвуют также другие мелкие баги
  - Баги в реализации сложно исправить

BeanDiffer

- Плюсы
  - Позволяет сравнивать не только JavaBean объекты
  - Даёт возможность задавать гибкие стратегии сравнения
- Минусы
  - Баги позволяют использовать весь функционал
  - Присутсвуют также другие мелкие баги
  - Баги в реализации сложно исправить
  - Плохая расширяемость

Содержание

Что такое поле?

Что такое поле?

- Определим для конкретного класса характеристики, которые будем называть *полями*

Что такое поле?

- Определим для конкретного класса характеристики, которые будем называть *полями*
- Каждое поле будет иметь своё имя

Что такое поле?

- Определим для конкретного класса характеристики, которые будем называть *полями*
- Каждое поле будет иметь своё имя
- У экземпляра класса каждое поле имеет значение некоторого типа

Что такое поле?

- Определим для конкретного класса характеристики, которые будем называть *полями*
- Каждое поле будет иметь своё имя
- У экземпляра класса каждое поле имеет значение некоторого типа
- Есть специальное значение для полей, которое обозначает «отсутствие значения»

Пример: JavaBean

Пример: JavaBean

```
class Point {
    private Double x;
    private Double y;

    Point(Double x, Double y) { /*...*/ }
    // getters: getX, getY
}
Point point = new Point(1.0, 2.0);
```

Пример: JavaBean

```
class Point {
    private Double x;
    private Double y;

    Point(Double x, Double y) { /*...*/ }
    // getters: getX, getY
}
Point point = new Point(1.0, 2.0);
```

- Поля — то, что возвращают геттеры **getX**, **getY**

Пример: JavaBean

```
class Point {
    private Double x;
    private Double y;

    Point(Double x, Double y) { /*...*/ }
    // getters: getX, getY
}
Point point = new Point(1.0, 2.0);
```

- Поля — то, что возвращают геттеры **getX**, **getY**
- Названия полей — **x**, **y**

Пример: JavaBean

```
class Point {
    private Double x;
    private Double y;

    Point(Double x, Double y) { /*...*/ }
    // getters: getX, getY
}
Point point = new Point(1.0, 2.0);
```

- Поля — то, что возвращают геттеры **getX**, **getY**
- Названия полей — **x**, **y**
- Значения полей для объекта point — **1**, **2**

Пример: List

Пример: List

```
import java.util.List;
import java.util.Arrays;

List<Integer> list = Arrays.asList(42, 24);
```

Пример: List

```
import java.util.List;
import java.util.Arrays;

List<Integer> list = Arrays.asList(42, 24);
```

- Поля — то, что возвращает метод **get(i)**, **i = 0, 1, ...**

Пример: List

```
import java.util.List;
import java.util.Arrays;

List<Integer> list = Arrays.asList(42, 24);
```

- Поля — то, что возвращает метод **get(i)**, **i = 0, 1, ...**
- Названия полей — индексы, передаваемые в get

Пример: List

```
import java.util.List;
import java.util.Arrays;

List<Integer> list = Arrays.asList(42, 24);
```

- Поля — то, что возвращает метод **get(i)**, **i = 0, 1, ...**
- Названия полей — индексы, передаваемые в get
- Значения полей для объекта list — **42, 24, NO_VALUE, NO_VALUE, NO_VALUE, ...**

Пути полей

Пути полей

- Имеется класс, для которого определены поля

Пути полей

- Имеется класс, для которого определены поля
- Типы этих полей — классы с определёнными полями

Пути полей

- Имеется класс, для которого определены поля
- Типы этих полей — классы с определёнными полями
- Можно определить *пути полей* как в файловой системе

Пример: Пути полей

Пример: Пути полей

```
class Circle {
    private Point center;
    private Double radius;

    // getters: getCenter, getRadius
}
```

Пример: Пути полей

```
class Circle {
    private Point center;
    private Double radius;

    // getters: getCenter, getRadius
}
```

- Названия полей для Circle — **center**, **radius**

Пример: Пути полей

```
class Circle {
    private Point center;
    private Double radius;

    // getters: getCenter, getRadius
}
```

- Названия полей для Circle — **center, radius**
- Пути полей — **center/x, center/y, radius**

Содержание

Что такое Differ?

Что такое Differ?

- Differ — интерфейс с точки зрения Java

Что такое Differ?

- Differ — интерфейс с точки зрения Java
- Differ сравнивает два объекта с определёнными полями

Что такое Differ?

- Differ — интерфейс с точки зрения Java
- Differ сравнивает два объекта с определёнными полями
- Поля сравниваются «один-к-одному»

Что такое Differ?

- Differ — интерфейс с точки зрения Java
- Differ сравнивает два объекта с определёнными полями
- Поля сравниваются «один-к-одному»
- Differ выдаёт разницу между объектами

Пример: Differ

Пример: Differ

```
class PointDiffer implements Differ {
  @Override
  public List<Diff> compare(Object l, Object r) {
      /* ... */
  }

  // other methods
}
Differ differ = new PointDiffer();
Point p1 = new Point(1.0, 1.0);
Point p2 = new Point(2.0, 3.0);
differ.compare(p1, p2);
```

Пример: Differ

```
class PointDiffer implements Differ {
  @Override
  public List<Diff> compare(Object l, Object r) {
      /* ... */
  }

  // other methods
}
Differ differ = new PointDiffer();
Point p1 = new Point(1.0, 1.0);
Point p2 = new Point(2.0, 3.0);
differ.compare(p1, p2);
```

- Поле x отличается у объекта **p1** и **p2**

Пример: Differ

```
class PointDiffer implements Differ {
  @Override
  public List<Diff> compare(Object l, Object r) {
      /* ... */
  }

  // other methods
}
Differ differ = new PointDiffer();
Point p1 = new Point(1.0, 1.0);
Point p2 = new Point(2.0, 3.0);
differ.compare(p1, p2);
```

- Поле **x** отличается у объекта **p1** и **p2**
- Поле **y** отличается у объекта **p1** и **p2**

Содержание

Что такое Декораторы для Differ?

Что такое Декораторы для Differ?

- Реализует интерфейс Differ

Что такое Декораторы для Differ?

- Реализует интерфейс Differ
- Паттерн проектирования «Декоратор»

Что такое Декораторы для Differ?

- Реализует интерфейс Differ
- Паттерн проектирования «Декоратор»
- Наделяет Differ новыми возможностями

Что такое Декораторы для Differ?

- Реализует интерфейс Differ
- Паттерн проектирования «Декоратор»
- Наделяет Differ новыми возможностями
- Обрабатывает ситуации, когда параметры compare равны null или «значение отсутствует»

Реализованные декораторы

Реализованные декораторы

- AllFieldsDifferDecorator — сравнивать все поля объектов

Реализованные декораторы

- AllFieldsDifferDecorator — сравнивать все поля объектов
- OnlyFieldsDifferDecorator — только указанные поля

Реализованные декораторы

- AllFieldsDifferDecorator — сравнивать все поля объектов
- OnlyFieldsDifferDecorator — только указанные поля
- AllFieldsExceptDifferDecorator — все поля, кроме указанных

Реализованные декораторы

- AllFieldsDifferDecorator — сравнивать все поля объектов
- OnlyFieldsDifferDecorator — только указанные поля
- AllFieldsExceptDifferDecorator — все поля, кроме указанных
- OnlyExpectedFieldsDifferDecorator — только те, которые не null у ожидаемого значения

Содержание

Что такое стратегии сравнения?

Что такое стратегии сравнения?

- Стратегия сравнения — интерфейс CompareStrategy

Что такое стратегии сравнения?

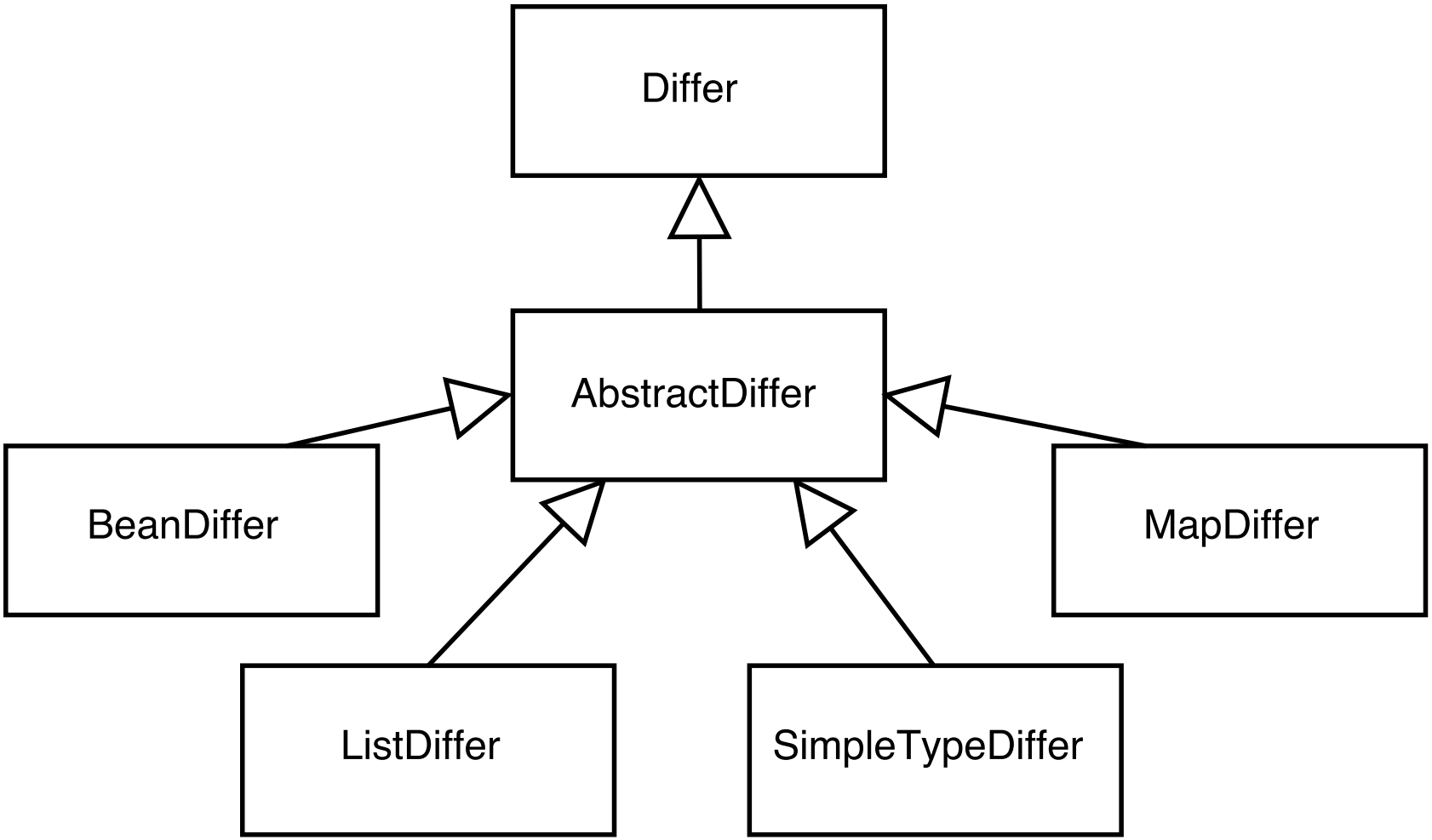
- Стратегия сравнения — интерфейс CompareStrategy
- «Какой Differ использовать для поля?»

Интерфейс CompareStrategy

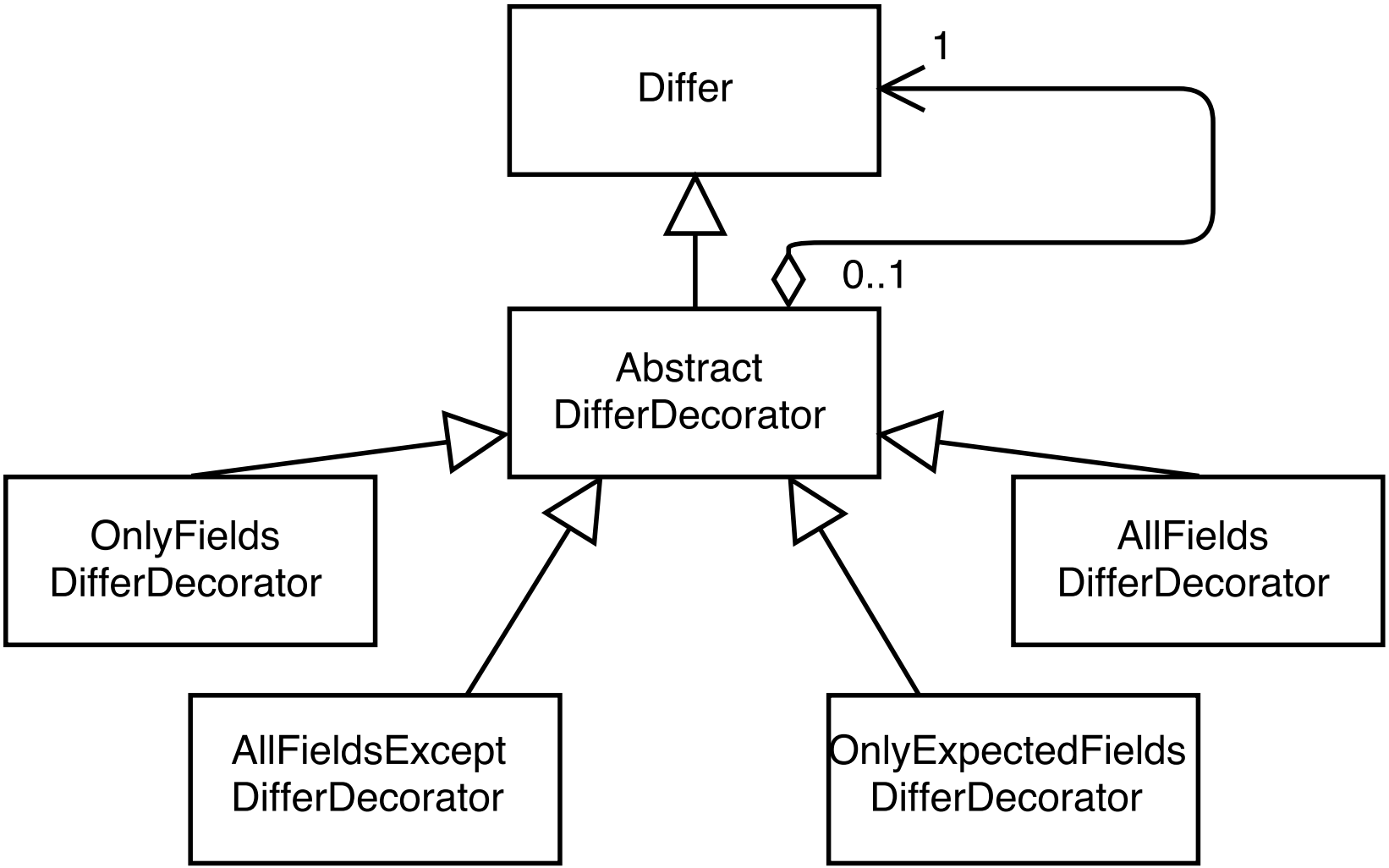
```
interface CompareStrategy {
    Differ getDefaultDiffer(BeanField field);

    default Differ getCustomDiffer(
            BeanField field) {
        return null;
    }

    default Differ getCustomOrDefaultDiffer(
            BeanField field) {
        Differ customDiffer =
            getCustomDiffer(field);
        if (customDiffer != null) {
            return customDiffer;
        }
        return getDefaultDiffer(field);
    }
}
```

Содержание

Иерархия Differ



Иерархия Декораторов



Классы стратегий

Классы стратегий

- DefaultCompareStrategy — стандартная стратегия

Классы стратегий

- DefaultCompareStrategy — стандартная стратегия
- AllFieldsDefaultCompareStrategy

Классы стратегий

- DefaultCompareStrategy — стандартная стратегия
- AllFieldsDefaultCompareStrategy
- AllFieldsExceptDefaultStrategy

Классы стратегий

- DefaultCompareStrategy — стандартная стратегия
- AllFieldsDefaultCompareStrategy
- AllFieldsExceptDefaultStrategy
- OnlyFieldsDefaultCompareStrategy

Классы стратегий

- DefaultCompareStrategy — стандартная стратегия
- AllFieldsDefaultCompareStrategy
- AllFieldsExceptDefaultStrategy
- OnlyFieldsDefaultCompareStrategy
- OnlyExpectedFieldsDefaultCompareStrategy

Другие классы

Другие классы

- BeanFieldPath — путь поля

Другие классы

- BeanFieldPath — путь поля
- BeanField — поле

Другие классы

- BeanFieldPath — путь поля
- BeanField — поле
- Diff — разница между полями

Содержание

Как реализовать Differ?

Как реализовать Differ?

- Следует реализовать метод compare

Как реализовать Differ?

- Следует реализовать метод compare
- Ситуации, когда параметры равны null или «значение отсутствует» обрабатывать не нужно

Как реализовать Differ?

- Следует реализовать метод compare
- Ситуации, когда параметры равны null или «значение отсутствует» обрабатывать не нужно
- Следует пройтись по всем полям и использовать для них Differ в соответствии со стратегией

Как реализовать Differ?

- Следует реализовать метод compare
- Ситуации, когда параметры равны null или «значение отсутствует» обрабатывать не нужно
- Следует пройтись по всем полям и использовать для них Differ в соответствии со стратегией

Как реализовать декоратор для Differ?

Как реализовать декоратор для Differ?

- Предполагается, что существующих декораторов хватает

Как реализовать декоратор для Differ?

- Предполагается, что существующих декораторов хватает
- Декоратор обязан корректно обрабатывать ситуации, когда одно из значений compare равно null

Как реализовать декоратор для Differ?

- Предполагается, что существующих декораторов хватает
- Декоратор обязан корректно обрабатывать ситуации, когда одно из значений compare равно null
- Декоратор обязан корректно обрабатывать ситуации, когда одно из значений «отсутствует»

Как реализовать декоратор для Differ?

- Предполагается, что существующих декораторов хватает
- Декоратор обязан корректно обрабатывать ситуации, когда одно из значений compare равно null
- Декоратор обязан корректно обрабатывать ситуации, когда одно из значений «отсутствует»

Как реализовать стратегию?

Как реализовать стратегию?

- Следует реализовать метод getDefaultDiffer

Как реализовать стратегию?

- Следует реализовать метод getDefaultDiffer
- Опционально: переопределить getCustomDiffer

Как реализовать стратегию?

- Следует реализовать метод getDefaultDiffer
- Опционально: переопределить getCustomDiffer
- Стратегия **обязана** возвращать Differ обёрнутый в декоратор

Как реализовать стратегию?

- Следует реализовать метод getDefaultDiffer
- Опционально: переопределить getCustomDiffer
- Стратегия **обязана** возвращать Differ обёрнутый в декоратор
- **Нельзя** переопределять getCustomOrDefaultDiffer

Содержание

Содержание