بررسی الگوهای طراحی– Flyweight

توسط : محمد حسین زارع
@zare88r: twitter
telegram
Join Us : @JavaLandCH

مفاهیم

- بهینه سازی در استفاده از حافظه
- تعداد زیادی از اشیاء مشابه به هم
- Immutable
- نمونه ها
  - javal.lang.String
  - java.land.Integer#valueOf(int)
  - Boolean , Byte , Character , Short , Long

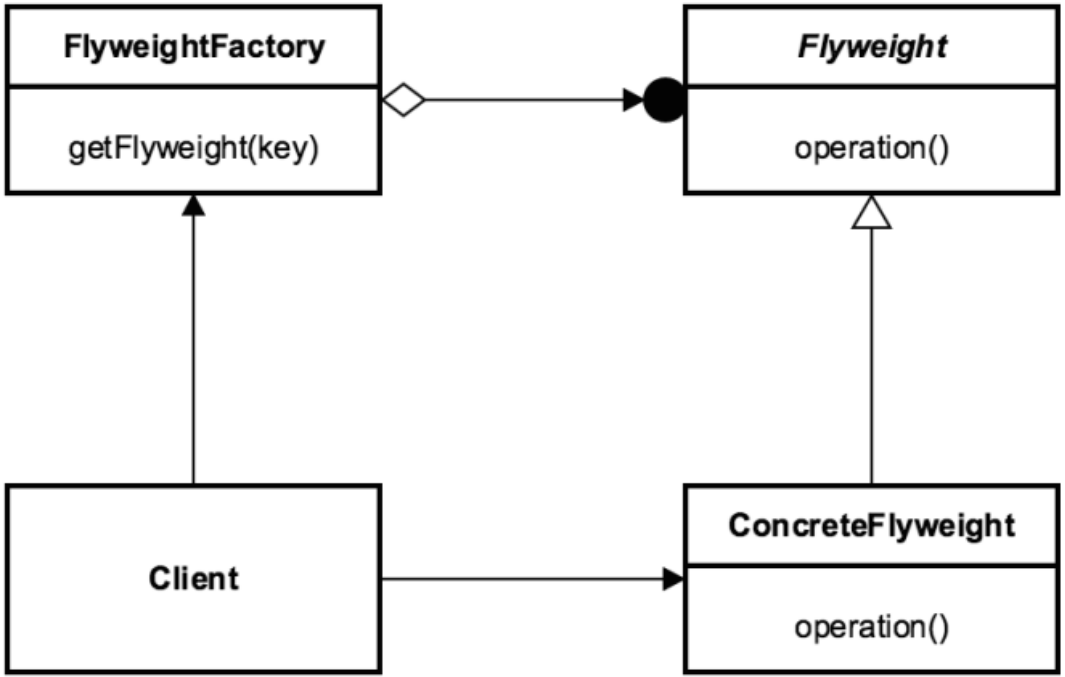
مفاهیم

Browser loads images just once and then reuses them from pool:

طراحی

- از یک Factory استفاده میکند
- این الگو میتواند در الگوی های دیگر نیز استفاده شود
- شامل :
- Client , Factory , Flyweight

طراحی


FlyweightFactory
getFlyweight(key)
Flyweight
operation()
Client
ConcreteFlyweight
operation()

مثال 1

```
String str1 = "sample";
String str2 = "sample";

System.out.println(System.identityHashCode(str1));
System.out.println(System.identityHashCode(str2));

Integer i1 = Integer.valueOf(100);
Integer i2 = Integer.valueOf(100);

System.out.println(System.identityHashCode(i1));
System.out.println(System.identityHashCode(i2));
```

یک نمونه

```
@Entity
@Table(name = "Departments")
public class Department {
    @Id
    private Integer departmentId;
    private String departmentName;
//getters and setters , hashcode , equals
//…
```

یک نمونه

```
@Service
public class DepartmentRepository {
    @PersistenceUnit
    private EntityManagerFactory emf;
    private List<Department> departmentCache = new ArrayList<>();

    public Department lookupDepartment(Integer departmentId) {
        Department tempData = new Department(departmentId);
        Department result;
        if (departmentCache.contains(tempData)) {
            result = departmentCache.get(departmentCache.indexOf(tempData));
        } else {
            EntityManager entityManager = emf.createEntityManager();
            result = entityManager.find(Department.class, departmentId);
            departmentCache.add(result);
            entityManager.close();
        }
        return result;
    }
}
```

یک نمونه

```
@Autowired
private DepartmentRepository departmentRepository;

@Test
public void contextLoads() {
    int hashCode1 = System.identityHashCode(
            departmentRepository.lookupDepartment(10));
    int hashCode2 = System.identityHashCode(
            departmentRepository.lookupDepartment(10));
    Assert.assertEquals(hashCode1, hashCode2);
    departmentRepository.lookupDepartment(20);
}
```

جمع بندی

- برای مدیریت حافظه راهکار بسیار مناسبی است
- کمی پیچیدگی دارد
- توسط API ها و فریم ورک ها بسیار مورد استفاده قرار میگیرد
- نیاز به دانستن Factory دارد