Inversion of Control (IoC)

Inversion of Control

- Exemplos
- Por que isso é bom?
- O que tem a ver com Dependency Injection?
- Refactoring da nossa TODO List

Definições

- Não é um Design Pattern, é um principio, uma técnica
- Hollywood prinicple: ”do not call us, we will call you”
- Utilizado na criação de frameworks
- Muitas vezes usamos IoC sem perceber

Exemplo: Window system

```
#ruby
puts 'What is your name?'
name = gets
process_name(name)
puts 'What is your quest?'
quest = gets
process_quest(quest)
```

```
require 'tk'
root = TkRoot.new()
name_label = TkLabel.new() {text "What is Your Name?"}
name_label.pack
name = TkEntry.new(root).pack
name.bind("FocusOut") {process_name(name)}
quest_label = TkLabel.new() {text "What is Your Quest?"}
quest_label.pack
quest = TkEntry.new(root).pack
quest.bind("FocusOut") {process_quest(quest)}
Tk.mainloop()
```

Exemplo: Junit setUp() e tearDown()

```java
public class IoCTest extends TestCase {

    protected void setUp() throws Exception {
        super.setUp();
    }

    public void testSetUpTearDown(){

    }

    protected void tearDown() throws Exception {
        super.tearDown();
    }

}
```

```java
public class IoCTest {

    @Before
    protected void setUp() throws Exception {
    }

    @Test
    public void testSetUpTearDown(){

    }

    @After
    protected void tearDown() throws Exception {
    }

}
```

Exemplo: criação de objetos

```
public class MovieLister {

    private MovieFinder finder;

    public MovieLister() {
        finder = new ColonDelimitedMovieFinder("movies1.txt");
      }

    @SuppressWarnings("unchecked")
    public Movie[] moviesDirectedBy(String arg) {
        List allMovies = finder.findAll();
        for (Iterator it = allMovies.iterator(); it.hasNext();) {
            Movie movie = (Movie) it.next();
            if (!movie.getDirector().equals(arg)) it.remove();
        }
        return (Movie[]) allMovies.toArray(new Movie[allMovies.size()]);
    }

}
```

```
public class MovieLister {

    private MovieFinder finder;

    public MovieLister(MovieFinder finder) {
        this.finder = finder;
    }
```

Dependency Injection

- É a inverção de controle aplicada na criação das dependências de objetos

Por que usar IoC?

- Retira o controle de certas atividades que não são de responsabilidade do ”pedaço” de software em questão
- Faz com que o sistema faça apenas o que ele foi projetado para fazer (coesão)
- Promove o desacoplamento
- Aumenta a testabilidade do código

TODO List

- Vamos aplicar o principio da inversão de controle na nossa aplicação exemplo
- Mas onde?
- Invertendo o controle de transação

Referências

- http://martinfowler.com/bliki/InversionOfControl.html
- http://www.laputan.org/drc/drc.html
- http://martinfowler.com/articles/injection.html
- http://www.objectmentor.com/resources/articles/dip.pdf