CONGRESSO CATARINENSE DE CIÊNCIA DA COMPUTAÇÃO
2019. Ano 6, Vol. 6. ISSN: 2319-0655

# VERIFICANDO DISPONIBILIDADE DE UM SITE E MUDANÇA NO CÓDIGO FONTE COM O FRAMEWORK ONCHANGE

**Willian Mauricio[1], Professor Me. Rodrigo Curvêllo[2]**
[1]Instituto Federal Catarinense – Campus Rio do Sul
Willian.mauricio.11@hotmail.com, rodrigo.curvello@ifc.edu.br

***Abstract.*** *This article describes a tool in which it is used to make checks on a particular site, such verification may be by availability of the site, ie if it is available or unavailable, or even check if there is any change in the source code. For this, a framework was developed (OnChange), that makes the queries about the site and also allowing a run time for such task. For the scheduling of tasks, an existing framework called Quartz Job Scheduler was imported, which was modularized to meet the requirements of the OnChange framework. In addition, classroom knowledge about the creative patterns of Object Oriented Programming was used.*

***Key-words:*** *Framework; OnChange; Job.*

***Resumo.*** *Este artigo descreve uma ferramenta na qual é utilizada para fazer verificações sobre um determinado site, tal verificação podendo ser por disponibilidade do site, ou seja, se ele está disponível ou indisponível, ou até mesmo verificar se há alguma mudança no código fonte do mesmo. Para isto, desenvolveu-se um framework (OnChange), que faz as consultas sobre o site e também possibilitando um tempo de execução para tal tarefa. Para o agendamento de tarefas, foi importado um framework existente denominado Quartz Job Scheduler, que foi modularizado para atender aos quesitos do framework OnChange. Além disso, foi utilizado os conhecimentos obtidos em sala de aula sobre os padrões criacionais da Programação Orientada a Objeto.*
***Palavras-chave:*** *Framework; OnChange; Tarefa.*

## 1. Introdução

Atualmente no âmbito do desenvolvimento de um software, procura-se facilitar e agilizar o processo de codificação ou implementação das regras de negócios desta determinada aplicação, de tal maneira que o programador/desenvolvedor não perca tempo programando algo que já é existente dentro do proposto pelo software.

Dessa forma, para que não haja essa perca de tempo no desenvolvimento do software, um dos conceitos básicos da programação que auxilia nesse processo de agilidade, é denominado de Framework.

Segundo Johnson & Foote (1988, p. 22), um framework é um conjunto de classes genéricas que constituem de forma abstrata, um projeto, cujo propósito é encontrar uma solução para uma cadeia de problemas. Para melhor entendimento, Mattsson (1996, p) diz que, é uma espécie de arquitetura na qual tem como objetivo atingir a reutilização máxima e com grande potencial de especialização.

Já para Bushmann et al (1996), um framework é uma aplicação parcialmente completa desenvolvida para ser instanciada em outro projeto, podendo herdar suas características e funções.

Com base nisso, o presente artigo visa descrever o desenvolvimento de um Framework cuja finalidade é verificar o estado em que um site ocupa, seja ele online/offline ou até mesmo se a mudanças no código fonte do mesmo.

## 2. Metodologia

Para o desenvolvimento do framework Onchange, cujo nome foi denominado pelo autor do trabalho, utilizou-se alguns paradigmas da programação orientada a objeto, sendo um deles os padrões de projetos criacionais.

Com os padrões de projetos foi possível realizar execução do Trigger e do Job que serão detalhados a seguir no tópico 2.2, para assim retornar a saída esperada na ferramenta.

| | | **Propósito** | | |
|---|---|---|---|---|
| | | **De criação** | **Estrutural** | **Comportamental** |
| **Escopo** | **Classe** | Factory Method | Adapter (class) | Interpreter<br>Template Method |
| | **Objeto** | Abstract Method<br>Builder<br>Prototype<br>Singleton | Adapter (object)<br>Bridge<br>Composite<br>Decorator<br>Façade<br>Flyweight<br>Proxy | Chain of Responsability<br>Command<br>Iterator<br>Mediator<br>Memento<br>Observer<br>State<br>Strategy<br>Visitor |

*Figura 1 - Tabela de padrões de projeto.*

A figura 1 demonstra os padrões de projetos elaborado pela Gang of Four (GoF), sendo eles criacionais, estruturais e comportamentais. Os padrões de criação são responsáveis por controlar como as classes serão criadas. Estruturais, esses se preocupam com a forma como classes e objetos são compostos para formar estruturas maiores.

Já os comportamentais, esses segundo Deitel (2005), "[...] fornecem estratégias testadas para modelar a maneira como os objetos colaboram entre si em um sistema e

oferecem comportamentos especiais apropriados para uma ampla variedade de aplicativos. [...]". No trabalho será visto mais a fundo os padrões de criação.

### 2.1. Padrões de Projeto Criacionais

É indiscutível que toda aplicação necessita a criação de instâncias de classes, mas se isto não for feito de forma controlada, o projeto tende a tornar-se complexo e mal estruturado, podendo acarretar na dificuldade para uma possível manutenção desse. Portanto, os padrões de projetos criacionais são técnicas que ajudam controlar como as instâncias de classes são criadas.

"Os padrões de criação voltados para classes deixam alguma parte da criação de objetos para subclasses, enquanto que os padrões de criação voltados para objetos postergam esse processo para outro objeto." (GAMMA *et al.*, 2007, p.26).

Esses padrões permitem que o usuário não instancie um objeto em específico dentro de uma classe principal (Cliente), mas sim instanciar uma classe abstrata ou subclasse, cuja mesma concede esse objeto para o programa, precisando apenas declarar o objeto da subclasse e chamar o método de criação.

### 2.2. Framework Onchange

Com base nos conceitos acima, elaborou-se a ferramenta que auxilia no processo de verificação da disponibilidade do site ou até mesmo se houve alguma alteração no código fonte do mesmo. Adjunto a verificação, foi implementado um framework conhecido por Quartz Job Scheduler que consiste em agendar as tarefas (verificações) em um determinado tempo obtido por parametrização pela classe cliente.

Sobretudo, esse framework necessita de classes como Trigger e Job, em que respectivamente, uma classe é responsável por configurar o agendamento de tarefas e disparar as mesmas através da criação pelo método "build();", e outra responsável por implementar as funcionalidades que serão executadas.


*Figura 2 - Logotipo Framework Quartz Job Scheduler.*

## 3. Análise e Desenvolvimento

Durante o desenvolvimento do trabalho, foram criadas diversas classes tais como Biblioteca.java que seria o framework propriamente dito, nela está contido todos os métodos necessários, e também é nela que chama-se as demais classes, como FactoryTipo.java, que possui uma regra de negócio em que faz a verificação do tipo para instanciar uma fábrica para testar a conexão com o site (FabricaTesta) ou a fábrica responsável para chamar a classe que verifica se houve mudanças no código fonte do site (FactoryMudanca).

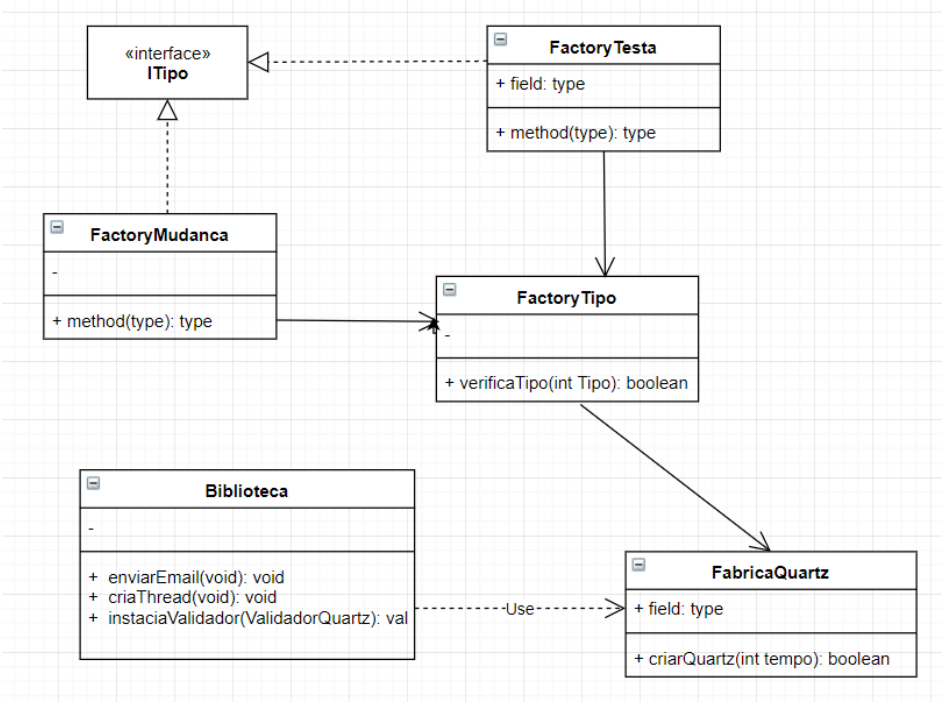


*Figura 3 - Diagrama de Classe framework.*

Além disso, foi criada uma classe para implementar a interface Job da ferramenta Quartz, para que nela se definiu o que seria executado. Para melhor entendimento, a figura 3 abaixo demonstra o código fonte desenvolvido para o framework.

```
1  package Quartz;
2
3  import javax.mail.MessagingException;
12
13 public class QuartzMain implements Job{
14
15     public void execute(JobExecutionContext arg0) throws JobExecutionException {
16
17         FactoryTipo tipo = new FactoryTipo();
18         boolean status = tipo.verificaTipo(ValidadorQuartz.getValidador().getTipo(),
19                 ValidadorQuartz.getValidador().getUrl());
20         System.out.println(status);
21         if(status) {
22             System.out.println("TUDO OK!");
23         } else {
24             try {
25                 Biblioteca.enviarEmail();
26             } catch (MessagingException e) {
27                 System.out.println("bli!");
28             }
29         }
30     }
31 }
```

*Figura 4 - Código fonte da Classe que implementa a interface Job.*

Como pode-se ver na figura 3, a classe QuartzMain.java está implementando a interface Job, que nela contém o método ”execute();” que é responsável por fazer a execução da tarefa defina pelo método “verificaTipo();” no qual verifica o tipo passado por parâmetro e por fim se retornar o valor booleano “false”, enviará o e-mail definido no método “enviarEmail();”.

```java
package Framework;
import Email.EnviaEmail;

public class Biblioteca {
    static private int tipoV;
    static private int cont;


    public static void enviarEmail() throws MessagingException {
        EnviaEmail email = new EnviaEmail();
        email.enviarEmail(tipoV);
    }

    public void criaThread(int tempo) {
        cont++;
        for(int i = 0; i <= cont; i++) {
            FactoryThread thread = new FactoryThread();
            thread.quartz(tempo);
            thread.run();
        }
    }

    public ValidadorQuartz instanciaValidador(int tipo, String url, int tempo) throws SchedulerException {
        tipoV = tipo;
        //System.out.println(tipoV);
        criaThread(tempo);
        ValidadorQuartz validador = new ValidadorQuartz(tipo,url);
        return validador;
    }
}
```

*Figura 5 - Classe do Framework contida os métodos de execução.*

Na figura 4 acima, tem-se a classe Biblioteca.java, cuja mesma consiste em armazenar os métodos que serão necessários para o funcionamento do Framework. Dentre os métodos, o principal é o "instaciaValidador();", que a partir do mesmo chama-se o método para criar uma instância do Quartz, cujo mesmo é "criaThread();", onde nele circunda todas as fábricas criadas para fazer a instância dos objetos.

## 4. Conclusão

O referido trabalho teve grande aprofundamento nos conhecimentos adquiridos na disciplina de Programação Orientada a Objetos II (POO II), cujo objetivo foi parcialmente alcançado, deixando possíveis estudos para dar continuidade no projeto, como tal fazer verificações em paralelo, ou seja, fazer os dois tipos de verificações tanto sobre disponibilidade do site quanto modificação no código fonte do mesmo, isso em tempos iguais ou até mesmo distintos.

Não obstante, esse projeto trouxe maior entendimento em relação a padrões de projetos e até mesmo a uma ferramenta (Quartz Job Scheduler) implementada e estudada para o desenvolvimento do framework Onchange com o intuito de agilizar o tempo na realização das tarefas em um determinado período.

## 5. Referências

BRASIL. PUCRJ., **Framework:** Conceitos Gerais. Disponível em: <https://www.maxwell.vrac.puc-rio.br/8623/8623_3.PDF>. Acesso em: 03 jul. 2019.

BUSCHMANN, F. et al., **Pattern-Oriented Software Architecture:** A System of Patterns. Chichester, UK: John Wiley & Sons. 1996.

DEITEL, Harvey M., **Java: como programar.** 6. ed. Brasil: Prentice Hall Brasil, 2005. 1110 p.

GAMMA, Erich et al. **Padrões de Projeto:** Soluções Reutilizáveis de Software Orientado a Objetos. Porto Alegre: Bookman, 2007. 364 p. Tradução de: Luiz A. Meirelles Salgado. Disponível em: <file:///C:/Users/willi/Downloads/Padroes_de_Projetos_-_Solucoes_Reutiliza.pdf>. Acesso em: 03 jul. 2019.

JOHNSON, Ralph E., FOOTE, Brian. Designing reusable classes. **Journal of Object-Oriented Programming**, 1988.

MATTSON, Michael. **Object-Oriented Frameworks – A survey of methodological issues.** 1996. 130p.