

# Proposta de arquitetura para aplicações que utilizam framework JSF

Arthur Pereira Gregório, André Roberto Ortoncelli

<sup>1</sup>Universidade Tecnológica Federal do Paraná - UTFPR/CP  
Avenida Alberto Carazzai, 1640 - CEP: 86300-000  
Cornélio Procópio-PR, Brazil

<sup>2</sup>Departamento de Computação

arthurshakal@gmail.com, ortoncelli@utfpr.edu.br

**Abstract.** *This article aims to present an architectural proposal for applications using JSF framework. In it we will see which key technologies we can use to build a simple architecture to maintain and fully standardized with the Java platform specifications Enthereprise Edition 7*

**Resumo.** *Este artigo tem por objetivo apresentar um proposta de arquitetura para aplicações que utilizam o framework JSF. Nele veremos quais a principais tecnologias podemos usar para construir uma arquitetura simples de manter e totalmente padronizada com as especificações da plataforma Java Enthereprise Edition versão 7.*

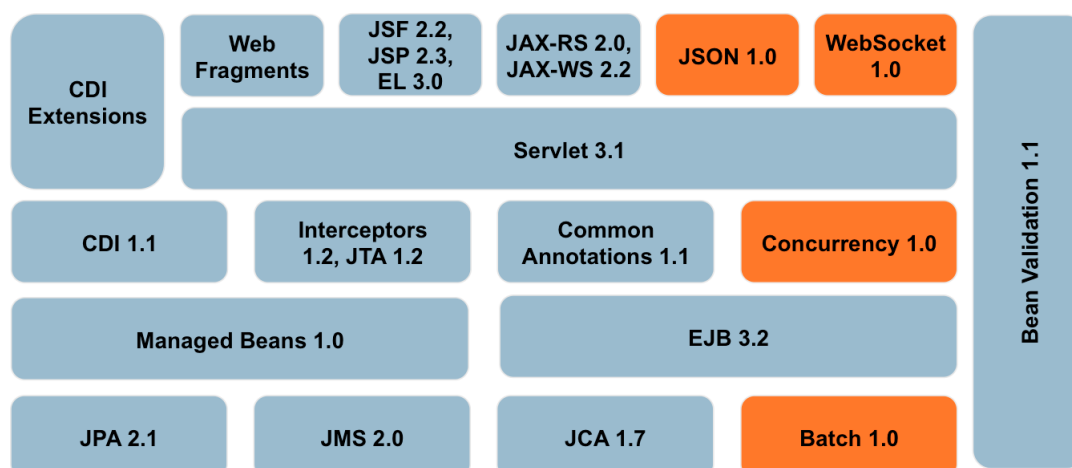
## 1. Introdução

Segundo pesquisa realizada pelo instituto Tiobe (2015) em setembro de 2015, Java é a linguagem de programação mais utilizada no mundo. A linguagem tornou-se a escolha mais frequente entre as equipes de desenvolvimento de software e acaba também por seguir as tendências do mercado atual, onde segundo Macoratti (2013) temos que projetar um software fácil de ajustar e barato de manter.

Segundo Xavier (2013) a definição de uma arquitetura se inicia pela seleção dos elementos que compõem a estrutura e como estes vão se relacionar, assim podemos dizer que a criação de uma arquitetura de software é algo que exige do arquiteto uma série de conhecimentos específicos sobre como estes componentes funcionam ou como podem interagir entre si.

Criar uma arquitetura não é algo simples, pois mais importante do que saber integrar os componentes selecionados, é saber identificar quais as possíveis escolhas e quais os fatores que influenciam nestas escolhas (XAVIER, 2013).

Atualmente em aplicações web temos de atingir uma série de requisitos funcionais através de nossas regras de negócios e ainda lidar com os requisitos não funcionais atendidos por nossa infraestrutura, proveniente da arquitetura na qual o sistema foi constituído. *Java Entherprise Edition (JEE)* consiste de uma série de especificações bem detalhadas, dando uma receita de como deve ser implementado um software que faz cada um desses serviços de infraestrutura (CAELUM, 2015). A figura 1 apresenta todos os componentes da plataforma JEE presentes na versão 7.



**Figura 1. Componentes do framework JEE**

Seguindo a idéia de se utilizar uma especificação padronizada e amplamente difundida como o JEE, diversos arquitetos de software hoje vem escolhendo o pacote de frameworks JEE para criação de suas aplicações. Dentre as opções disponíveis para uso dentro da plataforma, estão as especificações de persistência em banco de dados, transação, acesso remoto, web services, gerenciamento de threads, gerenciamento de conexões HTTP, cache de objetos, gerenciamento da sessão web, balanceamento de carga, entre outros.

O JSF, uma tecnologia integrante da especificação JEE incorpora características de um framework MVC (model-view-controller) para WEB e de um modelo de interfaces gráficas baseado em eventos (PITANGA, 2015) serve como base para boa parte das aplicações Java que rodam em ambiente web.

De acordo com Coelho (2013) JSF é uma tecnologia muito útil e prática de ser aplicada, porém, muitas vezes mal utilizada pois não está alinhada a uma estrutura de qualidade ou por falta de conhecimento de quem estruturou a aplicação.

Assim, nas próximas seções será apresentada uma proposta de arquitetura padronizada através da especificação JavaEE para aplicações que utilizam a tecnologia JSF a fim de minimizar os impactos com escolhas erradas de frameworks que possam influenciar na qualidade da sistema a ser construído.

## 2. Trabalhos Relacionados

Difícilmente uma empresa terá recursos suficientes para competir com a comunidade mundial. Em outras palavras, o arquiteto de software de uma corporação deve conhecer o máximo possível das opções de componentes e frameworks existentes no mercado para não cair no velho e já conhecido paradigma de tentar reinventar a roda (FRANZINI, 2015).

Deste modo, conhecer os principais componentes da arquitetura JavaEE antes de iniciar seu uso é indispensável para um arquiteto obter sucesso na tarefa de criar uma arquitetura duradoura.

## 2.1. Servlets

A API de servlets é um dos módulos presentes na plataforma JavaEE que é executado em um servidor web possibilitando a aplicação que o estiver executando, receber e responder requisições através do protocolo HTTP, o *HyperText Transfer Protocol* (ORACLE, 2011).

## 2.2. Context and Dependency Injection

Segundo Lopes (2012), *Context and Dependency Injection* ou CDI é a parte da especificação do JavaEE que cuida do controle e injeção dependências em nossos projetos, e caso a tecnologia de apresentação em uso seja o JSF versão 2, utilizar CDI é algo natural tendo em vista a integração entre as ferramentas.

## 2.3. Java Persistence API

JPA é um framework leve, baseado em *Plain Old Java Objects* - POJOS - para persistir objetos Java (MEDEIROS, 2006). A Java Persistence API, não é apenas um framework para Mapeamento Objeto-Relacional ela também oferece integração direta com outros frameworks da tecnologia JavaEE, como por exemplo a API de Beans Validation que será descrita a seguir.

## 2.4. Bean Validation

*Bean Validation* é a parte do framework Java responsável por validar objetos, membros deste objetos, métodos e construtores. Dentro de ambientes JavaEE, a API de *Bean Validation* oferece ainda integração com outros serviços disponíveis na arquitetura como por exemplo a API de JPA (DURAND; KRAFFMILLER, 2014).

# 3. Desenvolvimento

Nesta seção será abordada a integração entre as tecnologias previamente apresentadas na elaboração de um arquitetura totalmente compatível com o padrão especificado pelas *Java Specifications Requests (JSR)* integrantes do *framework JEE*.

## 3.1. Estrutura

A estrutura da arquitetura segue uma organização em camadas conforme a proposta do *Model Driven Design (MDD)* apresentado por Eric Evans em seu livro sobre *Domain Driven Design (DDD)*, que basicamente estabelece um conjunto de práticas para a construção de um software que expresse de maneira clara o problema em questão (EVANS, )

### 3.1.1. MDD

Segundo Durand e Kraffmiller (2010) o MDD divide as estruturas do software em quatro camadas básicas:

- Camada de interface com o usuário, que como o próprio nome sugere, é responsável pela parte de interação com as ações do usuário
- Camada de aplicação, realiza a conexão entre a camada de usuário e as camadas inferiores, não deve possuir regras de negócio

- Camada de Domínio, a representação dos conceitos, regras e lógicas do software, ou seja, onde o DDD se aplica
- Camada de Infra-estrutura, é o apoio para as camadas superiores, aqui encontram-se métodos para acesso a dados, envio de emails entre outros

#### **4. Conclusão**

## Referências

- CAELUM. **Java para desenvolvimento web**. [S.l.]: Caelum, 2015.
- COELHO, H. **JSF Eficaz**. [S.l.]: Casa do Código, 2013.
- DURAND, G.; KRAFFMILLER, S. **DDD: Introducao a Domain Driven Design**. Daniel Cukier, 2010. Disponível em: <<http://www.agileandart.com/2010/07/16/ddd-introducao-a-domain-driven-design/>>. Acesso em: 19 de dezembro de 2015.
- DURAND, G.; KRAFFMILLER, S. **Bean Validation: Practical Examples from a Real-World Java EE 7 Application**. Harvard University, 2014. Disponível em: <<http://datascience.iq.harvard.edu/presentations/bean-validation-practical-examples-real-world-java-ee-7-application>>. Acesso em: 24 de setembro de 2015.
- EVANS, E. **Domain Driven Design: Atacando as Complexidades no Coracao do Software**. [S.l.]: Alta Books. ISBN 9788576085041.
- FRANZINI, F. **Listagem de frameworks Java**. iMasters, 2015. Disponível em: <<http://imasters.com.br/linguagens/java/listagem-de-frameworks-java/>>. Acesso em: 22 de setembro de 2015.
- LOPES, S. **Use CDI no seu próximo projeto Java**. Caelum, 2012. Disponível em: <<http://blog.caelum.com.br/use-cdi-no-seu-proximo-projeto-java/>>. Acesso em: 24 de setembro de 2015.
- MACORATTI, J. C. **.NET - Definindo a arquitetura de um projeto de software**. iMasters, 2013. Disponível em: <<http://imasters.com.br/framework/dotnet/net-definindo-a-arquitetura-de-um-projeto-de-software/>>. Acesso em: 20 de setembro de 2015.
- MEDEIROS, H. **Introdução a JPA - Java Persistence API**. Devmedia, 2006. Disponível em: <<http://www.devmedia.com.br/introducao-a-jpa-java-persistence-api/28173>>. Acesso em: 24 de setembro de 2015.
- ORACLE. **JavaEE 6 Docs**. Oracle Corporation, 2011. Disponível em: <<http://docs.oracle.com/javae/6/api/javax/servlet/Servlet.html>>. Acesso em: 23 de setembro de 2015.
- PITANGA, T. **JavaServer Faces: A mais nova tecnologia Java para desenvolvimento WEB**. Grupo de Usuários Java - GUJ, 2015. Disponível em: <<http://www.guj.com.br/content/articles/jsf/jsf.pdf>>. Acesso em: 22 de setembro de 2015.
- TIOBE. **TIOBE Programming Community Index**. Tiobe, 2015. Disponível em: <<http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>>. Acesso em: 20 de setembro de 2015.
- XAVIER, K. **O que é arquitetura de software**. Globalcode, 2013. Disponível em: <<http://blog.globalcode.com.br/2013/11/o-que-e-arquitetura-de-software.html>>. Acesso em: 20 de setembro de 2015.