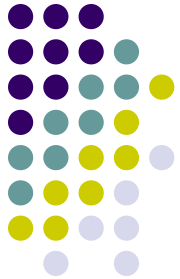
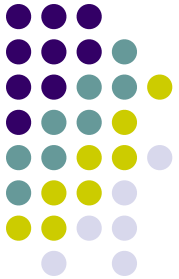


Aula 02: Frameworks

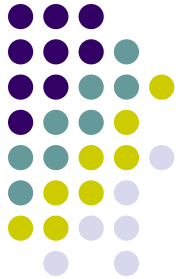
Prof. Luís Henrique Ries

Sumário



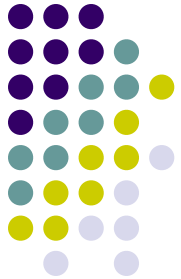
- Frameworks - Introdução
  - Por que utilizar?
  - Definição
  - Características
  - Framework x Biblioteca
  - Benefícios e Desafios
- Estudo de Caso: JUnit
- Estudo de Caso: Jest
- Discussão

Por que utilizar?



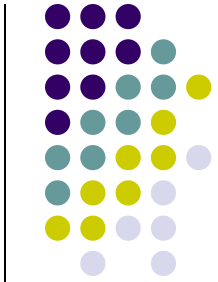
- Desenvolver software é complexo
  - Pensar em abstrair o problema e desenvolver todas as funcionalidades do zero é complexo.
    - Desenvolvimento de funcionalidades
    - Projeto do sistema como um todo
  - O reuso é a solução
  - Algumas formas de reuso:
    - Biblioteca: só funcionalidade
    - Framework: funcionalidade e concepção do projeto.
      - Reuso de código e design

Definição



- Frameworks
  - Voltados para solução em funcionalidades comum a várias aplicações
  - São voltados para tratar aplicações que possuem uma série de problemas semelhantes
  - Suas classes são flexíveis e extensíveis:
    - Facilita a construção de aplicações com pouco esforço
    - Especifica-se apenas as particularidades de cada aplicação

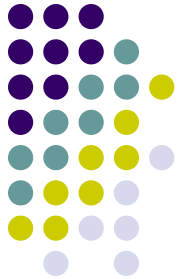
Definição (formal)



- Gamma: "A framework is a set of cooperating classes that make up a reusable design for a specific class of software"

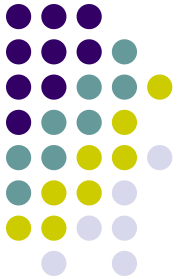
- Taligent: "A framework is an extensible set of object-oriented classes that are integrated to execute well-defined sets of computing behavior“

Definição (formal)



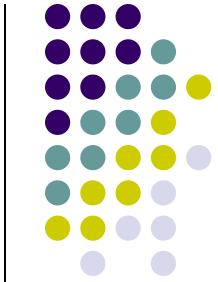
- Fowler: “A framework embodies some abstract design, with more behavior built in. In order to use it you need to insert your behavior into various places in the framework either by subclassing or by plugging in your own classes. The framework's code then calls your code at these points.”

Características



- Modular
  - Divisão em módulos facilmente acoplados, ou seja, facilitar a integração.
- Reusável
  - É sua grande proposta. Por isso deve ser bem documentado e fácil de usar.
- Extensível
  - Caracterizado por funcionalidades abstratas onde o programador deve “completar”.

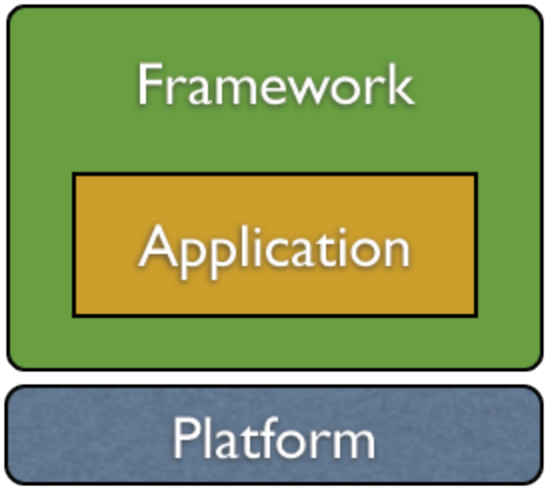
Características

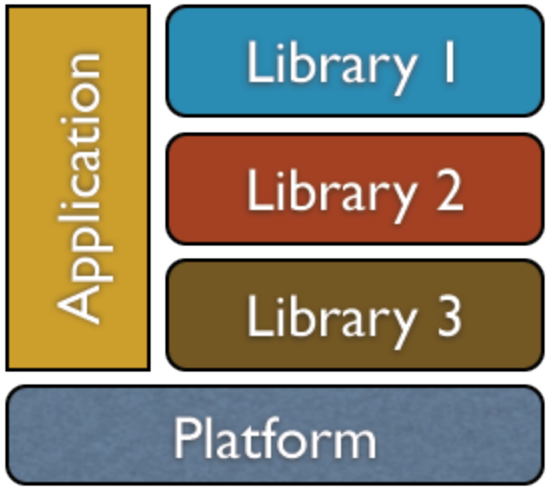


- Completo
  - Deve ser voltado para tratar domínio do problema pretendido
- Seguro
  - Não deve permitir que o desenvolvedor destrua suas funcionalidades e padrões.
- Inversão de Controle (IoC)
  - Framework é o responsável pelo fluxo de controle.

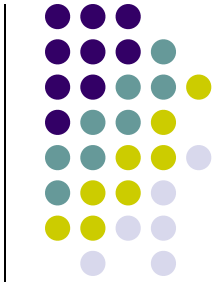
Framework x Biblioteca

- Diferença





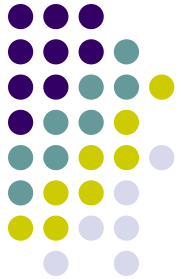
Framework x Biblioteca



- Biblioteca:
  - Aplicação faz uso da aplicação para funcionalidades de propósito geral.
  - Reuso de funcionalidades

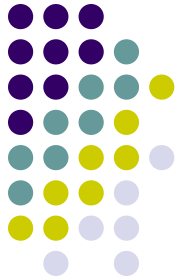
- Framework:
  - Inversão de controle
  - Reuso de funcionalidades e design.

Benefícios e Desafios



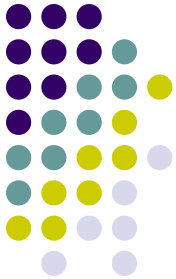
- Benefícios:
  - Redução de custos
  - Redução de Time-to-Market
  - Melhor compatibilidade entre aplicações
- Motivos:
  - Maximização do reuso (análise, projeto, desenv.)
  - Desenvolvedor não precisam “reinventar a roda”
  - Maior estabilidade e consistência

Benefícios e Desafios



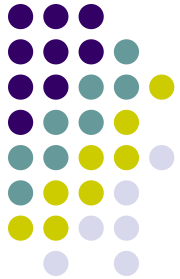
- Desafios:
  - Desenvolver um framework é complexo
  - Entendimento do framework pode demandar tempo de aprendizagem.
  - Aplicações usando frameworks podem apresentar um desempenho inferior a aplicações específicas
  - Frameworks devem ser mantidos e evolutivos.

Estudo de Caso 1: JUnit



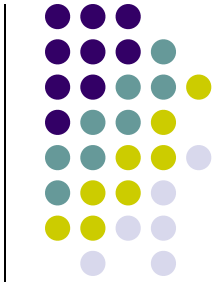
- Documentação do JUnit:
  - https://junit.org/junit5/
- Leitura – Uso do JUnit:
  - http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/junit.htm
- Demonstração

Estudo de Caso 2: Jest



- Documentação do JUnit:
  - https://jestjs.io/pt-BR/
- Leitura – Uso do Jest:
  - https://www.luiztools.com.br/post/tdd-como-criar-unit-tests-em-node-js-com-jest/
- Demonstração

Discussão



- Quais vantagens tivemos ao trabalhar com o JUnit/Jest?
- Quais são as características que denotam o JUnit/Jest como framework?
- Qual a importância de trabalhar com frameworks (geral)?