

# Integração Contínua

Implementando e utilizando a  
Integração Contínua para melhorar a  
qualidade do software

# Sobre o Palestrante

- Rodrigo Urubatan - SCJP 1.4 e SCWCD
- Trabalha com arquitetura de sistemas J2EE e treinamento
- Já desenvolveu projetos utilizando as linguagens Delphi, C++, PHP, ASP, ColdFusion, Leather, Assembly, Perl, ...
- Trabalha com Java/J2EE a 4 anos e com desenvolvimento de sistemas a 9 anos
- Atualmente colabora com pequenas correções a alguns projetos Open Source como o GUI2, Lombos e Veloclipse e faz parte da coordenação do RSJUG
- Já ministrou palestras em universidades (UCS, ULBRA, UNISC) e diversos eventos (Just Java, FISL, Seminário do RSJUG, Maratona 4 Java, Infosul) tutoriais para o RSJUG e já teve um artigo publicado na revista Mundo Java
- Atualmente trabalha como consultor na AdvancedIT, como gerente de tecnologia e qualidade na Tech Office IT, e ministra cursos e alguns pequenos projetos pela USI Informática
- É o principal desenvolvedor do projeto Spring-Annotation

# Problemas do dia a dia

- Equipes trabalhando no desenvolvimento
  - Todos trabalhando no mesmo código fonte
  - Processo de build complexo
  - Qualquer um pode “quebrar o processo”
  - Sempre ocorrem problemas na integração do trabalho da equipe
- Muito tempo entre o desenvolvimento e o teste
- Mais tempo ainda para o cliente poder ver os resultados do trabalho

# Vantagens da Integração Contínua

- Problemas de integração são detectados no mesmo dia em que são criados
- Aviso rápido de código quebrado ou incompatível
- Testes unitários para todas as alterações são executados imediatamente
- Disponibilidade constante da versão “atual” para testes ou demo
- O Impacto imediato da integração de código incompleto ou com erros é os desenvolvedores aprenderem a trabalhar incrementalmente com ciclos menores de feedback.

# Passo 1 para integração contínua

- Manter todo o código fonte do projeto em um único lugar
  - CVS
  - Subversion
  - Perforce
  - ...
- Considerações escolhendo a ferramenta
  - Suporte da IDE utilizada
  - Integração com o sistema de build
  - Administração

# Let`s Play



# Passo 2 para Integração Contínua

- Automatização do Build
  - Java
    - ANT
    - Maven 2
  - .NET
    - Nant
    - MSBuild
  - Genérico
    - Make file

# Let`s Play



# Passo 3 para Integração Contínua

- Faça o seu build auto testável
  - (X)Unit
  - Selenium
  - FIT
  - (X)Mock
  - Muitos outros dependendo do seu ambiente

# Let`s Play



# Passo 4 para Integração Contínua

- Check-In de código diário
  - Integração é um meio de comunicação
  - Comunicação rápida e eficiente ajuda na melhoria da qualidade
  - Integrando o código frequentemente os desenvolvedores ficam sabendo rapidamente de alterações ou incompatibilidade criadas por outros desenvolvedores ou pelo próprio código

# Let`s Play



# Passo 5 para Integração Contínua

- Todo commit deve gerar um build em um servidor de integração contínua
  - Assim todos os testes serão executados
  - Uma nova versão será disponibilizada para testes e demonstrações
  - Escolha de uma ferramenta (...)

# Mantenha o build rápido

- A principal vantagem da integração continua é feedback rápido
- Nada atrapalha mais isto do que um build muito complexo e demorado

# Teste em um “clone” do ambiente de produção

- Todos sabemos que por mais que informática seja uma ciência “exata” o ambiente em que um software esta sendo executado interfere em seu comportamento
- Por isto a integração continua deve fazer um deploy do sistema em um clone do ambiente de produção

# Torne fácil para qualquer um obter a ultima versão “executável”

- Em uma aplicação WEB todos os envolvidos no processo devem ter acesso ao servidor de testes ou ao ultimo deployment gerado
- Para uma aplicação desktop todos devem ter um modo fácil de obter o ultimo executável
- Isto facilita os testes e as demonstrações, tornando o feedback do cliente mais rapido

# Automatize o deployment

- Para tornar possível o item anterior o deployment da aplicação para o servidor de testes deve ser automatizado
- Isto também facilita com que o ambiente de testes seja um clone do ambiente de produção, pois o processo de deployment vai ser igual
- E evita problemas no software causado por deployments incorretos

# Escolha da ferramenta

- Suporte a linguagem utilizada
- Suporte ao sistema de build utilizado
- Integração com ferramenta
- Sistemas de monitoramento disponíveis

# Escolha da Ferramenta

- Ant Hill Pro
- Build Forge
- Cruise Control
- Cruise Control.NET
- Damage Control
- Team City
- Continuum
- Luntbuild
- Gump

# Cruise Control

- Disponibilizado como Open Source pela Thought Works
- Implementado em Java
- Administração Desktop
- Interface um pouco confusa
- Suporte a Java apenas, mas com uma versão dedicada a .NET

# Continuum

- Open Source desenvolvido pela Apache Foundation
- Desenvolvido em Java
- Administração via WEB
- Suporte nativo a ANT, Maven 1 e 2 e Shell Script para outras linguagens

# Team City

- Comercial desenvolvido pela JetBrains
- Desenvolvido em Java
- Build Distribuído
- Suporte a Java, .NET,

# Exemplos

- Projeto Simples em Java
- Configuração do Team City
- Configuração do Continuum
- Configuração de notificações

# Let`s Play



# Referencias

- <http://www.martinfowler.com/articles/continuousIntegration.html>
- <http://damagecontrol.codehaus.org/Continuous+Integration+Server+Feature+Matrix>
- <http://www.jetbrains.com/teamcity/>
- <http://cruisecontrol.sourceforge.net/>
- <http://maven.apache.org/continuum/>
- <http://blog.urubatan.com.br>