Noções de Anti-padrões

Marum Simão Filho

marumsimao@gmail.com

Agenda

- Conceito
- Tipos
- Descrição
- Por que anti-padrões?
- Refactoring
- Exemplos de anti-padrões na Plafatorma JEE

Conceito

- Anti-padrão é a aplicação, repetidas vezes, de código ou projeto (*design*) que leva a um resultado negativo.
- Resultado negativo
  - Baixa performance
  - Código difícil de manter
  - Falha total do projeto

Tipos de Anti-padrões

- Aqueles que descrevem uma solução para um problema que resultou em uma situação ruim;
- Aqueles que descrevem como escapar de uma situação ruim e como proceder para, a partir dela, atingir uma boa solução.

Descrevendo Anti-padrões

- Uma forma de descrever anti-padrões é através de um *template*.
  - Nome
  - Forma Geral do Anti-padrão
  - Causas Típicas
  - Sintomas e Consequências
  - Solução Reconstrução (*Refactoring*)
  - Exceções Conhecidas

Por que anti-padrões?

- Anti-padrões acontecem principalmente pelas seguintes razões:
  - Inexperiência
  - Código ilegível
  - Desenvolvimento usando *copy-e-paste*

Por que anti-padrões?

- Inexperiência
  - É a maior causa dos anti-padrões.
  - Acontece principalmente quando do uso de novas tecnologias.
  - Na plataforma Web, o perigo é maior ainda:
    - Evolui rapidamente;
    - Muitos tutoriais e ferramentas de geração de código são cheias de anti-padrões.

Por que anti-padrões?

- Inexperiência
  - Como combater:
    - treinamento e qualificação das equipes,
    - revisão de código por parte dos desenvolvedores seniores, e
    - entender e reconhecer os padrões é a chave para evitá-los.

Por que anti-padrões?

- Código ilegível
  - O foco do programador volta-se para entender **o que** o código faz, e não **como** ele faz.
  - Assim, muitos anti-padrões conhecidos acabam passando desapercebidos.
  - Os desenvolvedores muitas vezes buscam concisão e otimização, negligenciando os custos futuros de manutenção.

Por que anti-padrões?

- Código ilegível
  - Como combater:
    - publicar e reforçar o uso e leitura de boas práticas de codificação,
    - usar comentários sempre que possível (documentação de código), e
    - política de revisão de código agressiva.

Por que anti-padrões?

- Desenvolvimento usando copy-e-paste
  - Código que é copiado de um lugar e colocado em outro, sendo, por vezes, considerado uma forma de reuso.
  - Leva a código menos confiável, uma vez que o trecho de código copiado é usado fora de seu contexto original.
  - Pior ainda: correções de erros não se propagam automaticamente no código copiado.
- Como combater:
  - usar herança;
  - classes utilitárias separadas.

Refatoração (*Refactoring*)

- É uma maneira disciplinada de transformar a implementação de código, de modo a fazer o projeto (*design*) melhor, sem alterar o comportamento visível externamente.
- A chave para refatorações de sucesso é ter um bom conjunto de testes de unidade.
- Também conhecimento como Reconstrução.

Refatoração (*Refactoring*)

- Envolve:
  - Motivação: o porquê de refatorar.
  - Exemplo: trecho de código ou diagrama onde se identifica o código a ser reconstruído (geralmente um anti-padrão).
  - Diagramas UML da refatoração (antes e depois)
  - Mecanismos: como migrar o código ruim para o novo código.

Catálogo de Padrões JEE (Dudney et al)

- Distribuição e Escalabilidade – 5
- Persistência – 4
- Arquitetura Baseada em Serviço – 4
- Uso e Abuso de JSP – 6
- Servlet – 5
- Entity Beans – 6
- Session Beans – 6
- Message-driven Beans – 3
- Web Services – 7
- J2EE Services – 7

Exemplos de Anti-padrões na Plafatorma JEE

Multiservice

- Um *multiservice* é um serviço que possui um grande número de métodos da interface pública que implementam processos relacionados a várias diferentes abstrações ou entidades.
- O resultado é um serviço que é acoplado a muitos componentes técnicos ou de negócio, e que também é utilizado por muitos clientes, tornando difícil o desenvolvimento e a manutenção, e reduzindo a flexibilidade e o reuso.

Tiny Service

- Um *tiny service* é um serviço que representa, de forma incompleta, uma abstração, implementando somente um subconjunto dos métodos necessários, provocando a necessidade de ter múltiplos serviços para implementarem completamente a abstração.
- O resultado é que os clientes têm que utilizar diversos serviços para executarem um processo de negócio significativo, e precisam implementar a sequenciação e o fluxo de atividades.

Ignoring Reality

- Este anti-padrão aborda a tendência dos desenvolvedores de ignorarem a realidade de que erros acontecem.
- Frequentemente, usuários se deparam com erros (*stack trace*) do Java, o que faz com que percam a confiança na aplicação, e se o problema não for resolvido, pode acontecer até de o projeto ser cancelado.

Too Much Code

- Este anti-padrão versa sobre a grande quantidade de código Java colocada em páginas JSP, o que acontece quando os desenvolvedores ou não conhecem ou não aplicam o princípio da separação lógica de tarefas entre as camadas JEE.
- A principal consequência é uma interface ao usuário muito frágil e difícil de manter.

Copy and Paste JSP

- Um hábito comum aos desenvolvedores é o de copiar e colar trechos de código entre páginas JSP.
- Quanto este anti-padrão acontece, o resultado típico é um sistema difícil de manter, com sutis diferenças entre as páginas, que tendem a confundir o usuário.

Too Much Data in Session

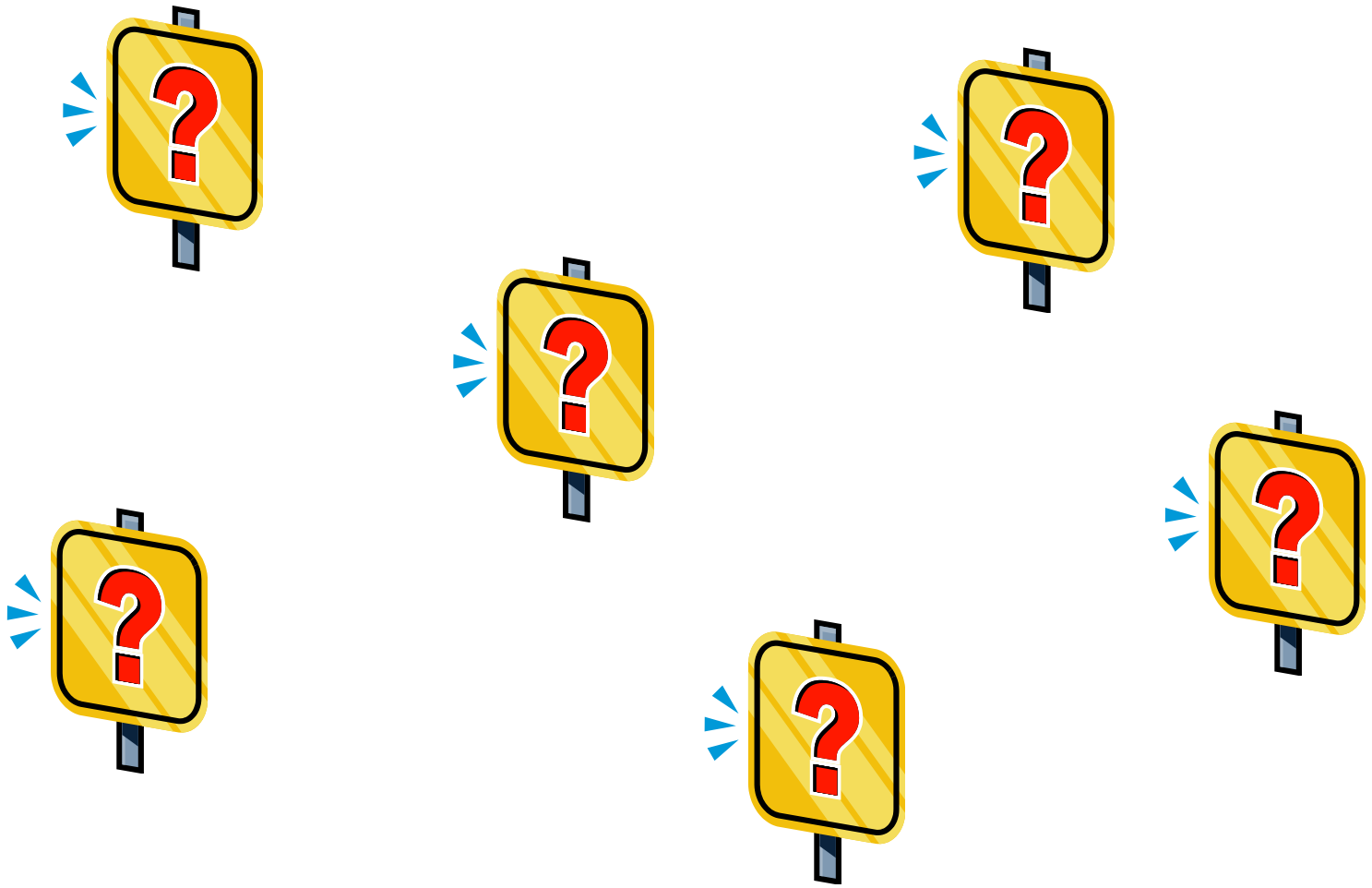
- Este anti-padrão aborda a tendência de os desenvolvedores usarem a sessão como um repositório gigante de dados para guardar tudo que eles precisam para a aplicação.
- O resultado é a ocorrência frequente de falhas e exceções aparentemente sem sentido.
- Isto acontece porque os dados são colocados na sessão, às vezes com chaves conflitantes, e até mesmo tipos de diferentes.

Template Text in Servlet

- Este anti-padrão refere-se à tendência de alguns desenvolvedores de codificarem (*hard-code*) strings de HTML dentro dos servlets.
- Desenvolvedores cujas aplicações estão presas a este anti-padrão precisarão de um bom tempo para corrigir os erros no HTML que é gerado porque este HTML é embutido em código Java ao invés de um arquivo HTML próprio.

Referências

- J2EE AntiPatterns. Bill Dudney et al. Wiley Publishing, Inc.
- Bitter Java. Bruce Tate. Manning.
- J2EE Design Patterns. William Crawford e Jonathan Kaplan. O'Reilly.
- Transparent Façade. David de Almeida Ferreira e Karine Roberta Vieira. http://davidferreirafz.wordpress.com



Obrigado!!!

Faculdade 7 de Setembro

marumsimao@gmail.com