Tiger: Funcionalidades do Java 5, a nova versão da plataforma Java

Summa Technologies

www.summa-tech.com

Sobre o palestrante

- JSR Community Manager @ java.net
- Expert na JSR 207 (PD4J)
- Thinlet commiter
- Contributor do AspectWerkz
- Membro da organização do SouJava
- SCPJ 1.2 & 1.4, SCWCD, SCMAD
- 5 anos de experiência com Java
- Palestrante no JavaOne 2003, Abaporu, JustJava, Javali e COMDEX

Tiger - Revolução?

- Por que tanta expectativa?
  - Novas features na linguagem
  - Mudanças de design em classes do core
  - Novos padrões para integração e monitoramento da VM
  - Facilidade de desenvolvimento

Como está sendo definida

- JCP – JSR 176: J2SE 5.0 (Tiger) Release Contents
- Objetivos
  - Confiabilidade, disponibilidade e serviços (monitoramento e gerenciamento)
  - Performance e escalabilidade
  - Facilidade de desenvolvimento
  - Clientes desktop

JSR guarda-chuva

- JSR 176 não define nenhuma API, mas sim quais JSRs deverão ser incluídas no Tiger
- Fazem parte do Tiger:
  - JSR 003 – JMX
  - JSR 013 – Decimal Arithmetic Enhancement
  - JSR 014 – Generics
  - JSR 028 – SASL

JSR guarda-chuva (cont)

- Fazem parte do Tiger (cont):
  - JSR 114 – JDBC Rowset
  - JSR 133 – Memory model
  - JSR 160 – JMX-Remoting
  - JSR 163 – Profiling
  - JSR 166 – Concurrency Utilities
  - JSR 174 – Monitoring and Management

JSR guarda-chuva (cont)

- Fazem parte do Tiger (cont):
  - JSR 175 – Metadata
  - JSR 200 – Network Transfer Format
  - JSR 201 – Enums, autoboxing, extended for loops, static import and varargs
  - JSR 204 – Unicode Supplementary Character Support
  - JSR 206 – JAXP 1.3

JSR-013: Decimal Enhancement

- BigDecimal modificado para suportar melhor operações de ponto flutuante
- Introduzidas as classes java.math.MathContext e java.math.RoundingMode
- Métodos para divisão inteira, resto da divisão e exponenciação foram adicionados

JSR 014 – Generics

- Uma das JSRs mais antigas do JCP
- Melhoram o mecanismo de tipos da linguagem
- Várias APIs – java.util, java.lang e java.lang.reflect – foram adaptadas para tirarem proveito do uso de generics

JSR 014 – Generics

```
public class CollectionsSample {
   public static void main(String[] args) {
      List<String> lista = new ArrayList<String>();
      lista.add("James Gosling");
      lista.add("Joshua Bloch");
      //Erro de compilação
      //lista.add(new Integer(0));

      String nome = lista.get(0);
   }
}
```

JSR 014 - Generics

- Embora tenha benefícios, existem algumas desvantagens:
  - A sintaxe pode ser confusa

    <? super T> getSuperClass()
  - Pode ser contra-intuitivo

    List<Animal> = new ArrayList<Cachorro> (); //inválida!!!!
  - Não foi incluída a informação de tipo genérico em runtime (erasure)

JSR 160 e 174

- É importante ter métricas e controle sobre a JVM em ambientes de produção
- Este tipo de administração precisa ser feita remotamente e sem a necessidade de se programar código adicional

JSR 160 e 174

- O Java possui há muito tempo uma API de monitoramento e gerenciamento, JMX (JSR 003)
- JSR 160 estendeu essa API de modo que se possa acessar remotamente via RMI, de forma padronizada, um MBeanServer em outra JVM

JSR 160 e 174

- JSR 174 traz uso prático de JMX no próprio kernel da JVM
- Alguns recursos monitoráveis da JVM a partir do Tiger são:
  - Carregamento de classes
  - Compilação JIT
  - Garbage Collection (por região de memória)

JSR 160 e 174

- Alguns recursos monitoráveis da JVM a partir do Tiger são:
  - Pools e níveis de memória
  - Informações básicas da máquina
  - Informações de execução da JVM
  - Threads

JSR 160 e 174

- Permite as seguintes aplicações práticas:
  - Monitoramento real de um ambiente de produção
  - Receber notificações quando a memória disponível diminui
  - Extrair informações sobre contenção de threads

JSR 160 e 174

- Trazem para a plataforma:
  - Maior confiabilidade
  - Previsibilidade
  - Estabilidade
  - Possibilidade de ajustes remotos por administradores de rede/sistemas

JSR 166 – Concurrency Utilities

- Programação com threads é complexa
- Linguagem suporta somente estruturas primitivas como synchronized, wait(), notify() e notifyAll()
- Necessidade de abstração de mais alto nível

JSR 166 – Concurrency Utilities

- Define o pacote java.util.concurrency e seus subpacotes
- Suporta construções de nível mais alto como executores, semáforos, locks, futures (resultados assíncronos)
- Provém de API open-source de Doug Lea, bastante estável

JSR 175 - Metadata

- O conceito de anotações é bem estabelecido:
  - XDoclet
  - .NET
- A API de anotações no Tiger é uma das mais importantes para atingir a meta de simplicidade de desenvolvimento (java.lang.annotation)

JSR 175 - Metadata

- Anotações são metainformações ligados a classes, métodos, campos etc. que podem ser extraídos em tempo de compilação, carregamento de classe ou execução
- Podem ser usados para geração de código, configuração etc.

JSR 175 – Metadata

```
@Documented
@Retention(RUNTIME)
@Target(TYPE)
public @interface SampleAnnotation {
}

@SampleAnnotation
public class SampleClass {
...
}
```

JSR 175 – Metadata

- Várias JSRs já estão usando metadados como parte do seu modelo de implementação (JSR 220, EJB 3.0)
- Metadados trazem poder mas podem ser “abusados” (metadata hell)

JSR 200 – Network Transfer Format (Pack200)

- Define um novo formato de compactação, muito mais eficiente, para .jars
- Garante alta taxa de compactação (até 20% do tamanho original)
- Provê API programática via java.util.Pack200
- API não utiliza novas features

JSR 201

- Responsável pela maioria das mudanças da linguagem no Tiger
- Define:
  - Enum
  - Autoboxing
  - Enhanced for loop
  - Static Import
  - varargs

JSR 201

- Enums são tipos bem definidos, com instâncias conhecidas e limitadas

```
public enum Status {
     ON, OFF, BROKEN;
}
```

JSR 201

- Autoboxing é o processo de conversão automático de primitivos para classes

```
Integer i = 5;
int j = i + 3;
```

JSR 201

- O enhanced for loop simplifica as iterações em Collections e arrays

```
Collection usuarios = ...;
for (Iterator i = usuarios.iterator();
   i.hasNext(); ) {
    Usuario usuario = (Usuario)i.next();
}

//no 1.5
for (Usuario usuario : usuarios) {
}
```

JSR 201

- Static import permite o uso de constantes no código sem necessidade de prefixar com o tipo que as define

```
import static Status.*

    if (getStatus() == ON) {
       //...
    }
```

JSR 201

- Varargs permite que você passe diversos argumentos para um método com uma sintaxe simplificada:

```
public void print(Object... args) { //code }
print(5);
print("Mensagem", "outra mensagem", new
   Integer(3));
print("exemplo", 4.5, null);
```

Outras novidades do Tiger

- Class Data Sharing: as classes principais da JVM, de 5-6 Mb, são compartilhadas e pré-preparadas
- Novas features para GC: algoritmo paralelo, comportamento adaptativo etc.
- Maior precisão em medição de tempo com System.nanoTime()

Outras novidades do Tiger

- Novas classes e features em java.lang e java.util: StringBuilder, Formatter, Scanner, ProcessBuilder, System.out.printf() e o retorno de System.getenv()
- Adição de Queue, EnumSet, EnumMap e outros a java.util
- API de instrumentação de bytecode, java.lang.instrument

Outras novidades do Tiger

- Funcionalidade de ping em InetAddress.isReachable(tempo)
- Suporte a mais padrões de segurança e internacionalização
- A API de reflection sofreu grandes modificações para suportar as novas features e tirar proveito delas

Outras novidades do Tiger

- RMI é capaz de gerar stubs e skeletons dinâmicos, possui factory para uso via SSL e rmiregistry pode ser iniciado via (x)inetd
- AWT possui muitas bug fixes, MouseInfo, always on top, novas features em DnD, Clipboard, z-order

Outras novidades do Tiger

- Dois novos L&F para o Swing: Ocean and Skin
- Ainda no Swing, diversas bugfixes e features novas, como suporte especial a impressão de tabelas
- JFrame.add() finalmente é o mesmo que JFrame.getContentPane().add()

Conclusão

- Diversas mudanças foram feitas no Tiger
- A maioria delas pode ser usada no dia a dia do desenvolvedor de forma benéfica
- Algumas features podem ser mal usadas e tornar o código difícil de ler, como generics e static imports

Obrigado pela atenção. Perguntas?

michael@summa-tech.com
www.summa-tech.com

SUMMA