SERIALIZAÇÃO DE OBJETOS XML - XSTREAM

Programação Orientada a Objetos

Prof. Christian Danniel Paz Trillo

Profa. Karina Valdivia Delgado

Lembrando: SERIALIZAÇÃO DE OBJETOS

- A serialização é um processo de converter um objeto em um formato que pode ser armazenado e recuperado posteriormente.
  - O objeto recuperado deve ser exatamente igual ao objeto que foi armazenado.
  - A descrição da classe (nomes dos atributos e operações) não requer ser armazenada.

Lembrando:
Serialização – Fluxo de bytes

- Java oferece um mecanismo de serialização que permite armazenar os dados de um objeto complexo em um único fluxo de bytes.

Exemplo:

Aluno *a* = new Aluno(“Christian”, {8, 5, 7, 9});

Serializando *a*:

¬í sr poo.Aluno…›S³ëÍ D mediaL nomet Ljava/lang/String;[notat [Ixp@ t Christianur [IMº`&vê²¥ xp

Lembrando:
Serialização – Fluxo de bytes

- É um formato mais difícil de entender e alterar.
- Armazena mais informações:
  - Nome da classe.
  - Nomes de alguns atributos (media, nome, nota).
  - Informações de verificação: São dados redundantes que permitem verificar se os valores lidos são válidos. Ex: O CPF utiliza esta idéia: os dois últimos dígitos do CPF permitem validar se um CPF é válido.
- Não preciso implementar métodos de serialização / desserialização, Java sabe o que armazenar e o que não armazenar.

Lembrando:
Serialização – Fluxo de bytes

- Para podermos serializar um objeto de uma classe X:
  - Precisamos informar que X implementa a interface Serializable.
  - Declarar uma constante que indica a versão (opcional)
- Utilizamos duas classes para serializar:

ObjectOutputStream e FileOutputStream.

- Utilizamos duas classes para desserializar:

ObjectInputStream e FileInputStream.

Lembrando:
Serialização – Fluxo de bytes

```
public class Aluno implements Serializable{



   protected String nome;
   protected int [] nota;
   private double media;
}
```

Lembrando:
Serialização – Fluxo de bytes

```
Aluno a = new Aluno("Christian", 8, 5, 7, 9);
try {
    ObjectOutputStream o = new ObjectOutputStream(
                           new FileOutputStream(
                           "c:\\arq.txt"));
    o.writeObject(a);
  } catch (IOException e) {
      e.printStackTrace();
  }
```

Lembrando:
Serialização – Fluxo de bytes

- ObjectOutputStream é responsável pela serialização, i.e., traduzir o objeto a um fluxo de bytes.
- FileOutputStream é responsável por escrever o fluxo de bytes a um arquivo no sistema operacional.
- Se trocamos a linha 3 por:

```
ObjectOutputStream o = new
                        ObjectOutputStream(System.out);
```

O fluxo será escrito diretamente na tela (System.out).

- O método writeObject realiza o processo de serialização.

Lembrando:
Serialização – Fluxo de bytes

Vantagens:

- Não preciso implementar nada.
- Menos fácil de alterar diretamente (dificulta fraudes).
- Algumas dicas adicionais sobre como personalizar a serialização de cada atributo podem ser encontradas em:

http://www.ibm.com/developerworks/br/library/j-5things1/

por exemplo para encriptar a senha dos usuários

SERIALIZAÇÃO - BINÁRIO

Desvantagens:

- Depende da linguagem de programação (C# tem seu próprio formato).
- Se bem alguns tipos de dados são armazenados eficientemente, não aproveita possibilidades de compactação para reduzir o tamanho do arquivo.
  - O arquivo cresce linearmente com o tamanho do objeto.
- Mudanças na estrutura da classe (versões) podem levar a incompatibilidade.
  - Ex: se mudamos o tipo de dado de algum atributo da classe.

SERIALIZAÇÃO – Outros formatos padrão

- Uma das principais desvantagens de um formato próprio de uma linguagem é a impossibilidade de trocar dados entre programas.
- Para isso existem outros formatos padrão para serializar objetos, textuais e binários:
  - Texto: XML, CSV, JSON.
  - Binários: BSON e Thirft .

SERIALIZAÇÃO – Outros formatos padrão

- Formatos Texto:
  - **XML**: Baseado na linguagem de marcação criada para páginas Web (HTML) e amplamente utilizado e suportado por bibliotecas.
  - **CSV** (Comma Separated Values): Armazenamento de dados simples no estilo banco de dados. Ex: Array de objetos:
    - cada objeto é representado por uma linha e os atributos separados por algum caractere (vírgula, ponto e vírgula, etc).
  - **JSON** (JavaScript Object Notation): Formato criado para troca de informações na Web.
    - Pode ser diretamente avaliado pelo compilador JavaScript.

SERIALIZAÇÃO – Outros formatos padrão

- Formatos Binários:
  - **BSON** (Binary JSON): Codifica os arquivos JSON no formato binário, traduzindo os objetos JSON seguindo algumas regras simples.
  - **Thirft**: Formato desenvolvido por Facebook e liberado como parte do projeto Apache (Open Source).
    - Altamente suportado em diversas bibliotecas para Java, C#, Perl, etc.

SERIALIZAÇÃO – Outros formatos padrão

- Veremos bibliotecas para serializar objetos a XML:
  - XStream

XML (Extensible Markup Language)

- É um dos formatos mais utilizados para descrever objetos e tipos de dados.
- É uma linguagem Padrão W3C (WWW Consortium).

XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita nome="pão" tempo_de_preparo="5 minutos"
   tempo_de_cozimento="1 hora">
<titulo>Pão simples</titulo>
<ingredientes>
    <ingrediente quantidade="3" unidade="xícaras">Farinha</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1.5" unidade="xícaras"
       estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
</ingredientes>
<instrucoes>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, coloque numa bandeja e asse num forno.</passo>
</instrucoes>
</receita>
```

XML

- XML permite descrever:
  - Valores (String):
    - **<titulo>**Pão simples**</titulo>**
  - Objetos:
    - **<ingrediente** quantidade="3" unidade="xícaras"**>**Farinha**</ingrediente>**
  - Coleções (String ou de Objetos):

<instrucoes>

<passo>Misture todos os ingredientes, e dissolva bem.**</passo>**

<passo>Cubra com um pano e deixe por uma hora em um local morno.**</passo>**

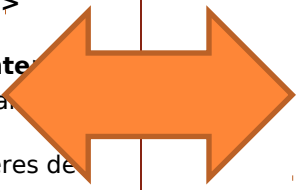
<passo>Misture novamente, coloque numa bandeja e asse num forno.**</passo>**

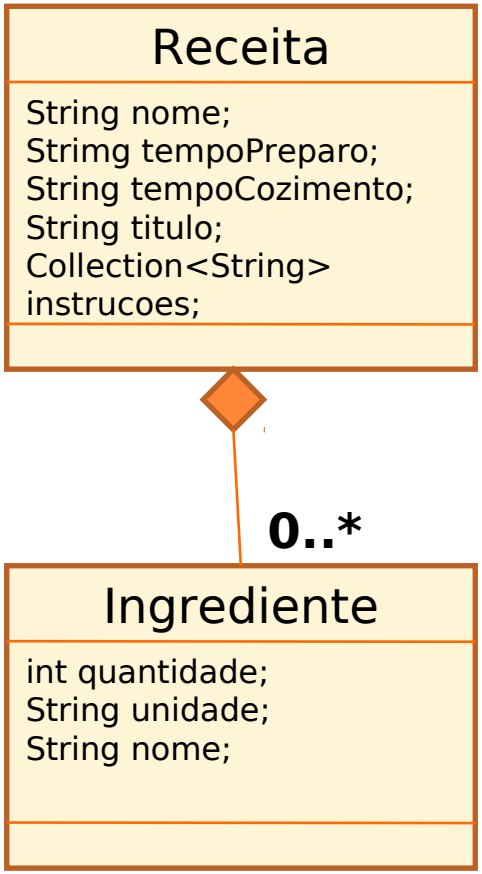
</instrucoes>

XML - OBJETO

- Com isso, é possível utilizar XML para serializar objetos.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<receita nome="pão" tempo_de_preparo="5 minutos"
    tempo_de_cozimento="1 hora">
<titulo>Pão simples</titulo>
<ingredientes>
    <ingrediente quantidade="3"
        unidade="xícaras">Farinha</ingrediente>
    <ingrediente quantidade="7"
        unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1.5" unidade="xíca
        estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de
        chá">Sal</ingrediente>
</ingredientes>
<instrucoes>
    <passo>Misture todos os ingredientes, e dissolva
        bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em
        um local morno.</passo>
    <passo>Misture novamente, coloque numa bandeja e
        asse num forno.</passo>
</instrucoes>
</receita>
```





XML – BIBLIOTECAS JAVA

- A Biblioteca Padrão Java provê uma funcionalidade para leitura e escrita de arquivos XML:
  - javax.xml.parsers.DocumentBuilder
- Porém é necessário escrever um código especial para reconhecer cada estrutura XML.

XML – BIBLIOTECAS JAVA

```
DocumentBuilderFactory f = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = f.newDocumentBuilder();
Document doc = builder.parse("input.xml");
NodeList nodeList = doc.getElementsByTagName("location");
// Iterar sobre a lista de locations
for(int i = 0; i < nodeList.getLength
    // Pega o i-ésimo elemento
    Element el = (Element)nodeList.
    // Imprime alguns dados do i-ésimo elemento.
    System.out.println(el.getAttribute("countryname") + " - " +
    el.getAttribute("cityname") + " - " +
    el.getAttribute("citycode"));
}
```

Devolve um NodeList com todos os Elements do documento que tem esse Tag

XML – BIBLIOTECAS JAVA

```
DocumentBuilderFactory f = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = f.newDocumentBuilder();
Document doc = builder.parse("input.xml");
NodeList nodeList = doc.getElementsByTagName("location");
// Iterar sobre a lista de locations
for(int i = 0; i < nodeList.getLength(); i++){
    // Pega o i-ésimo elemento
    Element el = (Element)nodeList.item(i);
    // Imprime alguns dados do i-ésimo elemento.
    System.out.println(el.getAttribute("countryname") + " - " +
    el.getAttribute("cityname") + " - " +
    el.getAttribute("citycode"));
}
```

XSTREAM

- Xstream é uma biblioteca para serializar e desserealizar objetos a XML de forma simples.

- Características:
  - Fácil de Usar.
  - Não é necessário realizar mapeamentos de tipos de objetos, a biblioteca já reconhece a maioria de tipos de dados:
    - String, int, bool, ...
    - Coleções, Mapas, Vetores, ...
  - O XML resultante é de fácil leitura para humanos.
  - Serializa atributos públicos, privados e protegidos.

XSTREAM

```
//Inicialização de Objetos
Receita receita = new Receita();
receita.nome = “pão”;
...
receita.ingredientes.add(new Ingrediente(“Farinha”,
3, “xicaras”));
...
//Serialização a um String com formato XML
XStream xstream = new XStream(new DomDriver());
String xml = xstream.toXML(receita);
System.out.println(xml);
Receita copia=(Receita)xstream.fromXML(xml);
```

Xstream atua como um FAÇADE. Ela permite o acesso aos principais recursos da biblioteca.

XSTREAM

```
//Inicialização de Objetos
Receita receita = new Receita();
receita.nome = “pão”;
...
receita.ingredientes.add(new Ingrediente(“Farinha”,
3, “xicaras”));
...
//Serialização a um String com formato XML
XStream xstream = new XStream(new DomDriver());
String xml = xstream.toXML(receita);
System.out.println(xml);
Receita copia=(Receita)xstream.fromXML(xml);
```

O argumento é Um Driver. Ex: SAX, DOM, DOM_4J, XPP_3

XSTREAM

```
//Inicialização de Objetos
Receita receita = new Receita();
receita.nome = “pão”;
...
receita.ingredientes.add(new Ingred   te(“Farinha”,
3, “xicaras”));
...
//Serialização a um String com formato XML
XStream xstream = new XStream(new DomDriver());
String xml = xstream.toXML(receita);
System.out.println(xml);
Receita copia=(Receita)xstream.fromXML(xml);
```

O método toXML devolve um String. Se a lista de objetos a serializar é muito grande, pode gastar muita memória.

XSTREAM – XML gerado e mostrado na tela

```
<poo.Receita>
  <nome>pão</nome>
  <titulo>pão simples</titulo>
  <tempoCozimento>1 hora</tempoCozimento>
  <tempoPreparo>5 minutos</tempoPreparo>
  <ingredientes>
    <poo.Ingrediente>
      <nome>Farinha</nome>
      <quantidade>3</quantidade>
      <unidade>xicaras</unidade>
    </poo.Ingrediente>
</ingredientes>
  <instrucoes/>
</poo.Receita>
```

XSTREAM – XML gerado e mostrado na tela

```
<poo.Receita>
  <nome>pão</n
  <titulo>pão sim
  <tempoCozimer
  <tempoPreparo>5
  <ingredientes>
    <poo.Ingrediente>
      <nome>Farinha</nome>
      <quantidade>3</quantidade>
      <unidade>xicaras</unidade>
    </poo.Ingrediente>
</ingredientes>
  <instrucoes/>
</poo.Receita>
```

Note que XStream gerou a tag raiz com o nome da classe e gerou o XML recursivamente para cada atributo daquela classe.

Xstream serializa por default os objetos através de seus atributos e não através de getters e setters

XSTREAM - ARQUIVOS

Para poder persistir a informação é necessário armazená-la em arquivos, assim mudaremos as chamadas ao fromXML e toXML para ler escrever em arquivos:

- Leitura:
  - Receita copia = (Receita)xstream.fromXML
    (**new FileInputStream("arq.xml"));**
- Escrita:
  - xstream.toXML(receita,
    new FileOutputStream("arq.xml"));
- Quando escrevemos em arquivos é necessário cuidar dos caracteres acentuados. Trabalharemos sem caracteres acentuados.

XSTREAM – REFERÊNCIAS

- Imaginemos o modelo de classes do projeto da Urna Eletrônica.
- Lembremos que o Candidato se encontra na lista de eleitores e na de candidatos.
  - O objeto candidato deve ser armazenado duas vezes?

  Não: pois na hora de desserializar perderíamos a informação de que o Candidato é o mesmo objeto em ambas listas.

Esse problema é chamado de problema de armazenamento de referências.

XSTREAM - REFERÊNCIAS

- Exemplo:

```
...
Candidato c1 = new Candidato(“Julio”, 234, 1, “PT”);
urna.eleitores.add(c1);
urna.candidatos.add(c1);
...
xstream.setMode(XStream.ID_REFERENCES);
xstream.toXML(urna);
```

XSTREAM - REFERÊNCIAS

```
<poo.Urna>
    <poo.eleitores>
        <poo.Candidato id="1">
        <nome>Julio</nome>
        ...
    </poo.eleitores>
    <poo.candidatos>
        <poo.Candidato reference="1">
        ...
    </poo.candidatos>
</poo.Urna>
```

XSTREAM – ALIAS

- É possível alterar os nomes utilizados para armazenar no xml.
- Nos nomes das classes:
  - xstream.alias(“urnaeletronica”, Urna.class);
  - <poo.Urna> vira <urnaeletronica>
- Nos nomes dos atributos:
  - xstream.aliasField(“listavotantes”, Urna.class, “eleitores”);
  - <poo.eleitores> vira <listavotantes>

XSTREAM - ATRIBUTOS

- O Xml gerado por padrão pelo Xstream cria um tag para cada atributo da classe.

<poo.Candidato>

<nome> Julio</nome>

...

- Quando falamos em **tipos simples de dados**, podemos utilizar a notação resumida:

<poo.Candidato nome="Julio">

...

Para isso, utilizamos o método useAttributeFor:

- xstream.useAttributeFor(Candidato.**class, "nome");**

SERIALIZAÇÃO – RESUMO

- A Serialização é um processo muito utilizado em sistemas complexos:
  - Não tanto para armazenar informação de forma permanente, pois os Bancos de Dados fazem isso de forma mais eficiente e flexível.
  - Para transmitir informação entre aplicações de diversas plataformas.
  - Para transmitir informação pela internet – Web Services.
  - Para popular páginas Web.
  - Para armazenar informações de configuração:
    - Propriedades de Conexão a BD.
    - Propriedades de uma aplicação: moeda, separador de decimais, etc.

SERIALIZAÇÃO - Exercício

- Sistema de Cadastro de Livros
  - Entidades:
    - Livros: Título, Autores, ISBN.
    - Autores: ID, Nome, Resumo Biográfico.
  - O sistema armazena (com XStream) a informação de uma lista de livros e na próxima execução os dados já cadastrados aparecerão.