
Representação externa de dados e empacotamento

— Guilherme Baccarin —
03 de Junho de 2022
Sistemas Distribuídos

Informações armazenadas nos programas são representadas como estruturas.

Informações presentes nas mensagens, são uma sequência de bytes.

Informações devem ser simplificadas antes da comunicação (convertidas em bytes) e reconstruídas na sua chegada.

Exemplo:

- Tipos primitivos podem ser mal-interpretados.
- Padrão de representação de caracteres.

Conversão

São convertidos para um formato acordado anteriormente a comunicação e convertidos novamente para o formato original na recepção da mensagem (podendo ser omitida essa conversão).

A ordem dos bits nunca é alterada.

Para suportar RMI e RPC, tipos de comunicação, a comunicação deve ser sempre feita em um formato comum.

Um padrão que aceita tipos primitivos é chamado de: **Representação Externa de Dados**.

Empacotamento X Desempacotamento

Empacotamento (Marshalling):

- Pegar um conjunto de itens e organizá-los de forma que seja conveniente para a transmissão dos mesmos.
- Dados estruturados e tipos primitivos para uma representação externa.

Desempacotamento (Unmarshalling):

- Procedimento inverso após a chegada dos dados.
- Uma representação externa para dados primitivos e estruturados.

Representação Externa

- CORBA.
- Serialização de objetos da linguagem JAVA.
- XML.
- JSON.
- Buffers de protocolo.



```
{
  hey: "guy",
  anumber: 243,
  - anobject: {
    whoa: "nuts",
    - anarray: [
      1,
      2,
      "thr<h1>ee"
    ],
    more: "stuff"
  },
  awesome: true,
  bogus: false,
  meaning: null,
  japanese: "明日がある。",
  link: http://jsonview.com,
  notLink: "http://jsonview.com is great"
}
```



CORBA

Executada na camada de *middleware*.

Sem envolvimento do programador (a).

Dados primitivos empacotados de forma binária.

Não possui informação sobre o tipo do dado, apenas o valor.

Possuem 15 tipos diferentes que podem ser representados:

- short, long, unsigned short, unsigned long, float, double, char

Tipos primitivo

Transmitidos na ordem do remetente, sendo definida em cada mensagem.

x Tipos compostos

Ordem específica de acordo com o tipo de dado.

Cada caractere ocupa o espaço de um byte.

São preenchidos com zeros para completar o tamanho, caso necessário.

Tipo de dado não é fornecido.

<i>Índice na sequência de bytes</i>	<i>← 4 bytes →</i>	<i>Observações sobre a representação</i>
0–3	5	<i>Comprimento do string</i>
4–7	"Smit"	<i>'Smith'</i>
8–11	"h__"	
12–15	6	<i>Comprimento do string</i>
16–19	"Lond"	<i>'London'</i>
20–23	"on__"	
24–27	1984	<i>unsigned long</i>

Empacotamento pode ser gerado automaticamente através da descrição do IDL.

```
struct Person{  
    string name;  
    string place;  
    unsigned long year;  
};
```


Serialização de objetos da linguagem JAVA

Possui informação sobre o tipo do dado, apenas o valor.

Informações de tipos, inseridas de forma serializada.

Serialização é a atividade de simplificar um objeto para facilitar a transmissão ou armazenamento da informação.

Desserialização é o processo inverso.

Para desserializar não é necessário ter informações sobre o estado anterior da informação, mas caso tenha, durante a serialização, essa informação é serializada junto do resto do objeto.

Informação da classe consiste em seu nome e a versão em que se encontra, esse que muda a cada vez que forem feitas alterações.

O objeto inteiro é serializado, incluindo os que ele referencia.

```
public class Person implements Serializable {  
    private String name;  
    private String place;  
    private int year;  
    public Person(String aName, String aPlace, int aYear) {  
        name = aName;  
        place = aPlace;  
        year = aYear;  
    }  
    // seguido dos métodos para acessar as variáveis de instância  
}
```

Valores serializados				Explicação
Person	Número da versão de 8 bytes		h0	Nome da classe, número da versão
3	int year	java.lang.String name	java.lang.String place	Número, tipo e nome das variáveis de instância
1984	5 Smith	6 London	h1	Valores das variáveis de instância

Na realidade, a forma serializada inclui marcas adicionais de tipos; h0 e h1 são identificadores.

Referências são serializadas através dos identificadores (*handlers*).

Para serializar, a informação é escrita por extenso, sendo escrito os tipos seguido dos nomes das variáveis.

É um processo recursivo.

Cada classe recebe um identificador (*handler*) e é salva apenas uma vez.

Variáveis dos tipos primitivos são salvas no formato binário.

Serialização e desserialização normalmente é feita automaticamente pela camada de *middleware*.

Pode-se ter variáveis que devem ser serializadas e outras que não devem. As que não desejem serializar, devem ser declaradas como transientes.

XML

Linguagem de marcação.

Definida pela W3C.

Dados primitivos representados de forma textual.

Maior que a forma binária de representar os dados.

Possui informação sobre o tipo do dado, apenas o valor.

Informação sobre o tipo pode ser definida externamente (espaço de nome).

Rotuladas a partir de *tags*, usadas para descrever a lógica da estrutura.

Permite comunicação entre clientes e serviços *WEB*.

Usada no arquivamento e recuperação de sistemas.

Pode ser lido por qualquer computador.

É extensível, então é possível criar as próprias *tags*.

Para comunicação, os documentos XML, devem combinar em suas *tags*.

Uso das *tags*, permite o uso de apenas as partes do documento que sejam interessantes para aquele processo.

Fácil compreensão pelos humanos.

Maioria dos arquivos é gerado automaticamente.


```
<person id="123456789">  
    <name>Smith</name>  
    <place>London</place>  
    <year>1984</year>  
    <!-- a comment -->  
</person >
```

Estrutura


Elementos: Conjuntos de dados do tipo caractere.

Nomes: Identificam a *tag*.

CDATA: Para caracteres especiais



`<person id="123456789">... </person >.`



`<person id="123456789">... </person >.`

`<place> <![CDATA [King's Cross]]></place >`

Estrutura

Prólogo:

```
<?XML version = "1.0" encoding = "UTF-8" standalone = "yes"?>
```

Esquemas: Define elementos e atributos que podem aparecer no documento.

APIs:

```
<xsd:schema xmlns:xsd = URL das definições de esquema XML >  
  <xsd:element name= "person" type = "personType"/>  
    <xsd:complexType name="personType">  
      <xsd:sequence>  
        <xsd:element name = "name" type="xs:string"/>  
        <xsd:element name = "place" type="xs:string"/>  
        <xsd:element name = "year" type="xs:positiveInteger"/>  
      </xsd:sequence>  
      <xsd:attribute name= "id" type = "xs:positiveInteger"/>  
    </xsd:complexType>  
</xsd:schema>
```

Representação externa de dados e empacotamento

— Guilherme Baccarin —
03 de Junho de 2022
Sistemas Distribuídos
