
JAVA™

Trabalhando com datas

Date, Calendar e SimpleDateFormat

O Java fornece as classes acima para o uso de valores do tipo data. **Date** e **Calendar** são tipos de dados, enquanto que **SimpleDateFormat** permite a formatação e conversão de Strings para data e vice versa.

Além de converter valores também são fornecidos métodos para internacionalizar as datas de acordo com a região.

Date

A data representa um marco do tempo. Date é uma classe que representa esse marco, que é composto por ano, mês, dia atual, minuto atual, entre outras propriedades que essa classe possui.

Hoje a maioria dos métodos da classe Date estão classificados como deprecated (depreciados), ou seja, são métodos que não são mais utilizados, por isso essa classe foi substituída pela classe Calendar, que trouxe diversas melhorias para o tratamento de datas incluindo o suporte correto à internacionalização do sistema de datas.

Date

```
import java.util.Date;

public class ExemploDate {
    public static void main(String[] args) {
        Date data = new Date();
        System.out.println("Data agora: "+data);
    }
}
```

Calendar

Essa classe pode produzir os valores de todos os campos de calendário necessários para implementar a formatação de data e hora, para uma determinada língua e estilo de calendário. Por exemplo, japonês, americano, italiano, brasileiro entre outros.

Hoje, a classe Calendar é a mais usada quando se trata de datas, mas como é uma classe abstrata, não pode ser instanciada, portanto para obter um calendário é necessário usar o método estático getInstance().

Calendar


```
import java.util.Calendar;

public class ExemploCalendar{
    public static void main(String[] args) {
        Calendar c = Calendar.getInstance();
        System.out.println("Data e Hora atual: ”+c.getTime());
    }
}
```

Calendar

Dia da semana, mês e ano


```
import java.util.Calendar;

public class ExemploCalendar{

	public static void main(String[] args) {
		Calendar c = Calendar.getInstance();
		System.out.println("Data e Hora atual: ”+c.getTime());

		System.out.println("Data/Hora atual: "+c.getTime());
		System.out.println("Ano: "+c.get(Calendar.YEAR));
		System.out.println("Mês: "+c.get(Calendar.MONTH));
		System.out.println("Dia do Mês: "+c.get(Calendar.DAY_OF_MONTH));
	}
}
```

Calendar

Alterando com o método set


```
import java.util.Calendar;

public class AlteraData{

	public static void main(String[] args) {
		Calendar c = Calendar.getInstance();

		System.out.println("Data/Hora atual: "+c.getTime());

		c.set(Calendar.YEAR, 1995);
		c.set(Calendar.MONTH, Calendar.MARCH);
		c.set(Calendar.DAY_OF_MONTH, 20);

		System.out.println("Data/Hora alteradas: "+c.getTime());
		System.out.println("Ano: "+c.get(Calendar.YEAR));
		System.out.println("Mês: "+c.get(Calendar.MONTH));
		System.out.println("Dia do Mês: "+c.get(Calendar.DAY_OF_MONTH));
	}
}
```

Calendar

Manipulando os dados


```
import java.util.Calendar;

public class ExemploCalendarHora{

	public static void main(String[] args) {
		Calendar c = Calendar.getInstance();
		int hora = c.get(Calendar.HOUR_OF_DAY);

		if(hora > 6 && hora < 12){
			System.out.println("Bom Dia");
		}else if(hora > 12 && hora < 18){
			System.out.println("Boa Tarde");
		}else{
			System.out.println("Boa Noite");
		}
	}
}
```

DateFormat

Essa classe permite aplicar formatação aos valores. Por ser uma classe abstrata, não é possível instanciá-la, por isso deve ser usado para método estático getDateInstance().

DateFormat

```
import java.util.Calendar;
import java.text.DateFormat;
import java.util.Date;

public class ExemploFormatarData{

        public static void main(String[] args) {
                Calendar c = Calendar.getInstance();
                c.set(2013, Calendar.FEBRUARY, 28);
                Date data = c.getTime();
                System.out.println("Data atual sem formatação: "+data);

                //Formata a data
                DateFormat formataData = DateFormat.getDateInstance();
                System.out.println("Data atual com formatação: "+ formataData.format(data));

                //Formata Hora
                 DateFormat hora = DateFormat.getTimeInstance();
                System.out.println("Hora formatada: "+hora.format(data));

                //Formata Data e Hora
                DateFormat dtHora = DateFormat.getDateTimeInstance();
                System.out.println(dtHora.format(data));

        }
}
```

DateFormat


```
import java.util.Calendar;
import java.text.DateFormat;
import java.util.Date;

public class ExemploFormatarSaidaData{

        public static void main(String[] args) {
                Calendar c = Calendar.getInstance();
                Date data = c.getTime();
                DateFormat f = DateFormat.getDateInstance(DateFormat.FULL); //Data Completa
                System.out.println("Data brasileira: "+f.format(data));

                f = DateFormat.getDateInstance(DateFormat.LONG);
                System.out.println("Data sem o dia descrito: "+f.format(data));

                f = DateFormat.getDateInstance(DateFormat.MEDIUM);
                System.out.println("Data resumida 1: "+f.format(data));

                f = DateFormat.getDateInstance(DateFormat.SHORT);
                System.out.println("Data resumida 2: "+f.format(data));
        }
}
```

SimpleDateFormat

Às vezes é preciso transformar um texto em uma data ou vice versa, para isso existe a função parse e a classe SimpleDateFormat.

SimpleDateFormat


```
import java.util.Calendar;
import java.text.DateFormat;
import java.util.Date;
import java.text.SimpleDateFormat;
import java.text.ParseException;

public class ExemploConversaoData{

        public static void main(String[] args) throws ParseException{
                Calendar c = Calendar.getInstance();
                Date data = c.getTime();
                DateFormat f = DateFormat.getDateInstance();

                Date data2 = f.parse("12/01/1995");
                System.out.println(data2);

                SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
                System.out.println("Data formatada: "+sdf.format(data));

                //Converte Objetos
                System.out.println(“Data convertida: ”+sdf.parse("02/08/1970"));
        }
}
```

Internacionalização

O Java oferece a classe Locale, que permite definir de qual país deseja-se obter o retorno das informações de data e hora.

Nos exemplos anteriores não foi necessário instanciar essa classe, pois já é detectado automaticamente quais são as configurações regionais do computador.

Internacionalização


```
import java.util.Calendar;
import java.util.Locale;
import java.text.DateFormat;
import java.text.ParseException;
import java.util.Date;

public class ExemploLocale{

        public static void main(String[] args) throws ParseException {
                Calendar c = Calendar.getInstance();
                Date data = c.getTime();

                Locale brasil = new Locale("pt", "BR"); //Retorna do país e a língua
                Locale eua = Locale.US;
                Locale italia = Locale.ITALIAN;

                DateFormat f2 = DateFormat.getDateInstance(DateFormat.FULL, brasil);
                System.out.println(“Data e hora brasileira: ”+f2.format(data));

                DateFormat f3 = DateFormat.getDateInstance(DateFormat.FULL, eua);
                System.out.println(“Data e hora americana: ”+f3.format(data));

                DateFormat f4 = DateFormat.getDateInstance(DateFormat.FULL, italia);
                System.out.println(“Data e hora italiana: ”+f4.format(data));
        }
}
```