LEARNING jodatime

#jodatime

Table of Contents

About

Chapter 1: Getting started with jodatime

Examples

Introduction

Joda-Time is a robust alternative to the Java date and time classes.

Prior to Java SE 8, the standard Java date and time classes like `java.util.Calendar` are difficult to use and prone to errors. Joda-Time emerged as the de-facto standard library for date and time manipulation in many open-source-projects.

However, starting with Java SE 8 the package `java.time` (JSR-310) is available and users are asked to migrate to the same since Joda-Time is now in maintenance mode.

When to use Joda-Time

You want to manipulate dates and times **and**:

1. You are developing a project in an environment where Java SE8 is not available
2. You are maintaining a legacy project that already uses Joda-Time
3. You are developing a cross-platform project and you would like to maintain an API which has similarities to the APIs of other libraries like Noda Time and js-joda (although there is no exact match).

When not to use Joda-Time

1. You don't need to work with dates and times
2. You are developing a new project where Java SE8 is available: instead use the `java.time` (JSR-310) classes.

Considerations for using Joda-Time in Android apps

Since the standard Joda-Time library can inflate the memory-footprint of apps, consider using joda-time-android. This is a fork optimized for Android development, and also contains a Joda-Time port of Android's native `DateUtils`.

Installation

Using the library archive

Download the JAR and add it to the classpath for your Java project

Using a build tool

If you are using a build tool like Maven or Gradle:

1. Maven

    ```
    <dependency>
        <groupId>joda-time</groupId>
        <artifactId>joda-time</artifactId>
        <version>2.9.6</version>
    </dependency>
    ```

2. Gradle

    Add the following line to the `dependencies` closure inside your `build.gradle`:

    ```
     compile 'joda-time:joda-time:2.9.6'
    ```

Hello Joda!

We can now write the following HelloJoda program!

```
import org.joda.time.LocalDate;

public class HelloJoda {

    public static void main(String [] args) {
        LocalDate today = LocalDate.now();
        System.out.println("Hello Joda! Today's date is: " + today);
    }
}
```

Which will output something like this:

```
Hello Joda! Today's date is: 2016-11-26
```

Read Getting started with jodatime online: https://riptutorial.com/jodatime/topic/8006/getting-started-with-jodatime

Chapter 2: LocalDate (dealing with dates)

Examples

Introduction and use case

A `LocalDate` is a date without a timezone. Consider using them when you are only concerned with the year, month, and day of month and are not interested in an exact time. For example, when we write our date of birth (such as 1960-01-01) we don't care about the timezone in which it happened.

Basic LocalDates

A given year-month-day:

```
LocalDate oneJanuaryNineteenSixty = new LocalDate(1960,1,1);
```

Today's date:

```
LocalDate today = LocalDate.now()
```

Tomorrow:

```
LocalDate tomorrow = LocalDate.now().plusDays(1);
```

Yesterday:

```
LocalDate yesterday = LocalDate.now().minusDays(1);
```

Two weeks ago:

```
LocalDate twoWeeksAgo = LocalDate.now().minusWeeks(2);
```

Nine months before 1960-01-01:

```
LocalDate conception = new LocalDate(1960-01-01).minusMonths(9);
```

The Saturday of this week:

```
LocalDate saturday = LocalDate.now().withDayOfWeek(DateTimeConstants.SATURDAY);
```

Converting other types into LocalDates

Converting a `java.util.Calendar` object:

```
Calendar rightNow = Calendar.getInstance();
LocalDate today = LocalDate.fromCalendarFields(rightNow);
```

Converting a `java.util.Date` object:

```
Date rightNow = new Date();
LocalDate today = LocalDate.fromDateFields(rightNow);
```

Converting a string:

```
String dateString = "1960-01-01"
LocalDate birthDate = LocalDate.parse(dateString);
```

Read LocalDate (dealing with dates) online: https://riptutorial.com/jodatime/topic/8023/localdate--dealing-with-dates-

Credits

S. No	Chapters	Contributors
1	Getting started with jodatime	Community, Daniel Käfer, David Rawson, Meno Hochschild
2	LocalDate (dealing with dates)	David Rawson