Package trong Java

Một package trong Java là một nhóm các kiểu lớp, Interface và package con tương tự nhau. Package trong Java có thể được phân loại thành: Package đã xây dựng sẵn và package do người dùng định nghĩa. Có nhiều package đã xây dựng sẵn như java, lang, awt, javax, swing, net, io, util, sql, … Chương này chúng ta sẽ tìm hiểu cách tạo và sử dụng các package do người dùng tự định nghĩa.

Gói (package) được sử dụng trong Java để ngăn cản việc xung đột đặt tên, điều khiên truy cập, giúp việc tìm kiếm/lưu trữ và sử dụng lớp, interface, enumeration, annotation dễ dàng hơn.

Một package có thể được định nghĩa như một nhóm các kiểu có liên quan đến nhau (lớp, interface, enumeration và annotation) cung cấp việc bảo vệ truy cập và quản lý tên.

Một vài package có sẵn trong Java như:

- **java.lang** - Các lớp cơ bản
- **java.io** - Các lớp input và output cơ bản

Lập trình viên có thể định nghĩa gói riêng để bao bọc một nhóm các class/interface. Trong thực tế, việc nhóm các class liên quan đến nhau giúp cho lập trình viên dễ dàng xác định class, interface, enumeration, annotation liên quan đến nhau.

Từ việc một gói tạo một không gian tên mới trong các package khác nhau có thể tránh việc xung đột đặt chung tên tại các gói khác nhau. Với việc sử dụng package, có thể dễ dàng cung cấp khả năng truy cập và nó dễ dàng để chứa các class liên quan đến nhau.

Tạo một package trong Java

Khi tạo một package trong Java, bạn nên chọn tên cho package và đặt câu lệnh khai báo**package** ở trên cùng của source file.

Lệnh **package** nên đặt tại dòng code đầu tiên. Bạn chỉ có thể khai báo lệnh package này một lần trong một source file, và nó áp dụng tới tất cả các kiểu trong file.

Nếu một lệnh khai báo package không được sử dụng, kiểu class, interface, enumerations hoặc annotation sẽ được đặt vào package mặc định không có tên.

Ví dụ:

Cùng xem ví dụ về việc tạo một package tên là *animals*. Trong thực tế lập trình, việc sử dụng các package thường lấy tên viết thường để tránh xung đột giữa với tên class và tên interface.

Đặt một interface trong package *animals*:

```
/* Ten File : Animal.java */
package animals;


interface Animal {
   public void eat();
   public void travel();
}
```

Lợi thế của package trong Java

- Java package được sử dụng để phân loại các lớp và các interface để mà chúng có thể được duy trì dễ dàng hơn.
- Java package cung cấp bảo vệ truy cập.
- Java package xóa bỏ các xung đột về đặt tên.

Ví dụ khác về package trong Java

Từ khóa package được sử dụng để tạo một package trong Java.

```
//Luu duoi dang Simple.java
package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Chao mung ban den voi package trong Java");
   }
}
```

Cách biên dịch Java package

Nếu bạn không sử dụng bất cứ IDE nào, bạn cần theo cú pháp sau:

```
javac -d thu_muc ten_javafile
```

Ví dụ:

```
javac -d . Simple.java
```

Tùy chọn –d xác định đích, là nơi để đặt class file đã tạo. Bạn có thể sử dụng bất cứ tên thư mục nào như /home (với Linux), d:/abc (với Windows), … Nếu bạn muốn giữ package bên trong cùng thư mục, bạn có thể sử dụng dấu chấm (.).

Cách chạy chương trình Java package

Bạn cần sử dụng tên đầy đủ (ví dụ mypack.Simple) để chạy lớp đó.

- Để biên dịch: javac -d . Simple.java
- Để chạy: java mypack.Simple

–d là một switch mà nói cho trình biên dịch Compiler nơi để đặt class file (nó biểu diễn đích đến). Dấu chấm (.) biểu diễn folder hiện tại.

Từ khóa import trong Java

Nếu một class sử dụng một class khác cùng package, tên package không cần được sử dụng. Lớp trong cùng package tìm thấy nhau mà không cần cú pháp đặc biệt nào.

Ví dụ:

Tại đây, một lớp Boss được thêm vào một package payroll đã chứa Employee. Lớp Boss có thể ám chỉ đến lớp Employee mà không cần sử dụng tiền tố payroll, như được minh họa như sau bởi lớp Boss.

```
package payroll;


public class Boss
{
   public void payEmployee(Employee e)
   {
      e.mailCheck();
   }
```

```
}
```

Nếu xảy ra trường hợp Boss không nằm trong payroll package, lớp Boss phải sử dụng một trong những kỹ thuật sau đây để tham chiếu đến class thuộc package khác.

- Sử dụng tên đầy đủ của class có thể được sử dụng. Ví dụ:

```
payroll.Employee
```

- Package có thể được nhập bởi sử dụng từ khóa import và wild card (*). Ví dụ:

```
import payroll.*;
```

- Một class có thể import chính nó với từ khóa import. Ví dụ:

```
import payroll.Employee;
```

Ghi chú: Một class file có thể chứa bất kỳ số lệnh import nào. Lệnh import phải xuất hiện sau mỗi lệnh khai báo package và trước từ khóa khai báo lớp.

Cách truy cập package từ package khác?

Có nhiều cách để truy cập package từ package bên ngoài, đó là:

Sử dụng tenpackage.*

Nếu bạn sử dụng **package.***, thì tất cả các lớp và interface của package này sẽ là có thể truy cập, nhưng không với các package con. Từ khóa import được sử dụng để làm cho các lớp và interface của package khác có thể truy cập tới package hiện tại. Ví dụ:

```
//Luu duoi dang A.java


package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//Luu duoi dang B.java


package mypack;
import pack.*;
```

```
class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

Sử dụng tenpackage.tenlop

Nếu bạn import tenpackage.tenlop, thì chỉ có lớp được khai báo của package này sẽ là có thể truy cập. Ví dụ:

```
//Luu duoi dang A.java

package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//Luu duoi dang B.java

package mypack;
import pack.A;

class B{
  public static void main(String args[]){
   A obj = new A();
   obj.msg();
  }
}
```

Sử dụng tên đầy đủ

Nếu bạn sử dụng tên đầy đủ, thì chỉ có lớp được khai báo của package này sẽ là có thể truy cập. Bây giờ bạn không cần import. Nhưng bạn cần sử dụng tên đầy đủ mỗi khi bạn đang truy cập lớp hoặc interface. Nói chung, nó được sử dụng khi hai package có cùng tên lớp, ví dụ: hai package là java.util và java.sql chứa lớp Date. Ví dụ:

```
//Luu duoi dang A.java


package pack;
public class A{
  public void msg(){System.out.println("Hello");}
}
//Luu duoi dang B.java


package mypack;
class B{
  public static void main(String args[]){
   pack.A obj = new pack.A();//Su dung ten day du
   obj.msg();
  }
}
```

Ghi chú: Nếu bạn import một package, thì các package con sẽ không được import.

Nếu bạn import một package, thì tất cả các lớp và interface của package đó sẽ được import ngoại trừ lớp và interface của package con. Vì thế, bạn cũng cần import cả các package con.

Package con trong Java

Package mà bên trong package khác thì được gọi là package con (subpackage). Ví dụ: Sun Microsystem đã định nghĩa một package có tên là java chứa nhiều lớp như System, String, Reader, Writer, Socket, … Những lớp này biểu diễn một nhóm cụ thể, ví dụ như các lớp Reader và Writer là cho hoạt động I/O, các lớp Socket và ServerSocket là cho lập trình mạng, …. Vì thế, Sun đã lại phân loại java package thành các subpackage như lang, net, io, … và đặt các lớp liên quan tới IO vào io package, …

Ví dụ về subpackage

```
package com.vietjack.core;
class Simple{
  public static void main(String args[]){
   System.out.println("Hello subpackage");
  }
```

```
}
```

Để biên dịch: javac -d . Simple.java

Để chạy: java com.vietjack.core.Simple

Cách gửi class file tới thư mục hoặc drive khác?

Giả sử một tình huống, bạn muốn đặt class file của A.java source file trong thư mục classes của c: drive. Ví dụ:

```
//Luu duoi dang Simple.java


package mypack;
public class Simple{
 public static void main(String args[]){
    System.out.println("Chao mung den voi package");
   }
}
```

Để biên dịch: e:\sources> javac -d c:\classes Simple.java

Để chạy: Để chạy chương trình này từ thư mục e:\source, bạn cần thiết lập classpath của thư mục, nơi mà class file ở đó.

- e:\sources> set classpath=c:\classes;.;
- e:\sources> java mypack.Simple

Cách khác với -classpath switch

Bạn có thể sử dụng –class switch với javac và java tool. Để chạy chương trình từ thư mục e:\source, bạn có thể sử dụng –class switch của java mà nói cho nó biết nơi để tìm class file. Ví dụ:

e:\sources> java -classpath c:\classes mypack.Simple

Cách để tải class file hoặc jar file

Cách để tải class file hoặc jar file:

- Tạm thời: bởi thiết lập classpath trong command prompt hoặc bởi –classpath switch.

- Vĩnh viễn: bởi thiết lập classpath trong biến môi trường hoặc bởi tạo jar file, chứa tất cả class file, và sao chép jar file trong thư mục jre/lib/ext.

Qui tắc: Chỉ có một lớp public trong một java source file và nó phải được lưu trữ bởi tên lớp public.

```
//Luu duoi dang C.java neu khong se gay ra Compilte Time Error


class A{}
class B{}
public class C{}
```

Cách đặt hai lớp public trong một package?

Nếu bạn muốn đặt hai lớp public trong một package, bạn có hai java source file chứa một lớp public nhưng giữ tên package là giống nhau. Ví dụ:

```
//Luu duoi dang A.java


package vietjack;
public class A{}
//Luu duoi dang B.java


package vietjack;
public class B{}
```

Thiết lập biến hệ thống CLASSPATH trong Java

Để hiển thị biến CLASSPATH hiện tại, bạn sử dụng các lệnh sau trong Windows và UNIX (Bourne shell):

- Trong Windows -> C:\> set CLASSPATH
- Trong UNIX -> % echo $CLASSPATH

Để xóa các nội dung hiện tại trong biến CLASSPATH, bạn sử dụng:

- Trong Windows -> C:\> set CLASSPATH=
- Trong UNIX -> % unset CLASSPATH; export CLASSPATH

Để thiết lập biến CLASSPATH:

- Trong Windows -> set CLASSPATH=C:\users\jack\java\classes
- Trong UNIX -> % CLASSPATH=/home/jack/java/classes; export CLASSPATH