JAVA.UTIL.SERVICELOADER.ITERATOR METHOD

Description

The **java.util.ServiceLoader.iterator** method lazily loads the available providers of this loader's service. The iterator returned by this method first yields all of the elements of the provider cache, in instantiation order. It then lazily loads and instantiates any remaining providers, adding each one to the cache in turn.

Declaration

Following is the declaration for **java.util.ServiceLoader.iterator** method

```
public Iterator<S> iterator()
```

Parameters

- **NA**

Return Value

This method returns an iterator that lazily loads providers for this loader's service

Exception

- **NA**

Example

In order the service to be registered, we need a META-INF/service folder in our classpath. In this particular folder, we need a text file with the name of the interface we implementing containing a single line listing the concrete class name of the implementation. In our case the name of the file is **com.tutorialspoint.ServiceProvider** and contains this line :

```
com.tutorialspoint.ServiceImplementation
```

Our service code is the following :

```
package com.tutorialspoint;

public class ServiceImplementation extends ServiceProvider {

   public String getMessage() {
   return "Hello World";
   }
}
```

The following code loads the service that is registered and uses it to get the message from the service:

```
package com.tutorialspoint;

import java.util.Iterator;
import java.util.ServiceLoader;

public abstract class ServiceProvider {

   public static ServiceProvider getDefault() {

   // load our plugin
   ServiceLoader<ServiceProvider> serviceLoader =
```

```
   ServiceLoader.load(ServiceProvider.class);

   // load the available providers of this loader's service.
   Iterator<ServiceProvider> iterator = serviceLoader.iterator();

   // check if there is a provider
   System.out.println("Iterator has more provider:" + iterator.hasNext());

   // use the provider to get the message
   System.out.println("" + iterator.next().getMessage());

   // checking if load was successful
   for (ServiceProvider provider : serviceLoader) {
   return provider;
   }
   throw new Error("Something is wrong with registering the addon");
   }

   public abstract String getMessage();

   public static void main(String[] ignored) {
   // create a new provider and call getMessage()
   ServiceProvider provider = ServiceProvider.getDefault();
   System.out.println(provider.getMessage());

   }
}
```

Let us compile and run the above program, this will produce the following result:

```
Iterator has more providers:true
Hello World
Hello World
```