

JAVA.UTIL.SERVICELOADER.LOAD METHOD

http://www.tutorialspoint.com/java/util/serviceloader_load_loader.htm

Copyright © tutorialspoint.com

Description

The **java.util.ServiceLoader.loadClass** < S > *service*, *ClassLoader* *loader* method Creates a new service loader for the given service type and class loader.

Declaration

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

```
public static <S> ServiceLoader<S> load(Class<S> service, ClassLoader loader)
```

Parameters

- **service** -- The interface or abstract class representing the service
- **loader** -- The class loader to be used to load provider-configuration files and provider classes, or **null** if the system class loader or, *failingthat, thebootstrapclassloader* is to be used

Return Value

This method returns a new service loader

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.ServiceLoader;

public abstract class ServiceProvider {

    public static ServiceProvider getDefault() {

        // load our plugin with the default system class loader
        ServiceLoader<ServiceProvider> serviceLoader =
```

```

ServiceLoader.load(ServiceProvider.class,
ClassLoader.getSystemClassLoader());

//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:

Hello World

Loading [MathJax]/jax/output/HTML-CSS/jax.js