Reusing the Legacy Code Based on the LC-WS Method

Yuan ZHAO\(^1\) and Ke-Sen WANG\(^2\)

\(^1\)Naval Aeronautical University, Yantai Shandong 264001, China
Email: ascendtop@126.com

\(^2\)Ludong University, Yantai Shandong 264001, China
Email: wangkensen1996@qq.com

**Keywords:** Legacy code, Web Service, JNI, Reuse.

**Abstract.** Since there is currently a massive amount of legacy code in the old system, a new method, LC-WS for the purpose of reusing the legacy code, is presented in this paper. The legacy code is wrapped and published into Web Services which can be accessed and invoked by clients. By the method of LC-WS, a lot of legacy code can be reused at lower cost. The obtained result can not only shorten the research period, but also lower exploitation risk. Many experiments have proven this method feasible.

**Introduction**

Currently, there is a number of applications in different fields are still coded by traditional programming language. Even though, most of these applications cannot be directly run in the new system, they still possess great practical value. The code for these application routines are called ‘LC’ (Legacy Code)[1]. This paper emphasizes the idea of reusing these Legacy Code in some constructed and in-service information platforms through the method of WS (Web Service).

**Web Service**

Web Service is an open and distributive application routine model. The usage of Web Service can be independent from platforms. Clients can construct logical units of application routine that can be accessed via internet from the standardized Web Service agreement[2]. Certain organizations have already published some standards and agreements that are necessary to fulfill Web Service such as Soap, WSDL, UDDI, etc. Web Service aims at providing access of service and information on the internet and achieving cross-platform interoperability[3].

The fundamental structure of Web Service is based on provider, service requester, and service agent. Web Service is also defined as three actions which are service publication, service discovery, and the binding between service provider and service requester.

Provider: the entity that provides service, waiting for providing its own function to other services or clients.

Requester: the user of service, sending service request to provider.

Agent: the administrator that connects proper service provider with service requester.

Web Service is, therefore, the concrete service that is deployed on a particular provider server. Once Web Service is constructed, any language and all clients from different platforms can access its WSDL file to invoke its particular services[4].

**Building Information Integration Platform Based on Web Service**

In this paper, the information integration platform constructed by Web Service consists of information registration center, information provider, and information requester, as shown in figure 1. The nodal point of information registration center realizes information registration, information inquiry, and information discovery. Information requester can discover the provider’s information in information registration center. Then, the information requester can obtain the information through the ISS (Information Sharing Service) service of information provider’s nodal point. The process is shown below:
information provider publishes information in information registration center and write the meta-message of the information into local registry.

(2) information requester can seek for his or her ideal information in information registration center and acquire the corresponding url, r_id, t_id of the information that he or she is interested in.

(3) Information requester can fill in the JPS technical information inquiry page and provider’s ISS with the url, r_id, t_id obtained from the information registration center.

(4) At the same time when the information provider gets the requester from the information requester, information provider can extract and wrap up information from its isomeric data bases. These selected information will be processed into standard-XML-form information file. In the enterprise information system Loglo, information is uniformly processed into standard-REML-form information file.

(5) Information provider returns the processed standard-XML-form information file or REML form to the service requester.

After the information requester obtains the REML-form information, requester can save it as XML document file and name it as r_id#t_id.xml. Requester will then have to display the standard-REML-form information in a friendly interface to final clients based on XML interpretation techniques such as JDOM in his or her browser.

**JNI Technology**

JNI stands for Java Native Interface, a part of JDK[5], which has been employed into standard Java platform. It is convenient to achieve the cross-platform ability of code through JNI coding. JNI allows the Java code that runs on Java Virtual Machine to control the application routine and bases coded by other language. Using the method provided by JNI, Java can directly interact with binary bases of certain control systems and hardware platforms[6]. This interaction process, happening in the process of JVM, allows the invocation of Java method which is Java-type and marked as ‘local’ to be mapped to corresponding functions in the sharing binary base and load these two objects into the same process space.

JNI enables the local usage of Java objects to act like Java code itself. JNI can fulfill all of the present invocation methods of Java such as delivering the necessary parameters[7] and returning the result. The basic types in JNI and the basic local types shown in figure 2.

![Figure 1. The System Structure.](image-url)
<table>
<thead>
<tr>
<th>Java type</th>
<th>Local type</th>
<th>Digit</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>jboolean</td>
<td>8, unsigned</td>
</tr>
<tr>
<td>byte</td>
<td>jbyte</td>
<td>8</td>
</tr>
<tr>
<td>char</td>
<td>jchar</td>
<td>16, unsigned</td>
</tr>
<tr>
<td>short</td>
<td>jshort</td>
<td>16</td>
</tr>
<tr>
<td>int</td>
<td>jint</td>
<td>32</td>
</tr>
<tr>
<td>long</td>
<td>jlong</td>
<td>64</td>
</tr>
<tr>
<td>Float</td>
<td>jfloat</td>
<td>32</td>
</tr>
<tr>
<td>double</td>
<td>jdouble</td>
<td>64</td>
</tr>
<tr>
<td>void</td>
<td>void</td>
<td>n/a</td>
</tr>
</tbody>
</table>

Figure 2. The corresponding relationship.

**The Principle of LC-WS method**

The process of putting the Legacy Code in service includes: the preconditioning of Legacy Code, the packaging of Wrapper, the formulation of the bases of dynamic link, and legcode service invocation.

LC-WS method packages analyzed Legacy Code through JNI and forms dll dynamic-link base. Then the main service type will invoke this base. The operational process is shown in figure 3 (notice a.dll and b.dll in fig.3 are formulated after the packaging of Legacy Code via JNI).

![Fig 3 Operational Principle](image)

The Legcode service is fulfilled through Java type and through legcode type one can invoke middle types such as a.java which truly attains the invocation of dynamic-link base.

The key code is shown below:

```java
public class legcode{
    public String lge1(String par)
    { return aa(par) // invoke the dynamic-link base of a type
    }
}
```
The Procedures of LC-WS method

The following steps can package the legacy code into web service through JNI:

(1) Programming legcode type, deploy it to Web service, as shown in figure 4. 
legcode type is the main service type that Web clients invoke. There is no alternative method under the initial condition. Other methods will become available only if the Legacy Code is packaged by JNI.

(2) Programming Java type based on simple analysis of Legacy Code (This type contains the necessary local invocation method).

Analyzing the Legacy Code is the precondition of programming local code. In the preconditioning process of Legacy Code, the technique of routine microtomy can simplify Legacy Code and clarify its structure. Also, the programming of local type is different from normal Java type. The declaration of local type must include identifier ‘native’ to indicate that this method is achieved through a different language.

(3) Packaging Legacy Code with JNI includes:

① Ordering programming steps via javac (2) form Java type in the middle
② Formulating header file via javah including the declaration of function prototype achieved by local method.
③ Programming local-method in C language.
④ Editing local functions of C and formulating local link base.

(4) Programming method in legcode type and invoke the Java type formulated in step (2).
Conclusion

LC-WS method is open to Legacy Code that is concentrated and inaccessible to the current platform. LC-WS successfully integrate these Legacy Code into internet systems based on Web Service. One can avoid the inconvenience of multiple deployments by adding dynamic-link base that is formed by the invocation of Legacy Code through Java in an already deployed legcode service [8]. To transform Legacy Code to Web Service, local registrant only needs to have background in basic C language. However, the research area of this paper is only suitable for the C codes that have a returning value. As for those who do not have a returning value, they will be the future area of research.

Acknowledgment

Thanks for all the support my colleagues gave me in this paper’s writing.

References


