Design and Research of Data Service Platform Based on RESTful

Airong Yu a, Peiyi Yu, Jun Lai, Yanjie Niu
The Army Engineering University of PLA Nanjing 210007, China
a intraweb@163.com,

Abstract. Based on the analysis of research on existing data service technology, in view of the existing problems of data service release and data access efficiency. In this paper, a design scheme of test data service based on RESTful is proposed. The system implements abstract description and encapsulation for typical application scenario, real-time storage, which applies JSON data exchanging and releases REST service with standard interface. Application result shows that the proposed Data Service Platform Based on RESTful scheme benefits a lot in resource sharing and service extension in Experimental Data Management System.

1. Introduction
With the rapid development of information and data construction, many enterprise and government have achieved rapid development in the network infrastructure, construction of application system and data environment construction, but limited to the development and construction of the overall planning, leading to the application system independent act of one's own free will. At the same time, because of the different age and the establishment of techniques used in the data distributed in each application system, specifications and shapes data, both structured data stored in relational databases, XML, Excel or other files[1,2].

Web technology has developed very rapidly in the past decade, and has already penetrated into our daily life, such as medical treatment, transportation, education and other industries everywhere. Because of the increasing demand for information access, the importance of Web technology is also growing. Data service is produced in order to solve this problem. Data service, as the name suggests is to provide data as a service to the user, the user and the business data needed by the system can be accessed through a unified way. In the system architecture model of data service is located between the underlying data source and application systems. Service Oriented Architecture (Service-Oriented Architecture, SOA) is the service structure of a business driven, coarse-grained, loosely coupled, suitable for mutual connection, reusable services integration. We can realize flexible extension of data service platform using service oriented architecture.

2. Technical concept
2.1. Data Service
In research of data integrated access and sharing, in the form of packaging service operations on the data, providing various data services to the user or application system was first proposed by BEA in 2005, this system established the data service layer for unified management of data [3]. Then the researchers at home and abroad to implement data service platform for different uses all sorts of architecture and technology.
Garlic System, a data service system developed by IBM Company, is mainly for large-scale heterogeneous multimedia information data. The system will be a variety of data stored in the data warehouse is different, using a query processing "middleware" to achieve the statement conversion and execution of the data query, to achieve datasharing service.

Lore (Light Weight Object Repository), consists of data service system designed by America Standford University, mainly for semi-structured data to provide information sharing and management, including OEM (Object Exchange Model) semi structured data and XML data. Lore provides data services including for semi structured data storage, query and update and some other data operations tool [4].

HDIM (Heterogeneous Data Integration Middlew), a heterogeneous data integration middleware design at Northwestern University, the global ontology and local ontology to describe the data information, and use the XML language to describe the mapping relationship of ontology, to realize the communication between the data source and the middle part by adopting Web services, add, delete and modify dynamically so as to complete the data source.

2.2.  JSON
JSON [5] is designed to be a data exchange language which is human readable and easy for computers to parse and use. JSON is directly supported inside JavaScript and is best suited for JavaScript applications; thus providing significant performance gains over XML, which requires extra libraries to retrieve data from Document Object Model (DOM) [6] objects. JSON is estimated to parse up to one hundred times faster than XML [7] in modern browsers, but despite its claims of noteworthy performance, arguments against JSON include lack of namespace support, lack of input validation and extensibility drawbacks. Crockford addresses such arguments by claiming that “every object is a namespace. Its set of keys is independent of all other objects, even exclusive of nesting. Also, JSON uses context to avoid ambiguity, just as programming languages do,” that validation of inputs is the responsibility of individual domain applications, and that the lack of extensibility claims is addressed by the flexibility of JSON constructs. JSON’s syntax is human readable. Figure 1 describes an example where JSON is used to encode a firstname and a lastname.

{  "firstname" : "Tom"  "lastname" : "Hank"  }

Figure 1. A simple JSON construct describing the encoding of a name

2.3.  RESTful
RESTful [8] architecture is a kind of Internet software architecture. Its structure is clear and easy to understand. It is being adopted by more and more website systems. RESTful architecture is an Internet software architecture principle first proposed by Dr. Roy Thomas Fielding. REST is the abbreviation of Representational State Transfer. Dr. Roy Thomas Fielding is the main designer of HTTP protocol (version 1.0 and version 1.1).

The basic principles of this architecture [8] are described as follows:
- Point any resource on the Internet (including text, pictures, videos, songs, services, etc.) to it with a URI (Unified Resource Locator). Since resources represent an entity, URI should use nouns instead of verbs when naming. For example:
  
  http://api.website.cn/lists.

- Four operation methods in HTTP protocol (GET, POST, PUT and DELETE) are used to access network resources. GET is used to obtain resources (one or more), POST is used to create new resources (or to update resources), PUT is used to update resources, and DELETE is used to delete resources.

- The representation of resources requires client and server to negotiate through HTTP protocol, that is, client needs to request a specific resource format acceptable to client through Accept header, and server needs to tell client about the resource format sent by client through Content-Type header.
3. Data Service platform Based on RESTful Architecture

Based on the analysis of various data of the Experimental Data Management System, we found that data sources is not only a simple type of structured data (mainly stored in the database in the system), many data sources is semi-structured and unstructured data (such as simulation script, various documents, test planning documents etc.). In the implementation of unified management of data and access, we faced with the key problem of data heterogeneity, distribution and how to achieve a transparent access to data [9].

Data service platform should meet the management and access to test data, and support to the stage of data analysis. Any person registered in the platform of data resources available to carry on the effective management, and on this basis can provide a variety of customizable data service. Figure 2 describes platform application framework.

For text resources, HTML, XML, JSON and other formats can be used. For image resources, PNG or JPG formats can be used.
Combined with AJAX technology, it can read asynchronously and refresh the information and data requests of user browsers, thus reducing the data exchange volume of database servers.

By transforming heterogeneous data from different forms of networks and unifying data interfaces, the system realizes the Web integration application of experimental data resources. The REST services of typical application scenarios are published as follows:

- Topology real-time monitoring information of equipment in test network: Topology networking status of main monitoring equipment, equipment status information, etc.
- Summary and comprehensive analysis of test results: summary of test results; query and comparison of single test data; horizontal comparison of multiple test data, etc.
- Test Control Instruction Management: Instruction Execution and Feedback Information Management in Test Process.
- Management of Abnormal Information in Test: Management of Abnormal Information and Abnormal Disposal Information in Test.

According to the design principle of REST architecture, any data is recognized as a resource as long as it is necessary to be referenced. This requires that a unique URI be allocated to it in order to realize the one-to-one correspondence between the attributes of data resources and URIs. URI design rules are expressed in two ways. One is to add query parameters after URI to define a URI, such as "/search?Param1="type"&Param2="id";The other is to set it by hierarchical path variables, such as "/elements/\{type\}/\{id\}/..." The parameters in "\{"" indicate that they are variables.

Considering that only the service interface of business logic can be mapped through URI, and the extension of corresponding file and variable code of system architecture can be hidden, this system adopts the second way to design URI.

Although the REST services provided by the experimental data management system are different, they can be abstracted into different data modes from the attributes of spatial dimension, time dimension and type of test equipment. Each mode corresponds to a unique URI, and its typical corresponding relationship is shown in Table 1.

| Data Abstract mode                                      | URI                    |
|---------------------------------------------------------|------------------------|
| Data on the state of all devices at a given time         | .../equipments/time    |
| Data on the state of a certain type of equipment at a given time | .../equipmentall/time/type |
| State data of a device at a given time                   | .../equipments/id/time  |
| Result data of a device at a given time                  | .../equipments/id/time  |
| Data of abnormal information in a certain period of time | .../errordata/start&end |

4. JSON parsing in Java using Jackson parser

Parsing JSON files in Java can be accomplished quite well using the Jackson Java JSON parser (http://jackson.codehaus.org). The main thing to keep in mind is that you need to define Java classes that match your data. Then, you can use the data binding available with the ObjectMapper class to read the JSON data directly into the Java class objects that you define. As an example of how this is done, consider the following JSON file, “equipments.json”:

```json
{
  "equipments": [
    {"name":" Test terminal",
    "address":"127.0.8.89",
    "frequency":"30MHz",
    "description":" Test terminal is a test equipment. "},
    {"name":" Handheld devices",
    "address":"20.9.9.1",
    "frequency":"87.9MHz",
    "description":" Handheld devices is a phone. "}]
}
```
This JSON file consists of a key:value pair. The “equipments” key has a value that is an array (or ArrayList). Each element in the array is an associative array with information on an Open Source project.

5. Modeling data with Java classes

The first step is to recognize what kind of Java classes we need to use to model this JSON data. In this case, it is easier to work your way from the inside and move out. If you look inside the information for an Open Source project, one of the objects that can be seen is the “description”. We can use an ArrayList<String> for the description. The other keys point to Strings. Let's define a Project class that can store the “name”, “address”, “frequency”, and “description”.

```java
import java.util.ArrayList;
public class equipments {
    private String name;
    private String address;
    private String frequency;
    private ArrayList<String> description;
    public ArrayList<String> getDescription() {
        return description;
    }
    public void setDescription(ArrayList<String> desc) {
        description = desc;
    }
    public void setName(String s) {
        name = s.trim();
    }
    public void setAddress(String s) {
        address = s.trim();
    }
    public void setFrequency(String s) {
        frequency = s.trim();
    }
    public String getName() {
        return name;
    }
    public String getAddress() {
        return address;
    }
    public String getFrequency() {
        return frequency;
    }
    public String toString() {
        String temp = " {
" + name + ":" + address + ":" + frequency + "\n"
        + "description": ["" + description.get(i) + "\n"
        + (i < description.size() - 1) {  
            temp += "]n";
            temp += ",\n"
        if (i < description.size() - 1) {  
            temp += ""]n";
            temp += "\n";
            return temp;
    }
```

Just as would be the case for a Java bean, no constructor is needed. Here is the source code for “equipments.java”. This is a kind of container class for equipments Objects.
import java.util.HashMap;
import java.util.ArrayList;

public class equipments extends HashMap<String, ArrayList<Project>>{
    
    The Projects class definition is very simple. All you need to do is make this class inherit from HashMap<String, ArrayList< equipments >>. No constructor or any other methods are required.

6. Conclusion
Aiming at the problem of inconsistency of data interface of various test equipment in test data management, this paper presents a design scheme of test data management system based on REST, which abstracts the data provided by equipment, instrument and terminal in test environment into data modes of different granularity. Through the establishment of a reasonable and feasible system architecture and data service interface, the exchange and transmission of test data can be realized through JSON, and the test data can be encapsulated and released as standard REST service, which provides technical support and basic platform for the development and utilization of test data.

References
[1] Fielding R T. Architectural styles and the design of network—based Software architectures[D].University of California,Irvine,2000.
[2] U.Hilger, “Article: Client/server with java and xml-rpc,” 2005.http://www.w3.org/TR/2006/REC-xml-20060816
[3] S.Klarr,”Javascript:What is json?,” 2007. http://www.scottklarr.com/topic/18/javascriptwhat-isjson
[4] Li N,Cai HI Functionality semantic indexing and matching method for RESTful web Services based on resource state descriptions[C].Computer Science and Engineering, 2009. WCSE’ 09. Second International Workshop on.IEEE,2009,2:371—375.
[5] J. Bosak, “Xml, java, and the future of the web,” World Wide Web Journal, 2(4):219-227, 1997.
[6] JSON. json.org. http://www.json.org
[7] Feng X,Fan Y.Research on application of rdfa in restful web services[C].Internet Computing for Science and Engineering (ICICSE),2012 Sixth International Conference on. IEEE,2012:266—269.
[8] Liang T,Chen L,Ying H,et a1.Co-Clustering WSDL Documents to Bootstrap Service Discovery[C].Service—Oriented Computing and Applications(SOCA), 2014 IEEE 7th International Conference On.IEEE,2014:215-222.
[9] Kim C S.Building semantic ontologies for RESTful web services[C].2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM),2010:383—386.