XML-RPC

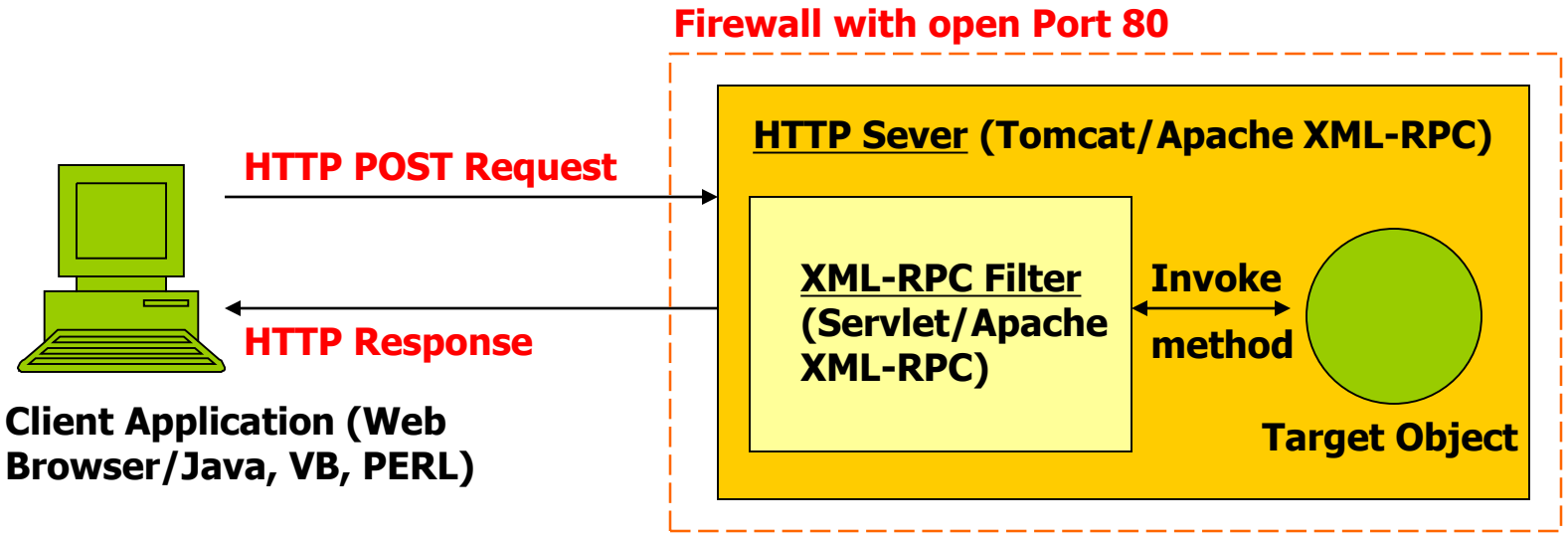
- *"Does distributed computing have to be any harder than this? I don't think so." – Byte Magazine*
- Provides an ***XML and HTTP-based mechanism*** for making method or function calls across a network.
- First published in early 1998 by UserLand Software (part of their Frontier product).
- A snapshot of the spec under consideration in 1998 became XML-RPC.
  - ***The rest of the spec went on to become SOAP.***
  - Has remained largely unchanged since.
  - Spec available from www.xml-rpc.com/spec.
  - ***Apache Java implementation available*** from xml.apache.org.
  - ***PERL implementation*** available from www.blackperl.com.
- Request and responses are described using a small XML vocabulary
- Note: JSON-RPC is a similar protocol to XML-RPC, but using JSON in place of XML

XML-RPC

- XML-RPC parameters are a simple list of types and content.
  - ***Structs and arrays*** are the most complex types available.
  - ***No notion of objects.***
  - ***No mechanism for including information that uses other XML vocabularies (e.g. Scalable Vector Graphics, Synchronized Multimedia Integration Language)***.
- A simple but effective inter-process communication mechanism.
- Can use for glue-code for integrating disparate applications and services on heterogeneous systems.
- Useful for publishing services:
  - Information recipients can be any kind of client that understands the XML-RPC interface.
  - E.g. O' Reilly Network' s Meerkat headline syndicator.

XML-RPC **Components**

- XML-RPC consists of three relatively small parts:
  1. ***XML-RPC Data Model:*** A set of types for use in passing parameters, return values and faults (error messages).
  2. ***XML-RPC Request Structures:*** A HTTP POST request containing method and parameter information.
  3. ***XML-RPC Response Structures:*** A HTTP response that contains return values or fault information.



XML-RPC ***Data Model***

- Defines ***six basic data types*** and ***two compound data types*** that represent combinations of types.
- More restrictive than most programming language types:
  - Enough however for many tasks.
  - Lowest common denominator approach.
- All types are represented by XML elements whose content provides the values:
- To define a string whose value is "Distributed Systems":
  - ***<string> Distributed Systems</string>***
- Basic types are always enclosed in *<value>* elements.
- Strings (**and only strings**) can be enclosed in *<value>* and ommit the data type.
- ***Basic types can be combined into two or more complex types – arrays and structs.***

Basic Data Types *in XML-RPC*

Type	Value	Examples
int or i4	32-bit integers between –2,147,483,648 and 2,147,483,647	<int>27</int> <i4>27</i4>
double	64-bit floating point numbers	<double>27.31415</double> <double>-.1465</double>
Boolean	True (1) or False (0)	<boolean>1</boolean> <boolean>0</boolean>
String	ASCII text, though many implementations support Unicode	<string>Hello!</string> <string>Crazy!!@</string>
dateTime.iso8601	Dates in ISO8601 format *CCYYMMDD*T*HH*:*MM*:*SS*	<dateTime.iso8601>20021125T02:20:04 </dateTime.iso8601>
base64	Binary information encoded in Base 64 (RFC 2045)	<base64>SVGsbG8sIFdvcmxkIQ</base64>

XML-RPC **Data Model**

- Arrays are indicated by the *<array>* element which contains a *<data>* element holding the list of values:

```
<value>
    <array>
        <data>
            <value><string>This</string></value>
            <value><string>is</string></value>
            <value><string>an</string></value>
            <value><string>array!</string></value>
        </data>
    </array>
</value>
```

- ***Note: Arrays elements do not have to be of the same data type…***

```
…<data>
    <value><int>24</int></value>
    <value><string>students in this class</string></value>
</data>…
```

XML-RPC **Data Model**

- Note: XML-RPC won't do anything to guarantee that arrays have a consistent number or type of values.
  - Need to ensure that your code consistently generates the right number and type of output values if consistency is necessary for your application.
- ***Structs contain unordered content identified by name.***
  - Names are strings (but don't have to be enclosed in a *<string>* element).
  - Each *<struct>* element contains a list of *<member>* elements.
  - *<member>* elements contain one *<name>* and one *<value>* element.
- Specification does *not require struct names to be unique*.
  - Also, order is not important.
- ***Structs can contain other structs and even arrays.***

XML-RPC ***Data Model***

- A sample struct might look like the following:

```
<value>
    <struct>
        <member>
            <name>firstName</name>
            <value>Joseph</value>
        </member>
        <member>
            <name>Surname</name>
            <value>Murphy</value>
        </member>
        <member>
            <name>Age</name>
            <value>28</value>
        </member>
    </struct>
</value>
```

XML-RPC ***Request Structure***

- XML-RPC requests are a combination of XML and HTTP headers.
  - ***The XML content uses the data-typing structure to pass parameters***.
  - Contains additional information ***identifying which procedure is being called***.
  - The ***HTTP headers provide a wrapper*** for passing the request over the web.
- Each request contains a ***single XML document*** with a root element called ***<methodCall>***.
  - Each ***<methodCall>*** element contains a ***<methodName>*** element and a ***<params>*** element.
  - ***<methodName>:*** The name of the remote procedure to be invoked.
  - ***<params>:*** a list of parameters containing <value> elements.
- ***Basically an ordinary HTTP request with a carefully constructed payload.***

Sample XML-RPC Request

```
POST / HTTP/1.1
Content-Length: 175
Content-Type: text/xml
User-Agent: Java1.4.0_02
Host: localhost:8898
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

<?xml version="1.0" encoding="ISO-8859-1"?>
<methodCall>
    <methodName>area.circleArea</methodName>
    <params>
        <param>
            <value>
                <double>7.0</double>
            </value>
        </param>
    </params>
</methodCall>
```

Library making call. Not browser…

Extended ASCII (8-bit) Latin-1 etc..

XML-RPC ***Response Structure***

- Assuming that the remote procedure was found, executed correctly and returned results:
  - The XML-RPC response looks much like a request.
  - A ***<methodResponse>*** forms the root element of the response XML payload.
  - ***A response can contain only one parameter*** (enclosed in a *<params>* element).
  - A *<params>* element can contain an array or struct (makes it ***possible to return more than one value***)
- *What about void methods? Still have to return something…*
  - Maybe a boolean (1) or a "success value".
- If there was a problem, a ***<fault>*** element is returned (with an enclosing ***<value>*** element).
  - Error codes vary between implementations.
  - ***<fault> element can contain only a single value***.

Sample XML-RPC Response

HTTP/1.1 200 OK
Server: Apache XML-RPC 1.0
Connection: close
Content-Type: text/xml
Content-Length: 158

Response code always 200 - Ok

Our server is the Apache XML-RPC server

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<methodResponse>
    <params>
        <param>
            <value>
                <double>153.93804002589985</double>
            </value>
        </param>
    </params>
</methodResponse>
```