REST web services and Google Protocol Buffers

Prasad Nirantar

BMC Software

6TH IndicThreads.com Conference On JAVA

2,3 DEC 2011

PUNE, INDIA

Agenda

- REST concepts
- Introduction Google Protocol Buffers (GPB)
- Details of Google Protocol Buffers messages
- Demo
- Performance Comparison with other serialization techniques
- Conclusions

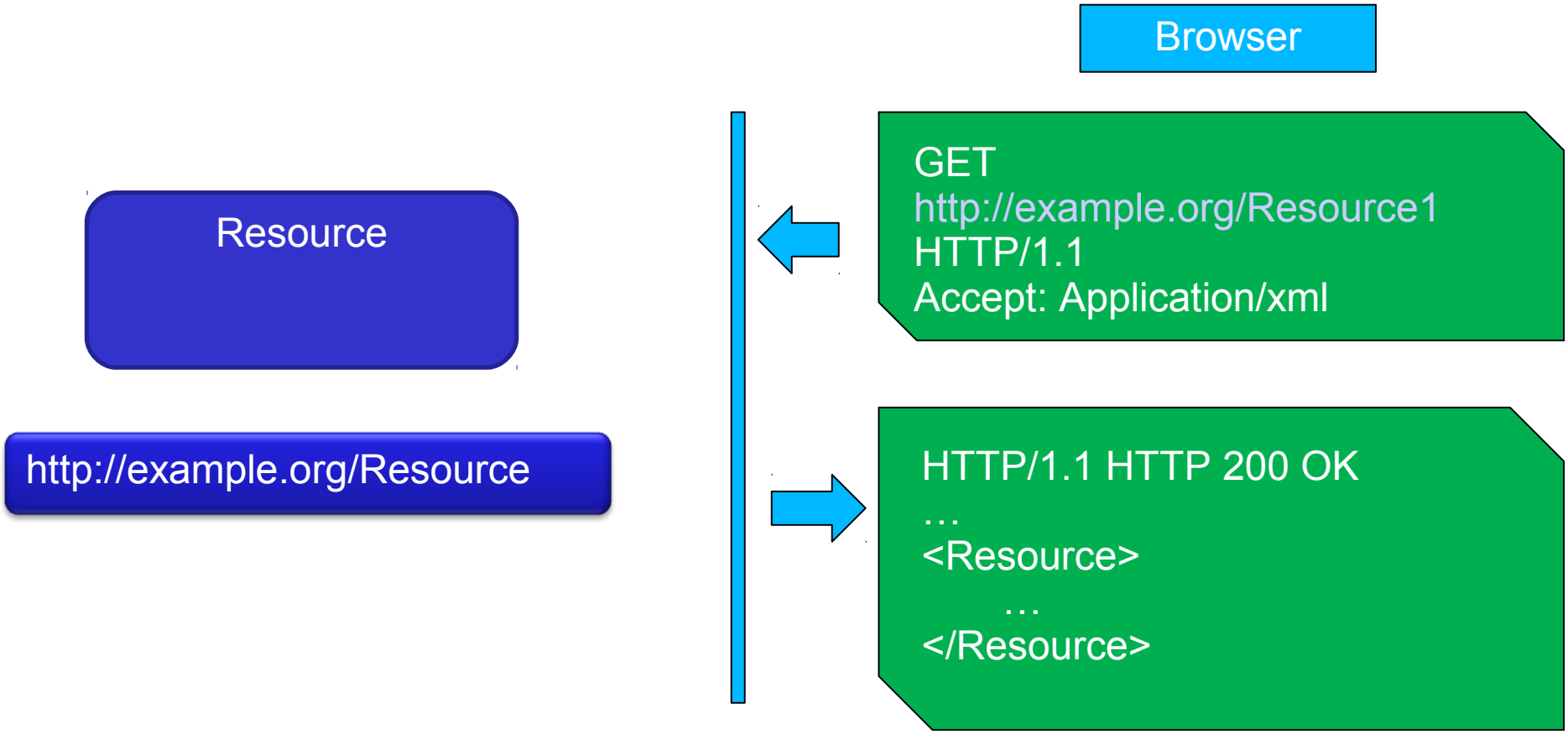
REST (Representational State Transfer)

- Identifies principles why web is so prevalent and ubiquitous
- Representational State Transfer
  - Architectural style for building loosely coupled systems
  - Architectural style – General principles informing/guiding creation of an architecture

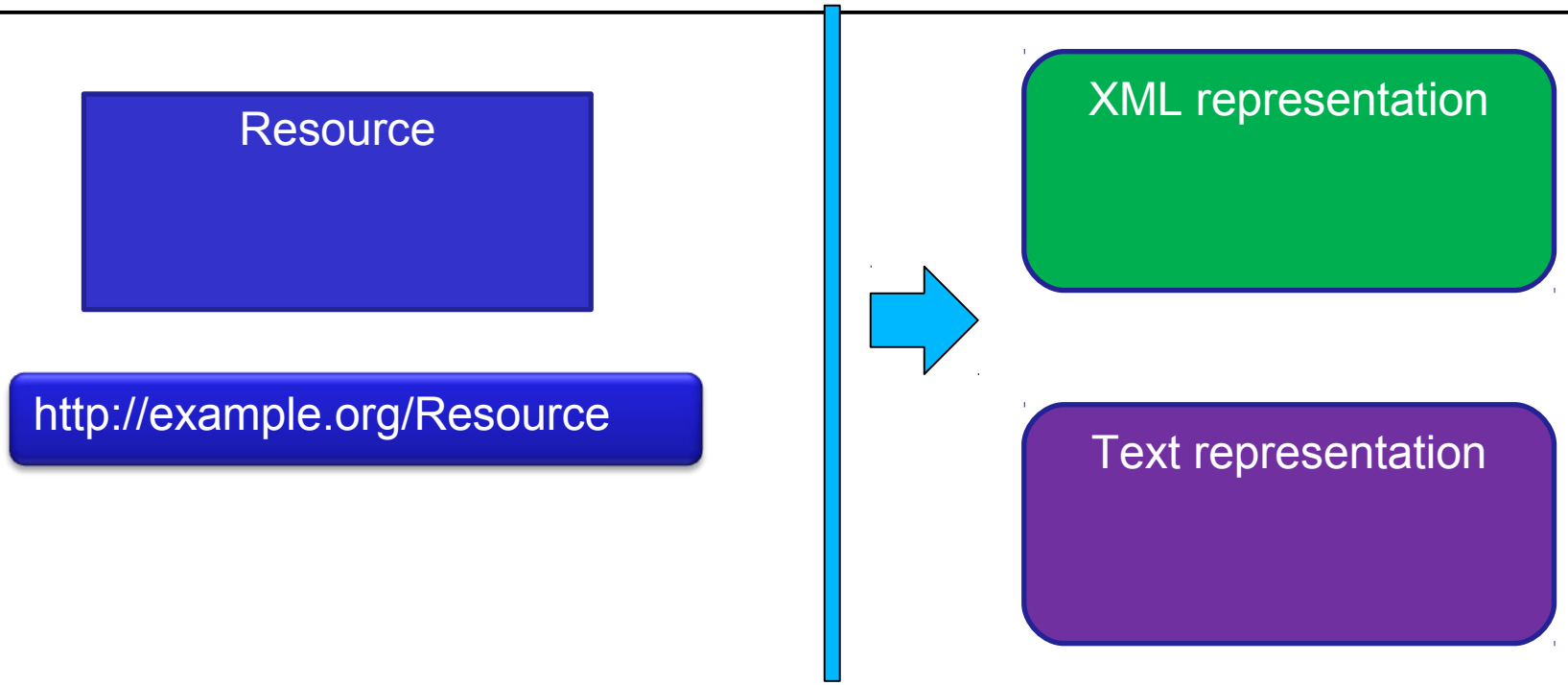
REST Style Principles

- Resource identification
- Uniform interface
- Self Describing messages
- Hypermedia driving application state
- Stateless transactions

Resource and its Representation



Multiple Resource representation with single URI

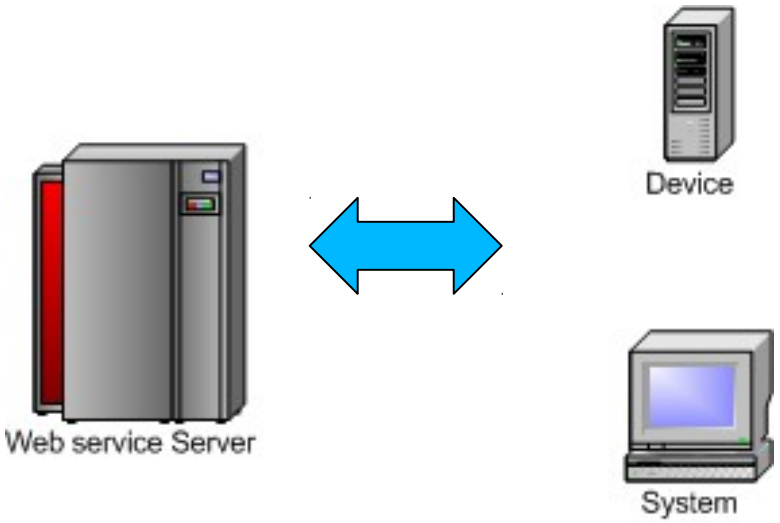


Content negotiation

- Content selection
- Different types of content supported
  - JSON
  - XML
- A resource can give response in a representation as per the request

Need for efficient mechanism

- Data intensive systems
- Efficient mechanism to send/receive data



Problems with XML

- Verbose
- Tends to be inefficient

Solution - Use of Binary content

Problems it tries to solve

- Performance (with Interoperability and Portability)

Examples

- Google Protocol Buffers
- Avro

Introducing Google Protocol Buffers

- Language neutral, platform neutral, extensible way of serializing structured data
- Flexible, efficient, automated
- Supported languages - Java, C++, Python
- Developed and used by Google
- GPB project has BSD license

Protocol buffer messages

Message –

- Specify structure and details of the information being serialized.
- Each protocol buffer message is a small logical record of information, containing a series of name-value pairs.

Details: .proto Message

- Data is hierarchically structured in messages
- Each message has one or more uniquely numbered fields
- Each field has name and value
- Value types can be
  - numbers (integer or floating-point)
  - boolean
  - strings,
  - raw bytes
  - other protocol buffer message types,

Details: .proto Message

Fields types

- Required
- Optional
- Repeated

Tag

- Unique number with each field in message
- From 1-15 take 1 byte to encode (for frequent)
- 16-2047 take 2 bytes and so on
- Smallest 1 largest is 2^{29} -1

.proto file example

```
message Person {
    required string name = 1;
    required int32 id = 2;
    optional string email = 3;
    enum PhoneType { MOBILE = 0; HOME = 1; WORK = 2; }
    message PhoneNumber {
     required string number = 1;
     optional PhoneType type = 2 [default = HOME];
    } repeated PhoneNumber phone = 4;
}
```

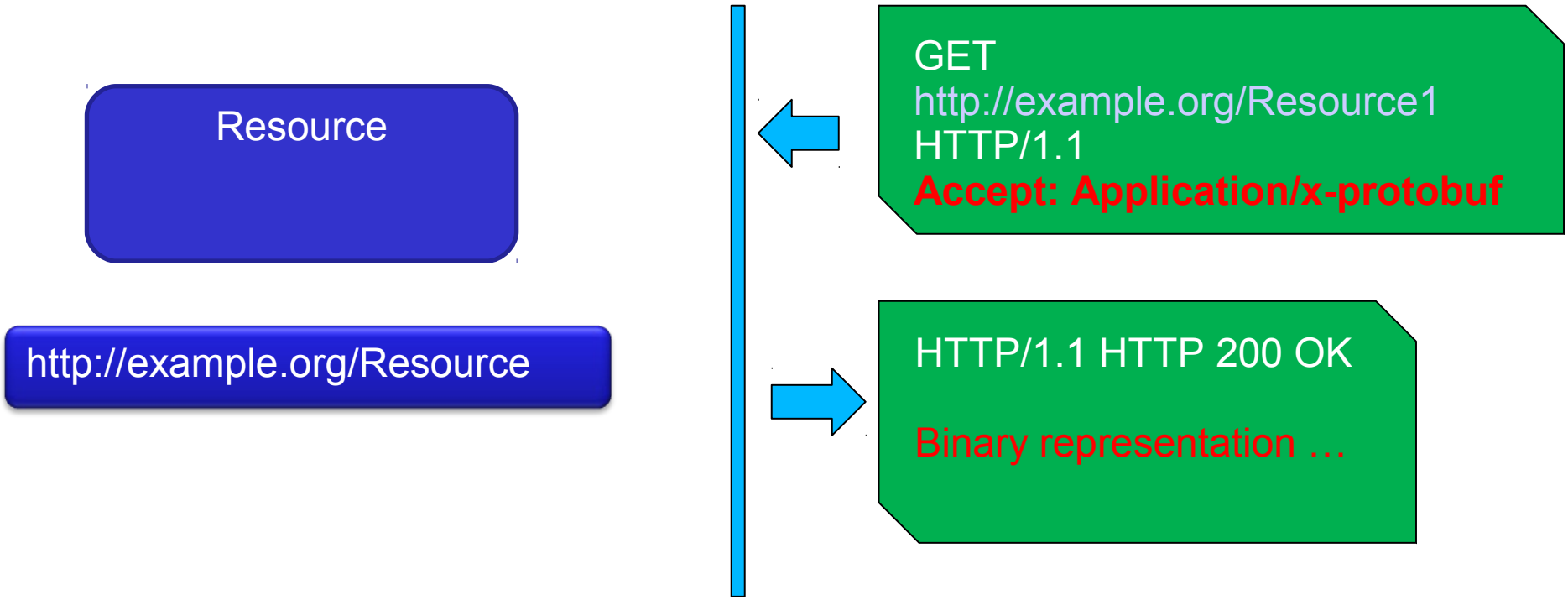
Compilation of .proto messages

- A “protoc” compiler should be downloaded first.
- From .proto file, the PB compiler generates code of your choice of language from -Java, C++, Python
- For Java – The compiler generates Java classes containing POJO and Builder
- Parsing and serialization API is available

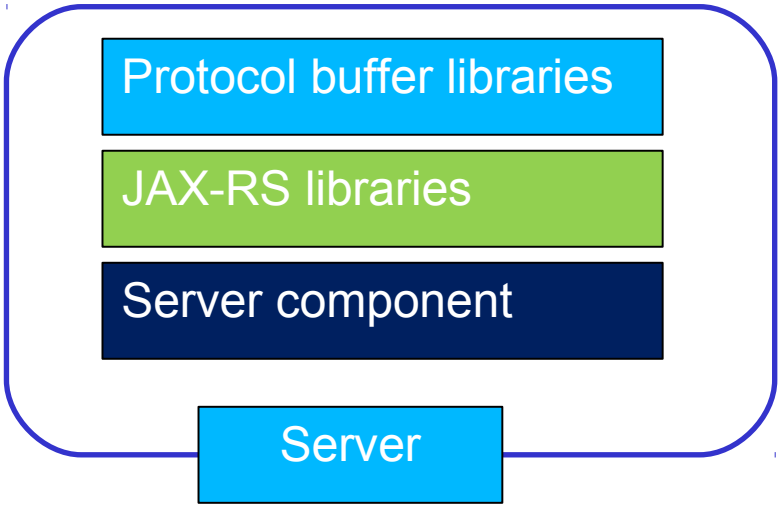
Protocol Buffer and REST

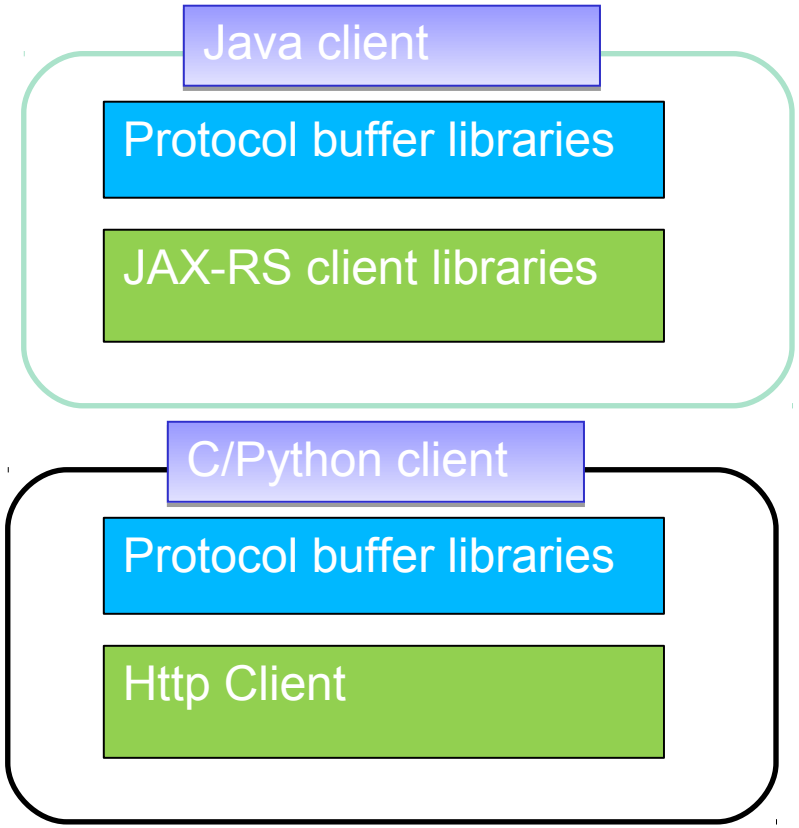
- Suits for the cases where you want to optimize on performance and size of data transferred
- Use the protocol buffer as a content
- Adding new MIME types for JAX-RS is easy
- The examples in demo are based on JAX-RS

Resource & Representation-protobuf



Stack - REST with GPB





Demo

Alternatives for binary content

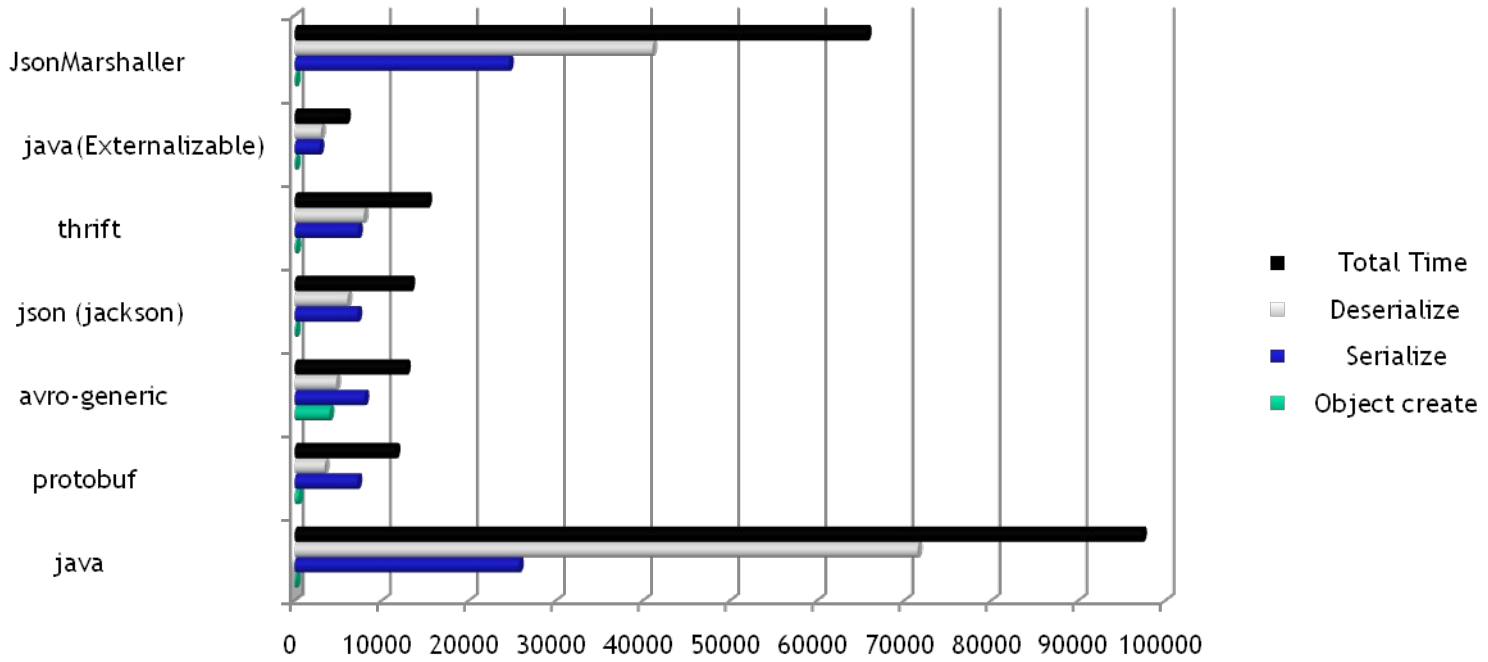
- Use of Java serialized objects
- Use of AVRO
  - Similar data format by Apache
  - Uses JSON object in string format to describe schema
  - Compatible with C, Java, Python

Comparison - GPB vs XML

- Where GPB wins
  - 3 to 10 times smaller
  - 20 to 100 times faster
  - Less ambiguous
  - Generate data classes for easy use
- Where XML wins
  - Humanly readable and editable
  - A good markup choice

Comparison : serialization techniques

technology	Object create	Serialize	Deserialize	Total Time	Serialized Size
java	169	25773	71574	97347	919
protobuf	471	7227	3479	11598	231
avro-generic	4024	8019	4779	12798	211
json (jackson)	176	7224	6084	13308	378
thrift	228	7303	7949	15252	353
java(Externalizable)	170	2874	3043	5917	264
JsonMarshaller	171	24618	41115	65733	370



Disadvantages of protocol buffers

- Careful with type of fields (required or optional) for backwards compatibility
- Debugging concerns
- Not humanly readable

Conclusions

- With the JAX-RS the newer data serialization formats can be plugged into REST
- Useful for the data intensive operations, in such cases performance benefits can be achieved
- Not to be used when humanly readable formats/markups are desired

References

Developer Guide - Protocol Buffers - Google Code

Rest in Practice – Webber, Parastaridis, Robinson Oreilly Publication

Restful Web Services : Principles, patterns, emerging technologies – Erik Wilde

Thrift-protobuf-compare Comparing various aspects of Serialization libraries on the JVM platform (http://code.google.com/p/thrift-protobuf-compare/)

Using JAX-RS with Protocol Buffers for high-performance REST APIs (http://www.javarants.com/)