Spring –RESTful Webservices

Authored and Presented by : Asfiya Khan

Document History

Version No.	Authored/ Modified by	Remarks/ Change History	Date <dd- mon-yy >
1	Asfiya Khan	Modified by adding REST with Spring	17/3/2017

Learning Objectives

RESTFull Webservices for the first time.

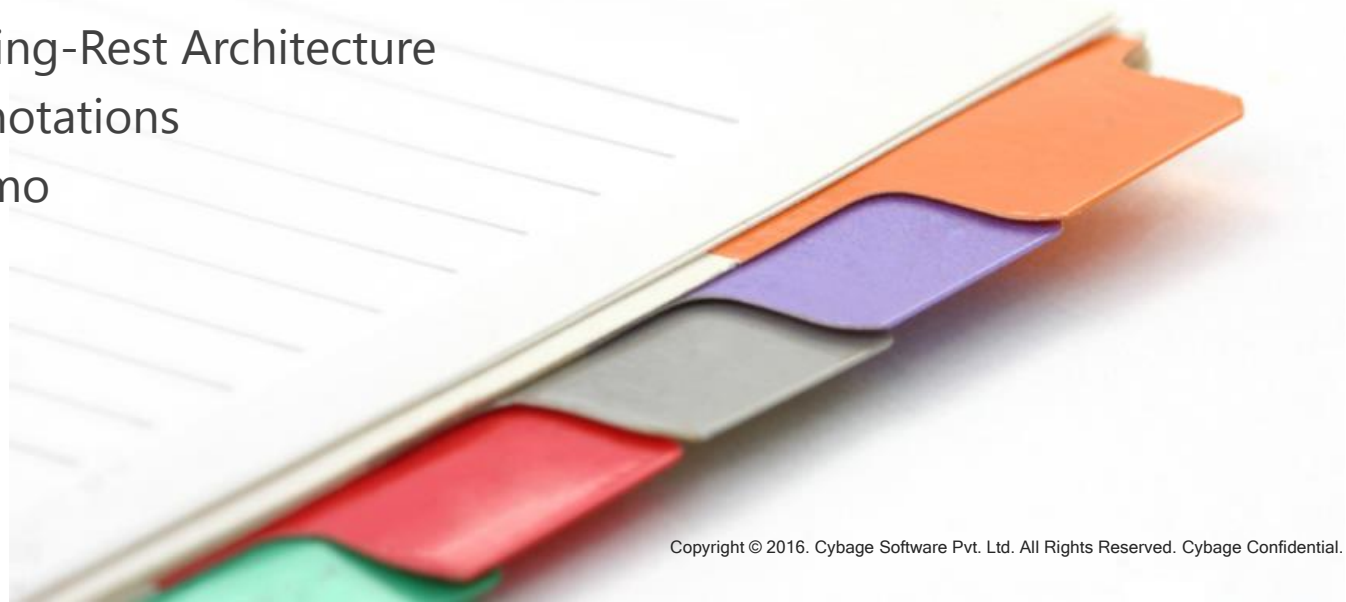
Course Structure

Target audience	Intermediate
Level	1,2
Pre-requisites	Should have idea of Web services
Training methods	
Evaluation	None

Cybage®
Delivering Value Scientifically

Agenda

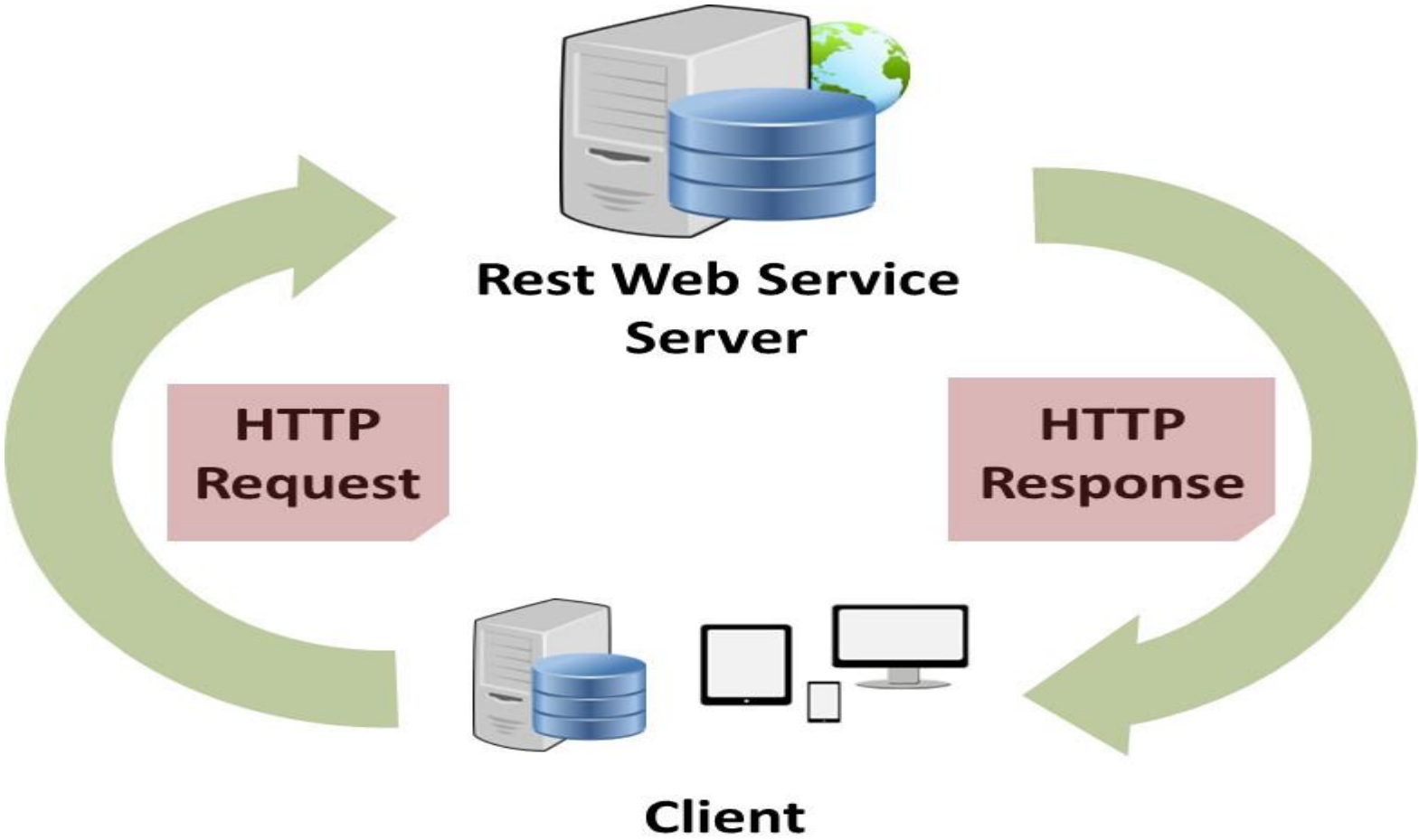
- What is WebServices?
- Introduction to REST
- Principles of REST
- REST Vs SOAP
- Why REST?
- Resource
- Resource Methods
- Spring-Rest Architecture
- Annotations
- Demo



Web Service

- **Web pages** allow **people** to communicate and collaborate with each other.
- **Web services** allow **programs** to communicate and collaborate with each other.
- A web service, is used by software developers, generally refers to an operation that is performed on a remote server and invoked using the XML/SOAP/REST specifications.

Web Service



Introduction

- Representational State Transfer (REST) is an architectural style for designing loosely coupled web services.
- First presented by Roy Fielding in 2000.
- Used to develop lightweight, fast, scalable, and easy to maintain, web services that often use HTTP.
- REST is not linked to any particular platform or technology
- The difference between a web service and a website is about who accesses it. The latter is accessed by human beings and former is accessed by programmed clients

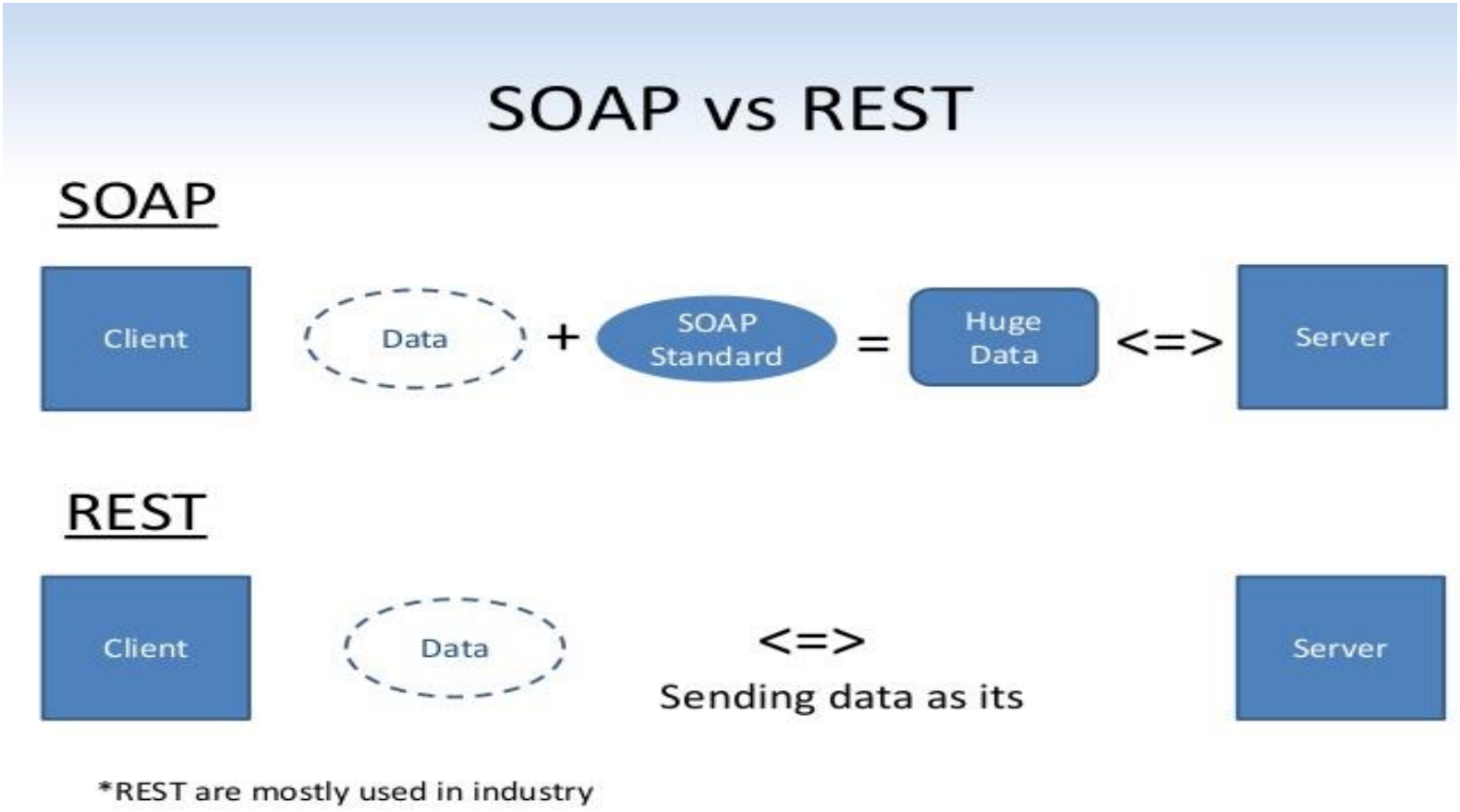
Guiding Principles of REST

1. **Client–server**
2. **Stateless**
3. **Cacheable**
4. **Uniform interface**
5. **Layered system**
6. **Code on demand (optional)**

SOAP Vs REST

#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.

SOAP Vs REST



Why REST?

1. REST can be consumed by any client e.g. Java, C++, Python client and even a web browser with Ajax and JavaScript.

2. REST is lightweight as compared to SOAP, it doesn't require CPU consuming XML parsing and it also consumes less bandwidth because unlike SOAP, REST doesn't require a SOAP header for every message.

3. SOAP is an old technology, all modern technical giant are using REST e.g. Google, Twitter, and Flickr.

4. REST is easy to learn, its just nouns and verbs. If you already know HTTP methods then its even easier.

5. Java has excellent support for RESTFul web services.

How REST work?

- *Representational* cause response is itself a representation of the **resource** present on the server.(ex: JSON, XML, PDF , DOC etc)

- REST is stateless as uses HTTP protocol.

- REST defines some 'verbs' in order to interact with the resources. Some of these are

  GET: to receive the resource representation

  POST: to add some information to the resource

  PUT: modify the resources

  DELETE: delete the resources

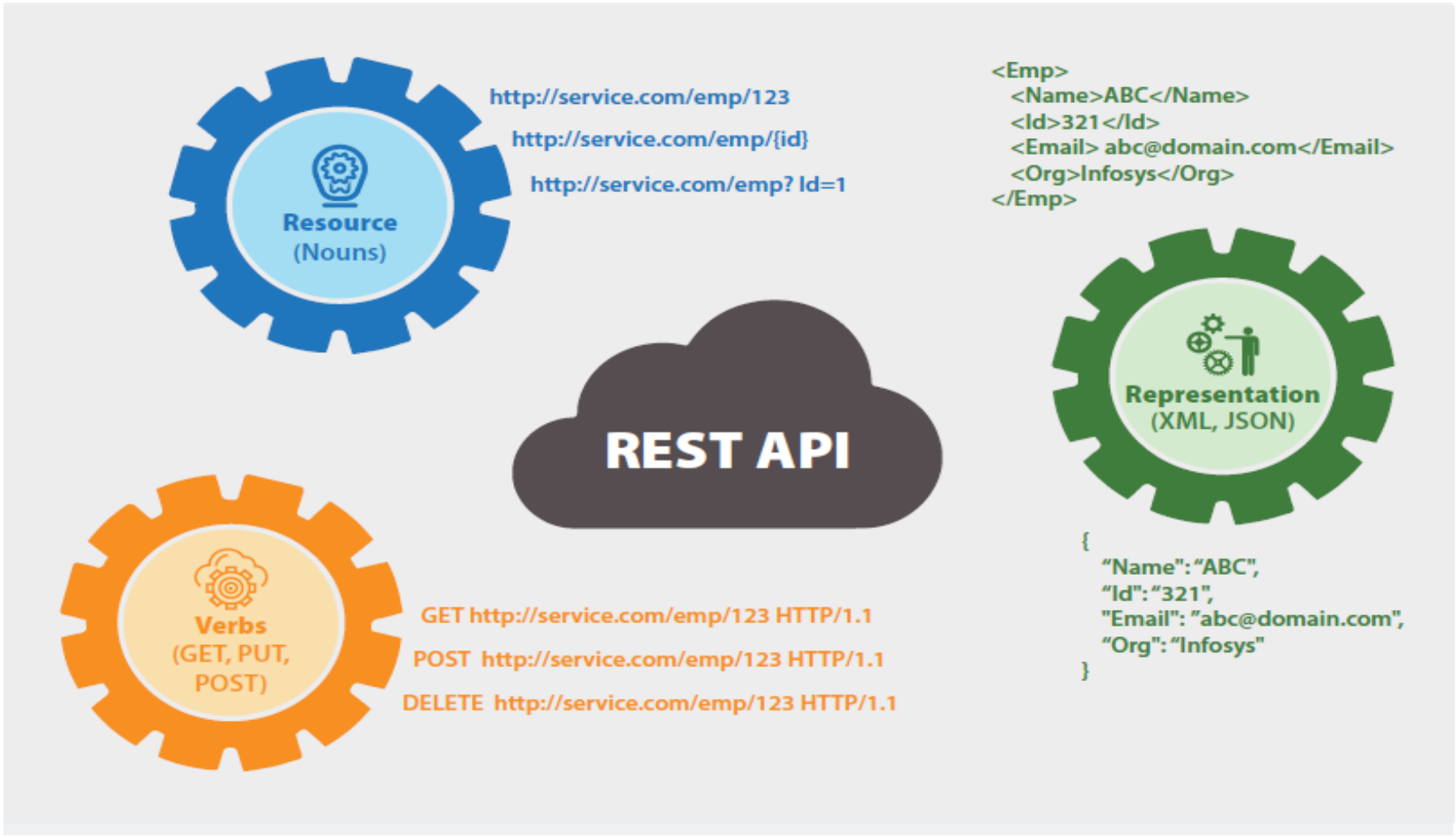
Resource

- The key abstraction of information in REST is a **resource.**
- Any information that can be named can be a resource: a document or image, a temporal service, a collection of other resources, a non-virtual object (e.g. a person).
- REST uses a **resource identifier** to identify the particular resource involved in an interaction between components.
- The data format of a representation is known as a **media type.**

Resource Methods

HTTP METHOD	CRUD	ENTIRE COLLECTION (E.G. /USERS)	SPECIFIC ITEM (E.G. /USERS/123)
POST	Create	201 (Created), 'Location' header with link to /users/{id} containing new ID.	Avoid using POST on single resource
GET	Read	200 (OK), list of users. Use pagination, sorting and filtering to navigate big lists.	200 (OK), single user. 404 (Not Found), if ID not found or invalid.
PUT	Update/Replace	404 (Not Found), unless you want to update every resource in the entire collection of resource.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
PATCH	Partial Update/Modify	404 (Not Found), unless you want to modify the collection itself.	200 (OK) or 204 (No Content). Use 404 (Not Found), if ID not found or invalid.
DELETE	Delete	404 (Not Found), unless you want to delete the whole collection — use with caution.	200 (OK). 404 (Not Found), if ID not found or invalid.

REST API



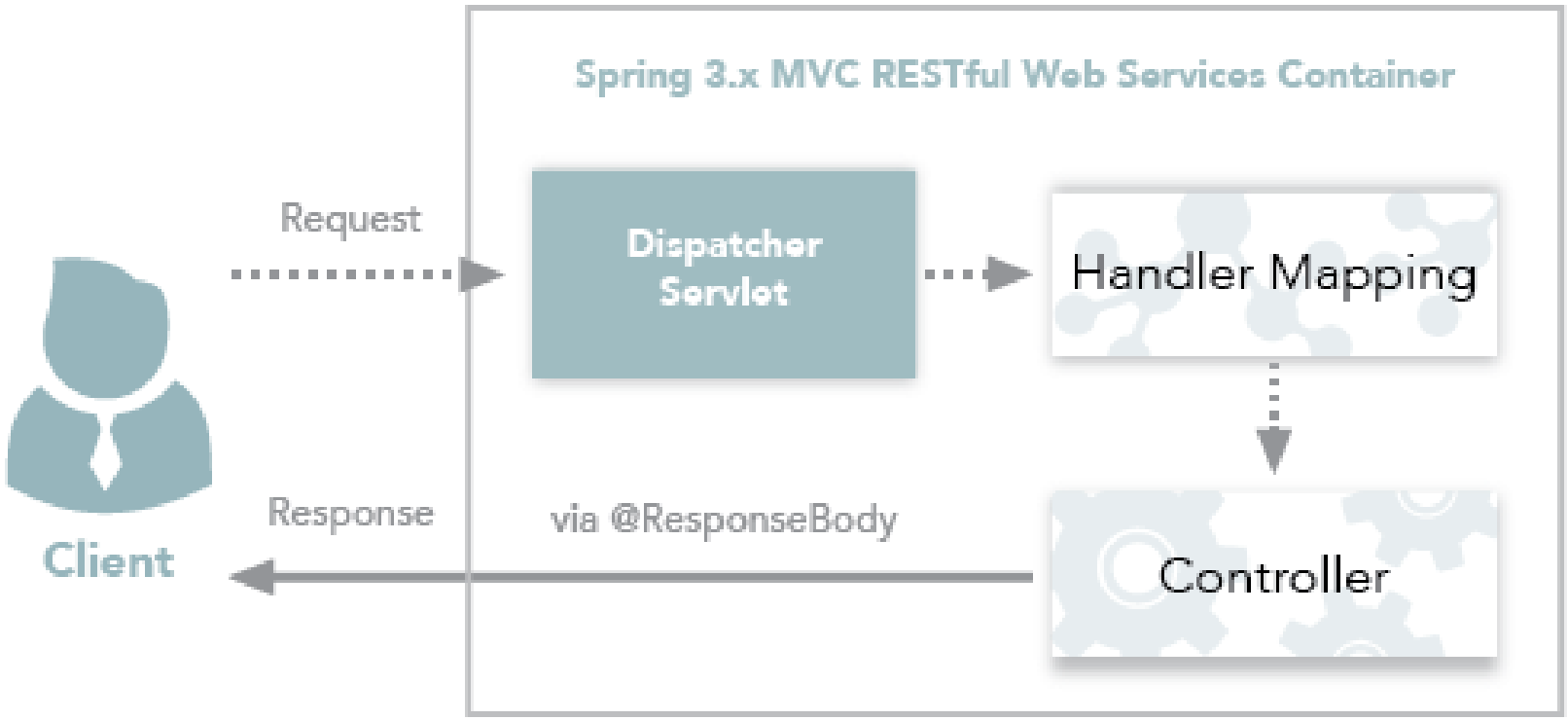
REST and HTTP are not same !!

- REST and HTTP are not same !!
- A lot of people prefer to compare HTTP with REST. REST and HTTP are not same.

REST != HTTP

Spring with REST

Spring-REST Architecture



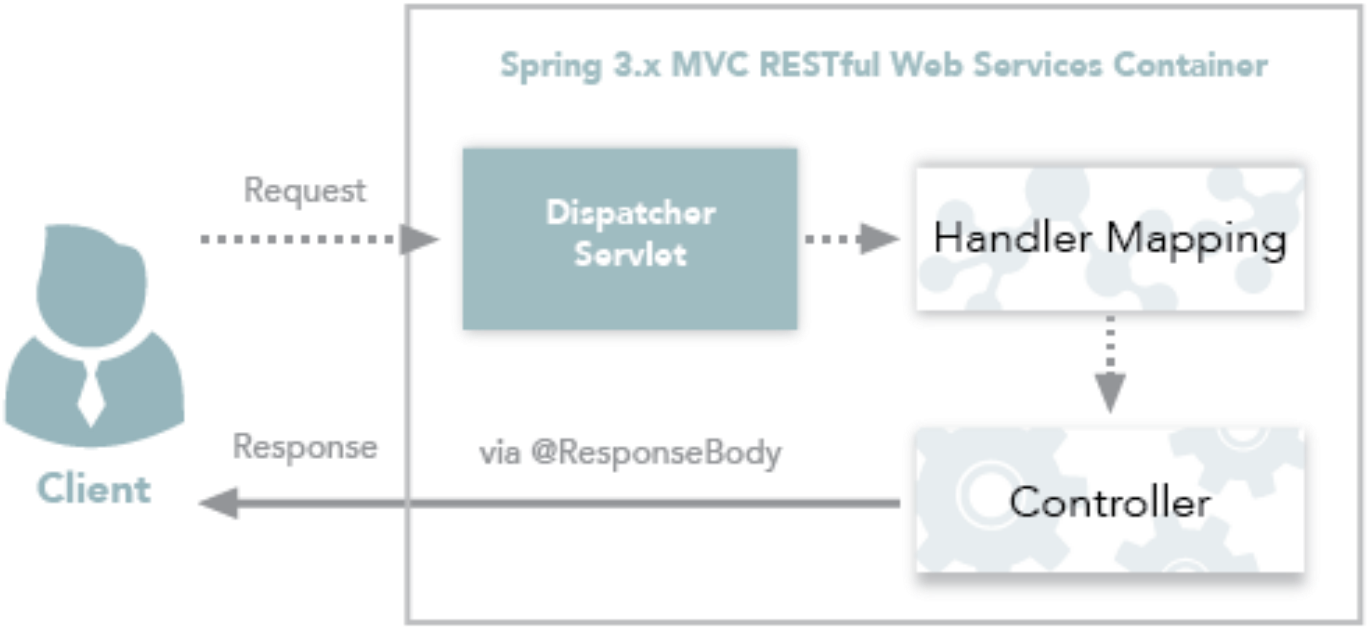
Spring MVC Controller & RestController

- The key difference between Spring MVC controller and the RESTful web service controller is the way the HTTP response body is created.

- Traditional MVC controller relies on the View technology, the RESTful web service controller simply returns the object and the object data is written directly to the HTTP response as JSON/XML.

@ResponseBody

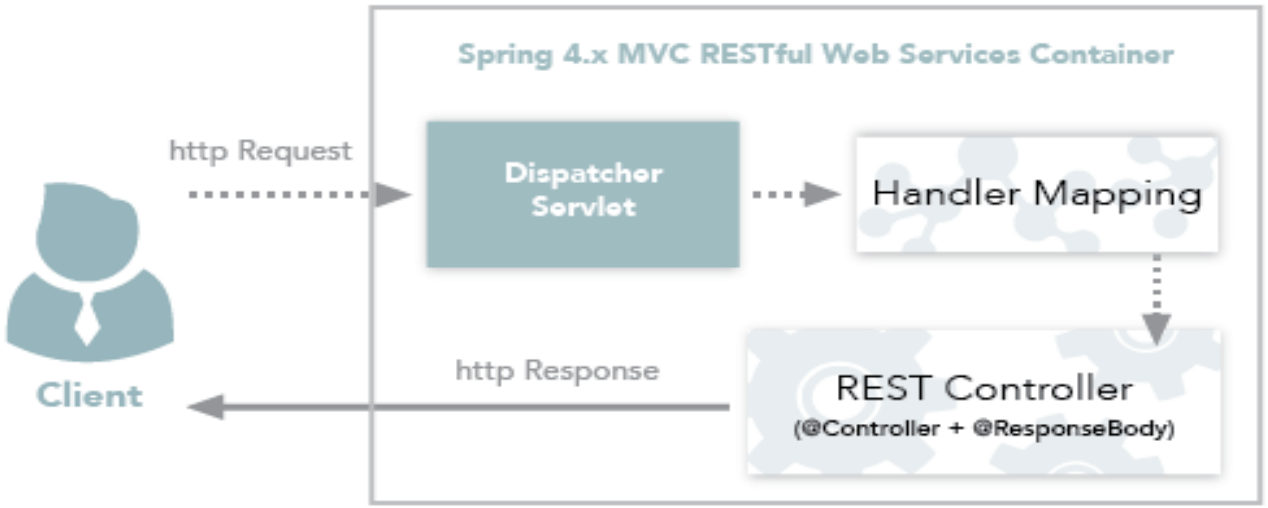
When you use the @ResponseBody annotation on a method, Spring converts the return value and writes it to the http response automatically. Each method in the Controller class must be annotated with @ResponseBody.



Cybage®
Delivering Value Scientifically

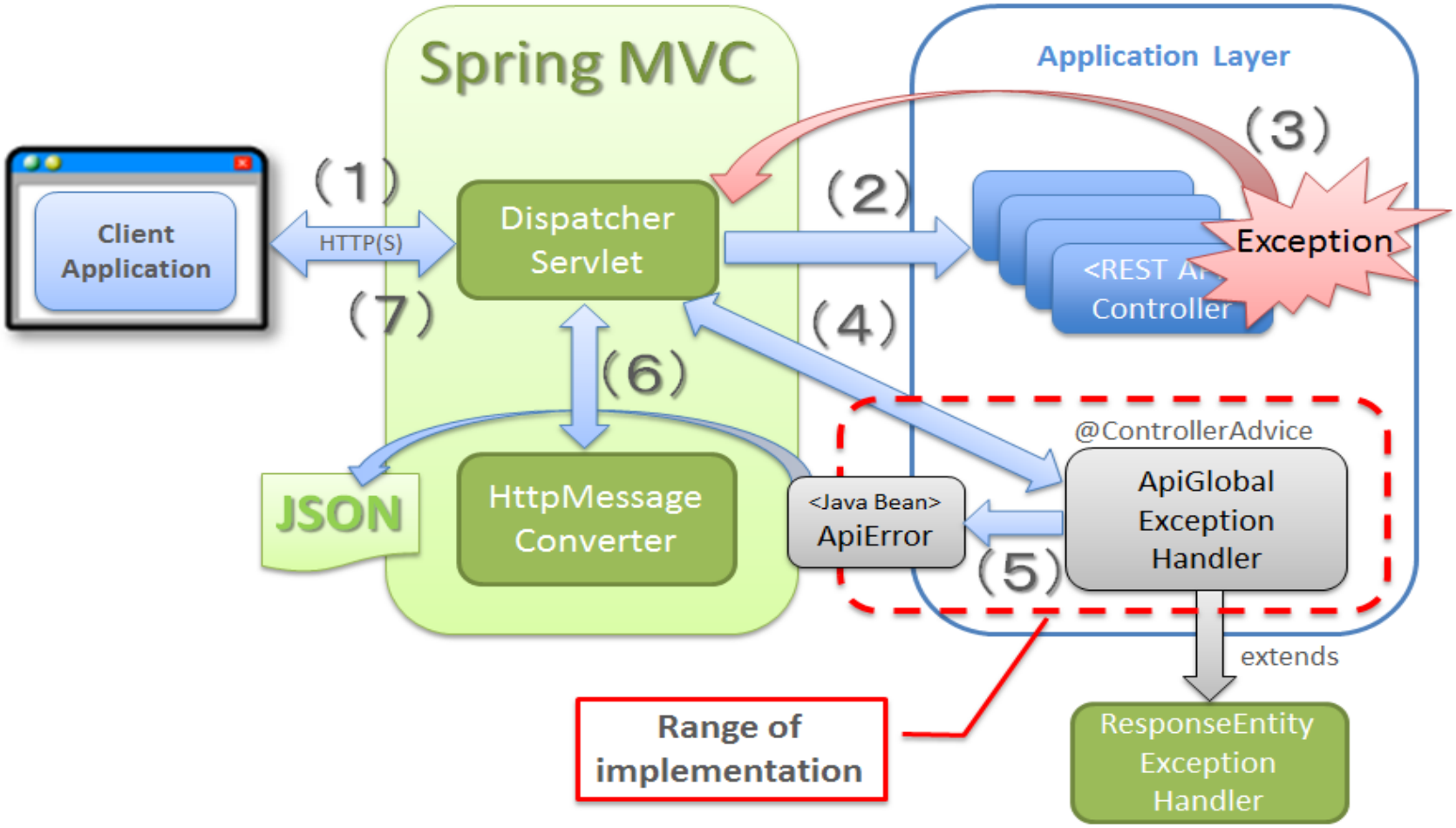
@RestController

- By annotating the controller class with @RestController annotation, you no longer need to add @ResponseBody to all the request mapping methods. The @ResponseBody annotation is active by default.



DEMO

Overall Working Stepwise



Any Questions?

Thank you!