infogain

REST Web Services

Part1

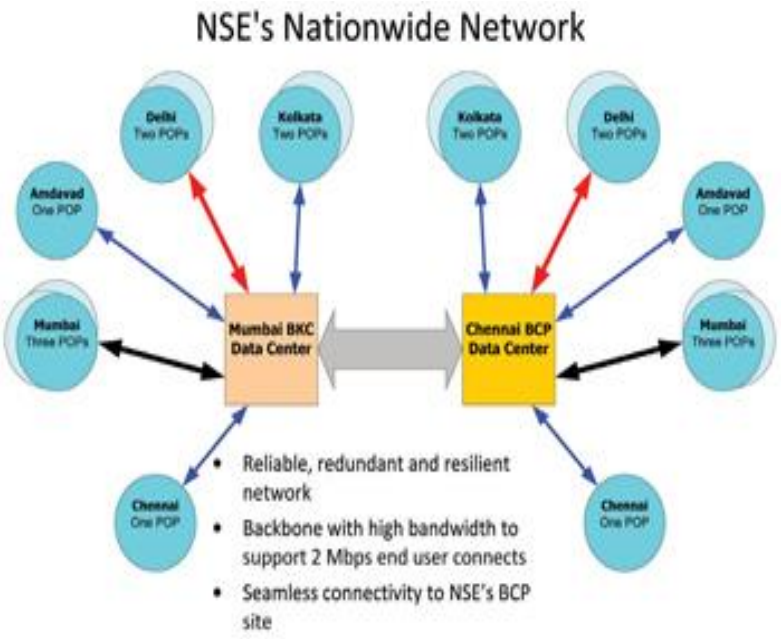
Objective

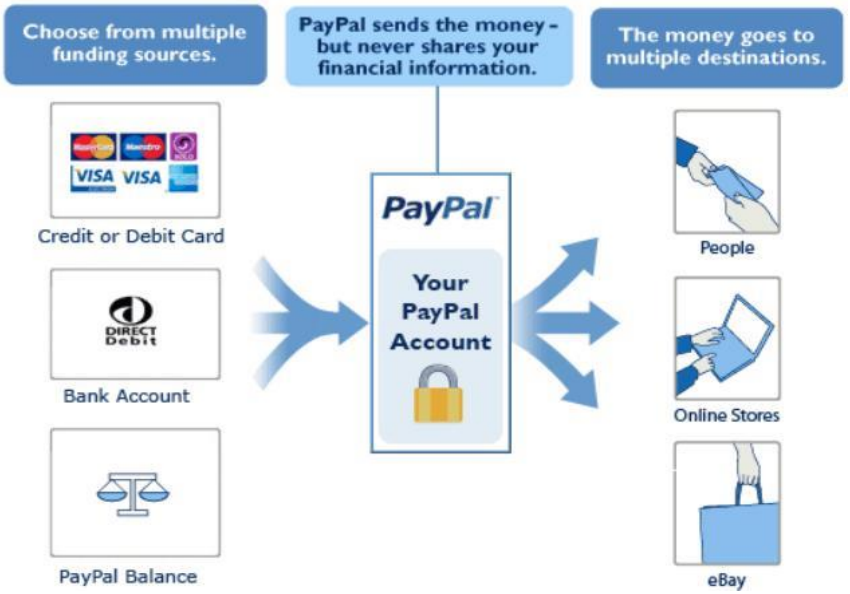
- Define Web Services.
- Explore types of Web Services.
- REST Overview
- Explore Spring Rest Services
- Http methods
- JSON/XML/JAXB
- CRUD functions

What are Web Services

- **Definition :-**
  - A Web service is a language independent B2B Component which facilitate interoperability .
  - It is a method of communication between two devices over network
  - Web service supports direct interaction with other software applications using XML based messages via internet based protocols.

Web Service Real Life Examples



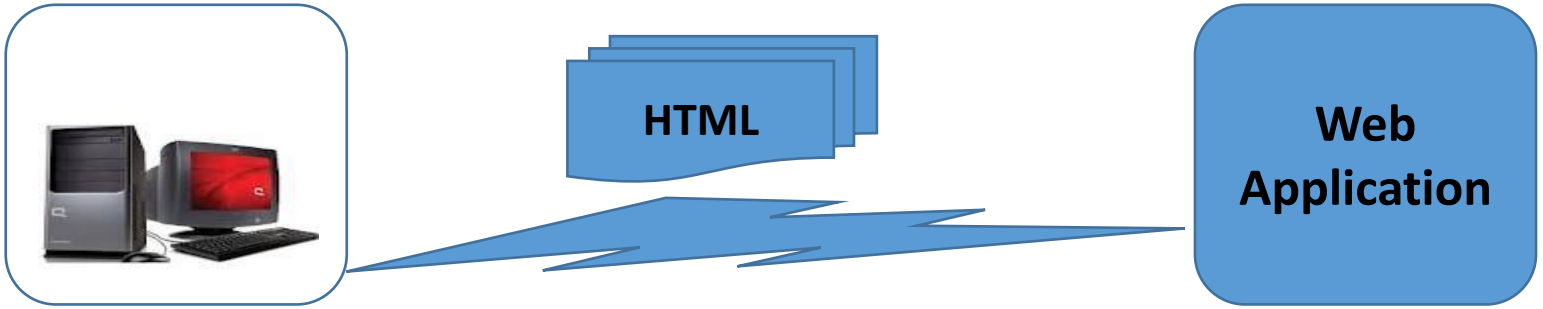


Web Services

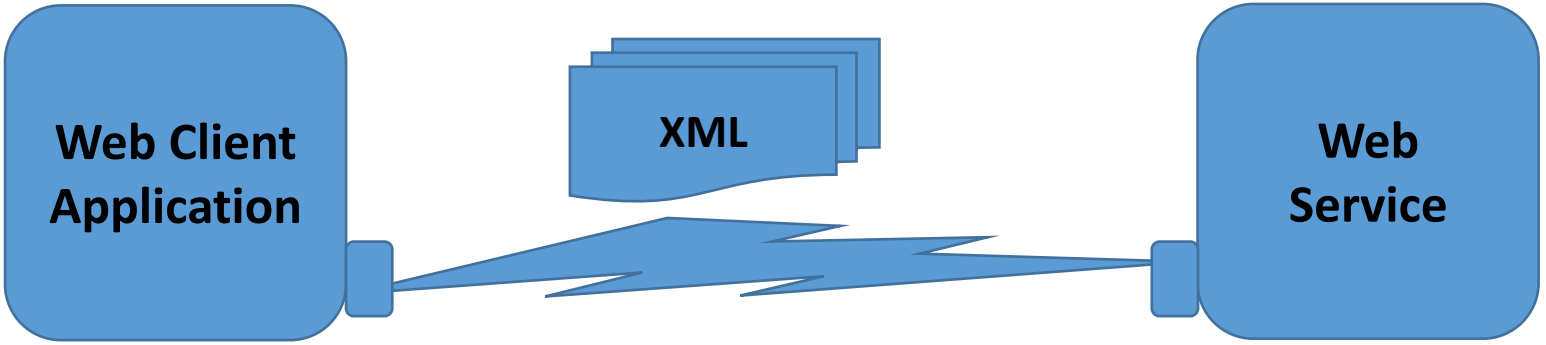
- Before web services interoperability was facilitated by CORBA .
- CORBA is a proprietary technology which provides a technology independent representation of data, a protocol to invoke methods over the network and supported implementations.
- **Limitations Of CORBA**
  - Expensive due to Limited vendor support
  - Third party drivers need to used at each end of the interoperable applications
  - Special Firewall configurations required on the machines of the network to allow CORBA packets.

Understanding Web Service

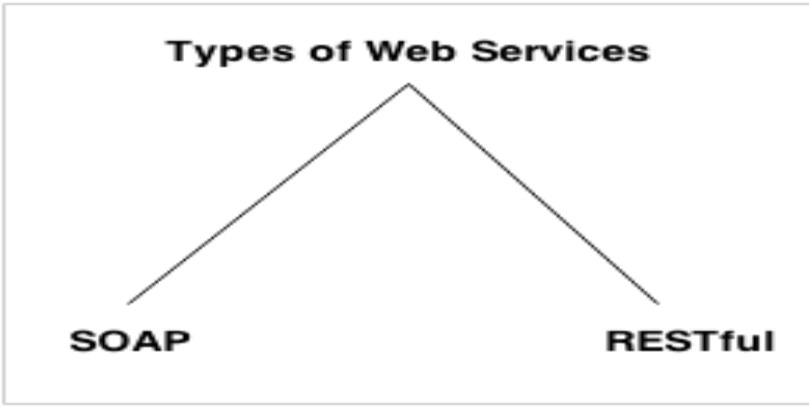
Tradition Way



Web Service Way



Types of Web Services



- **SOAP (JAX-WS) :**
  - Stands for Simple Object Access Protocol
  - It is a XML-based protocol for accessing web services.
  - It is W3C recommendation for communication between two applications.

RESTful web services

- REST stands for Representational state transfer. It is an architectural style not a protocol.
- Developed by Roy Fielding's which is based on the HTTP methods.
- It is based on HTTP protocol and its methods like PUT, GET, POST, and DELETE.
- Restful web services define the base URI (Universal Resource Identifier) for the web service.
- It also defines the end points of the service via links on the web.
- The name of the methods which are used by the service to perform these operations do not matter.

RESTful web services(Contd..)

- **Basic concepts of RESTful services are:**
  - A service is a collection of resources.
  - Each resource has a unique URI associated to it i.e. each resource is addressable.
  - Each resource may have different representations like text, html, xml ,json etc.
  - Different clients may request different representations of the requested resource.

- **Only four basic operations can be performed on the resources**
  - They can be created
  - They can be fetched
  - They can be updated
  - They can be removed

RESTful web services(Contd..)

- **In restful web services , these CRUD operations are mapped to HTTP Methods**
  - Create=POST
  - Read=GET
  - Update=PUT
  - Delete=DELETE

- Each service method is assigned a http URL on the basis of the above convention. Whenever a request is received, mapped service method is invoked.

Web Services(contd..)

- **Spring Rest web Services**
  - Spring supports annotation based MVC framework for creating RESTful web services.
  - The key difference between a traditional Spring MVC controller and the RESTful web service controller is the way the HTTP response body is created.
  - In 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.

Web Services(Contd..)

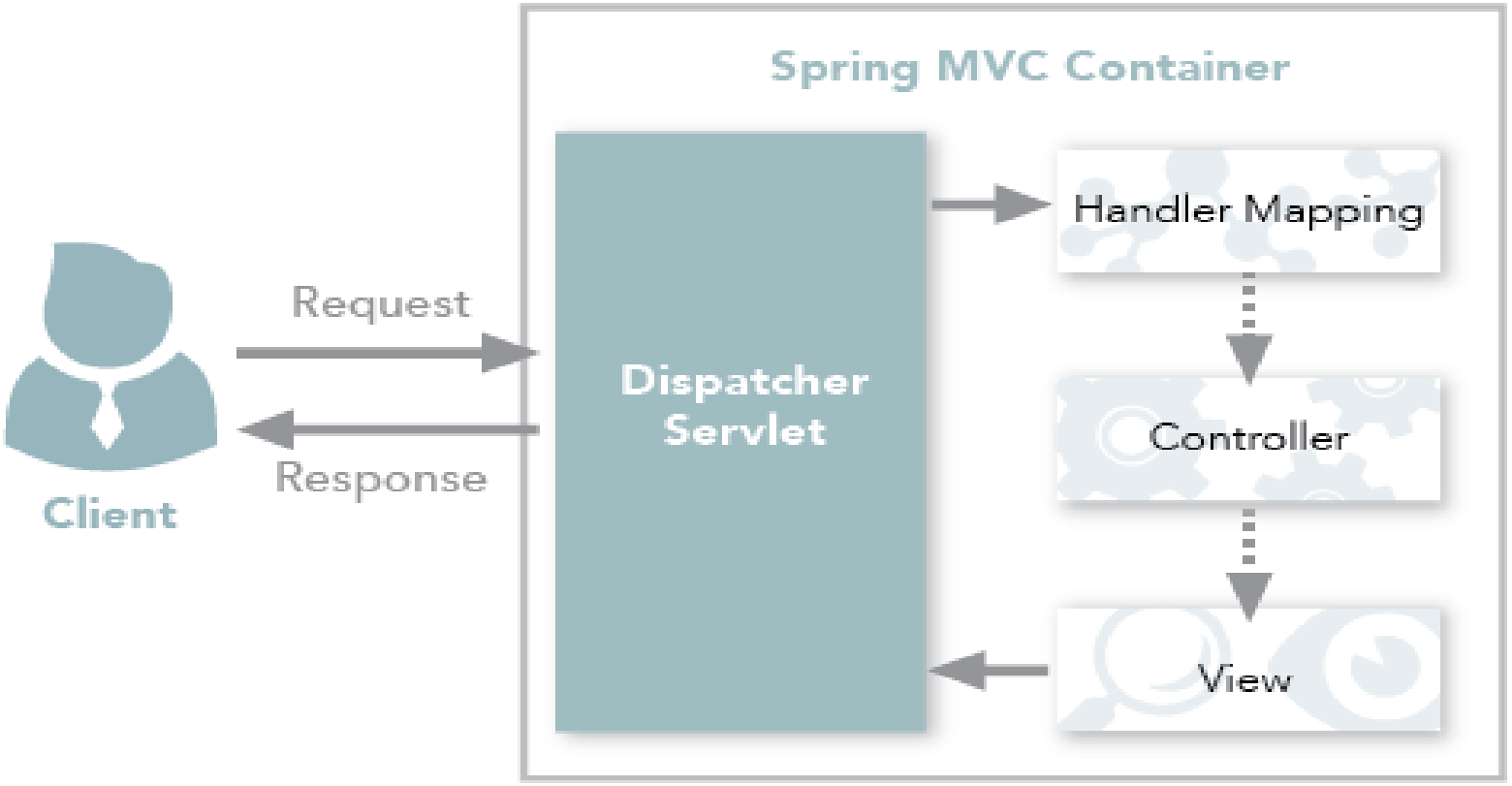


Figure 1: Spring MVC traditional workflow

Web Services(Contd.)

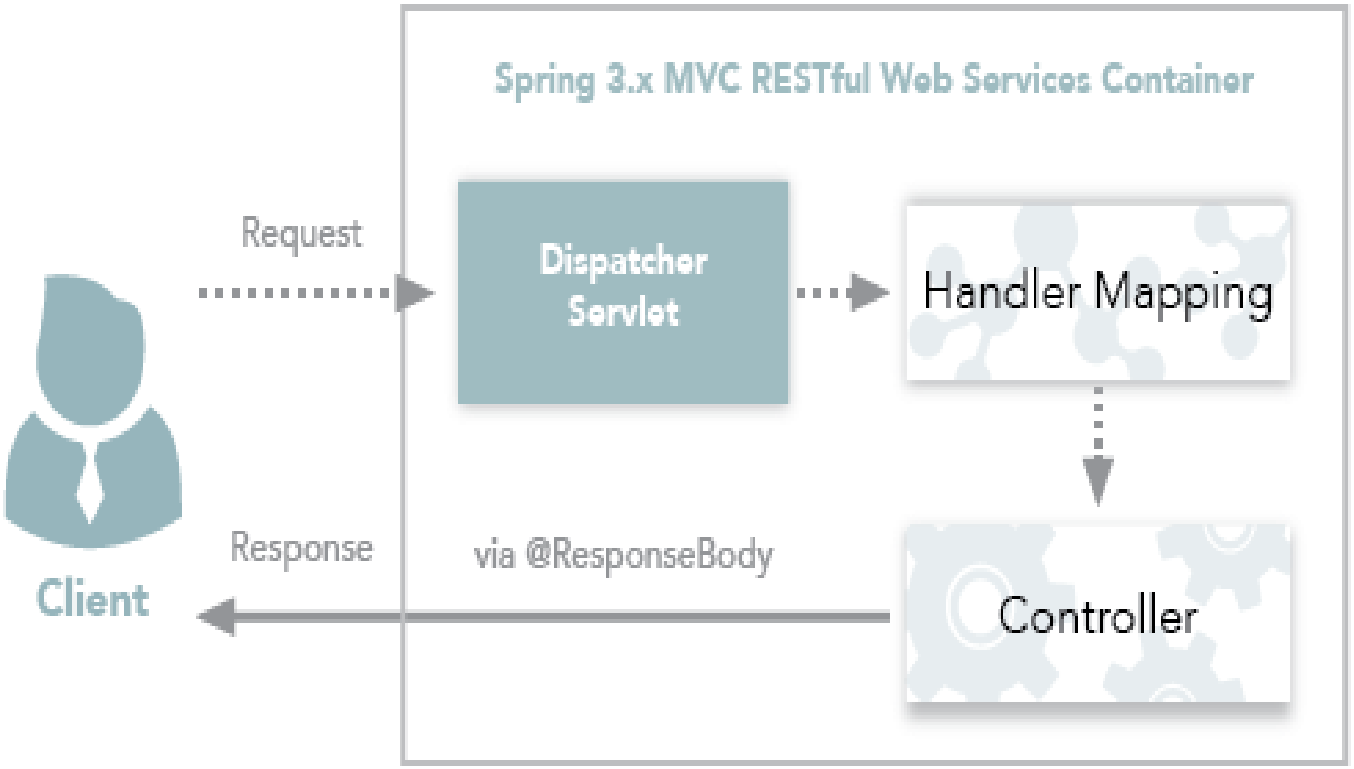


Figure 2: Spring MVC RESTful web services workflow

Web Services(Contd.)

- **@RestController**
  - Marks a class as Spring MVC Rest Controller.
  - @RestController= Controller + ResponseBody

  Example

  @RestController

  public class MyController

  { ….. }

Web Services(contd..)

- **@RequestMapping**
  - Can be used at class level as well as method level
  - Annotate the method that should handle certain HTTP request.

```
@RestController
public class  MyController {
@RequestMapping(value="/empJson", method=RequestMethod.GET,produces="application/json")
public Employee getJSON()  {
    Employee emp = new Employee();
    return emp;
        }
        }
```

Web Services(Contd..)

- **@PathVariable**
  - Indicates that a method parameter should be bound to a URI template variable [the one in '{}'].
- **Example**

  @RequestMapping(method=RequestMethod.GET, value="/emp/{id}")
  public ModelAndView getEmployee(@PathVariable String id)
  { … }
- **@RequestParam**
  - To inject a URL parameter into the method
- **@RequestHeader**
  - To inject a certain HTTP header into the method.

  Example:
  public @ResponseBody Employee getEmployeeById(@RequestParam("name")
  String name, @RequestHeader("Accept") String accept) {…}

Web Services(contd..)

- **@RequestBody**
  - If a method parameter is annotated with @RequestBody, Spring will bind the incoming HTTP request body to that parameter.
  - While doing that, Spring will [behind the scenes] use HTTP Message converters to convert the HTTP request body into domain object [deserialize request body to domain object].

```
@RestController
Public class RestController{
@RequestMapping(method = RequestMethod.POST, value = "/saveEmployee")
   public ModelAndView saveEmployeeInDB(@RequestBody String body) {
}}
```

Web Services(contd..)

- **@ResponseBody**
  - If a method is annotated with @ResponseBody, Spring will bind the return value to outgoing HTTP response body.
  - While doing that, Spring will [behind the scenes] use HTTP Message converters to convert the return value to HTTP response body [serialize the object to response body],
  - Based on Content-Type present in request HTTP header.
  - In Spring 4, you may stop using this annotation.

Web Services Response

- **Response**
  - XML
  - JSON

- **XML**
  - Often associated with SOAP
  - Still a great solution for REST
  - Easy to validate

Web Services(contd..)

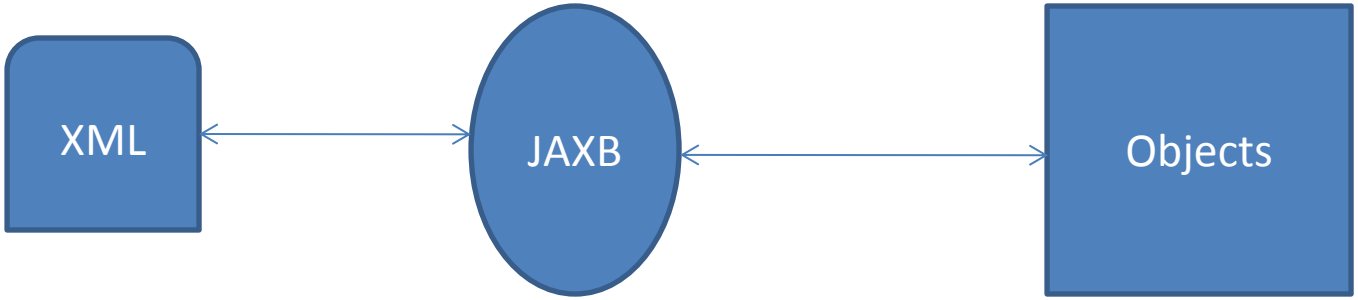
- **JSON**
  - JavaScript Object Notation
  - It's pronounced "Jason"
  - Preferred when working with JavaScript Clients
  - Flexible and easy to work with
  - Often preferred because it is not XML

- For JSON response use Jackson API in your application class Path.
- Spring automatically converts the response in JSON.

Web Services(Contd..)

- **JAXB**
  - Stands for Java Architecture for XML Binding.
  - It provides mechanism to marshal (write) java objects into XML and unmarshal (read) XML into java object.
  - Simply, you can say it is used to convert java object into xml and vice-versa.
- Since JDK1.6 JAXB is Automatically supported by java otherwise use JAXB API in your Application class Path.

Web Services (contd..)



Web Services (contd..)

Annotation	Package Detail
@XmlRootElement	javax.xml.bind.annotation.XmlRootElement;
@XmlElement	javax.xml.bind.annotation.XmlElement;
@XmlType(propOrder = { "field2", "field1",.. })	javax.xml.bind.annotation.XmlType;
@XmlType	javax.xml.bind.annotation.XmlType;
@XmlTransient	javax.xml.bind.annotation.XmlTransient;

Web Services (contd..)

- **@XmlRootElement**
  - Define the root element for the XML to be produced.
  - Annotate your class with @XmlRootElement JAXB annotation.
  - The name of the root XML element is derived from the class name

  Example
  @XmlRootElement
  public class Contact {…}
- You can also specify the name of the root element of the XML using its name attribute,
- **Example**
- **@XmlRootElement(name = "CompanyContact")**

Web Services (contd..)

- **@XmlElement**
  - Annotate all fields that needs to be included in XML/JSON output with @XMLElement.

  Example

  ```
  @XmlElement
  public String getName() {
    return name;
  }
  ```

- Either annotate all fields or all getter methods in your bean. A mix of both is not supported.
- If you want to annotate private fields instead of getter methods. Add XmlAccessorType(XmlAccessType.FIELD) at the class level

Web Services (contd..)

- **@XmlType**
  - Specify the order in which XML elements or JSON output will be produced.

```
@XmlRootElement
@XmlType(propOrder = { "id", “name",  "email"})
public class Contact {...}
```

- The above @XmlType annotation will produce the following XML.

```
<contact>
<id>38</id>
<firstname>name</firstname>
  <email>xyzl@gmail.com</email>
</contact>
```

- Similarly, it will produce the following JSON

```
{"id":"38",“name":”name","email":“xyz@gmail.com"}
```

Web Services (contd..)

- **@XmlTransient**
  - Annotate fields that you do not want to be included in XML or JSON output.

  Example

```
@XmlTransient
public Date getVersion() {
  return version;
}
```

Summary

- **In this session, we will Covered:**
- Define Web Services.
- Explore types of Web Services.
- REST Overview
- Explore Spring Rest services
- Http methods
- JSON/XML/JAXB
- CRUD functions

infogain

Thank You