# REST - Basics

Srividhya Umashanker

# We are talking about ……

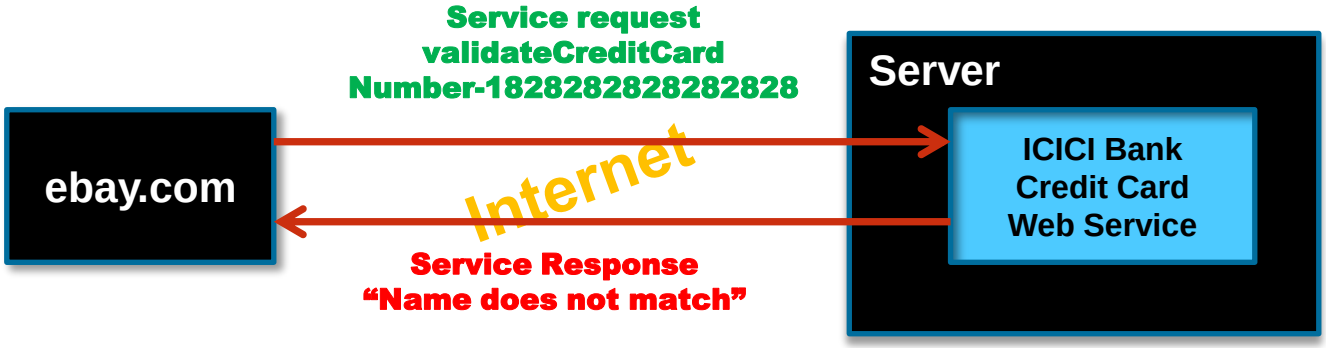

1. **Introduction**
   - Webservices, SOAP,vs REST
2. **Spring – Rest, JSON**
   - REST using Spring
   - Annotations
   - REST example with Spring

# Introduction


1. Webservices
2. SOAP
3. What is REST?
4. SOAP vs REST
5. REST in Detail
    - Frameworks,
    - HTTP Methods,
    - HTTP Headers
    - Status Code

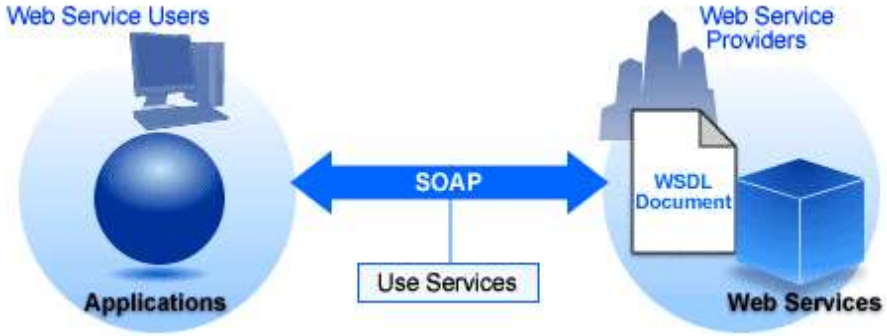
# Web Services!!

- Exposing existing software over the web (HTTP)
- Deployed independent of the OS and Programming Languages
- Standardized Protocol
- Easily connecting Different Applications i.e., Interoperability
- Low Cost of communication



# SOAP?

Simple Object Access Protocol

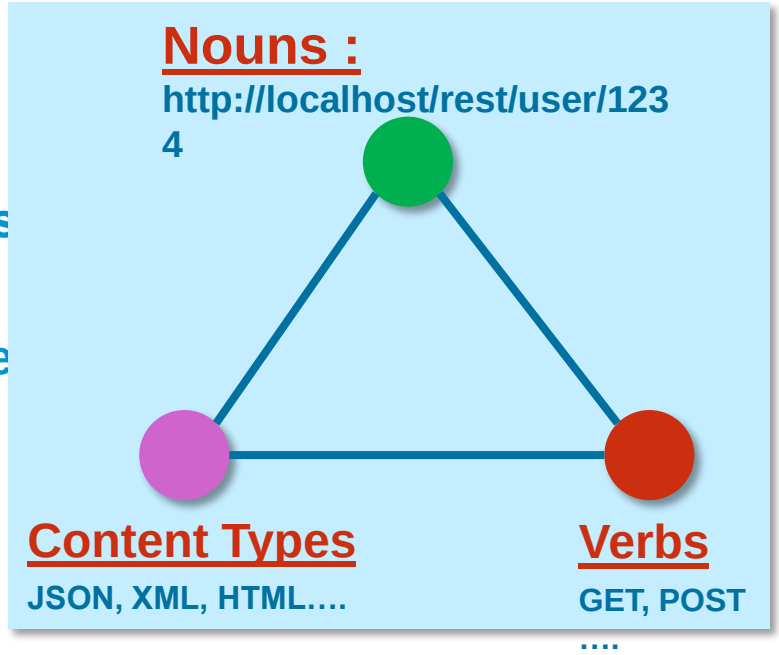



- **A simple XML-based protocol to let applications exchange information over HTTP.**
- **WSDL defines  endpoint, i/o format, security etc**

- **Example : getUser(userName);**

# What is REST? - Rest Triangle

- REST is a new and improved form of web services.
- Introduced as SOAP is complex
- There isn't a standard for REST architectures
- Expose directory structure-like URIs
- Used to expose a public API over the interne to handle CRUD operations on data.
- Works on Nouns, Verbs, Content Types




GET /server/1234

POST /server?name=SERVER-1&id=1234

# SOA vs REST – The  right webservice


## REST

- Exposes ***RESOURCES*** which represent ***DATA***
- Uses HTTP Verbs (GET/POST/DELETE)
- Emphasis on simple point-to-point communication over HTTP
- Supports multiple data formats
- Emphasizes stateless communication

**GET /user/Robert**
**GET /adduser?name=Robert**

## SOAP

- Exposes ***OPERATIONS*** which represent ***LOGIC***
- Uses HTTP POST
- Emphasis on loosely coupled distributed messaging
- Supports only XML (and attachments)
- Supports stateless and stateful/conversational operations
- Supports asynchronous messaging
- Strong Typing

**getUser(userName);**

# Rest Frameworks – List

Apache CXF

Jersey

Rest Easy

Restlet

**Spring MVC**





# HTTP METHODS – CRUD in REST

| Methods | CRUD | Example |
|---|---|---|
| GET | Fetch all or any resource | GET /user/ - Fetch all<br>GET /user/1 – Fetch User 1 |
| POST | Create a Resource | POST /user?name=user1&age=20 |
| PUT | Update a Resource | PUT /user/1? name=changed-user1&age=22 |
| DELETE | Delete a Resource | DELETE /user/1 |
| HEAD | Fetch Metainfo as header | HEAD /user |
| OPTIONS | Fetch all VERBS allowed | OPTIONS /user |

# HTTP Headers - To remember

| Headers | Example |
| --- | --- |
| Auth:<session-token> | **Auth:**1ajkdsdajsd922j-w8eis0-jssjss<br>Session token by logging in to Atlas |
| Accept:<media Type> | **accept:** application/json – Default |
| Content-Type | **content-type:**application/json |
| Allow: | Which method requests are allowed by the server |

# HTTP Status Codes


| To Remember - Status Codes | |
|---|---|
| 200 – OK | Successful Return for sync call |
| 307 – Temporarily Moved | Redirection |
| 400 – Bad Request | Invalid URI, header, request param |
| 401 – Un authorized | User not authorized for operation |
| 403 – Forbidden | User not allowed to update |
| 404 – Not Found | URI Path not available |
| 405 – Method not allowed | Method not valid for the path |
| 500 – Internal Server Error | Server Errors |
| 503 – Service unavailable | Server not accessible |

# Spring 3 - REST

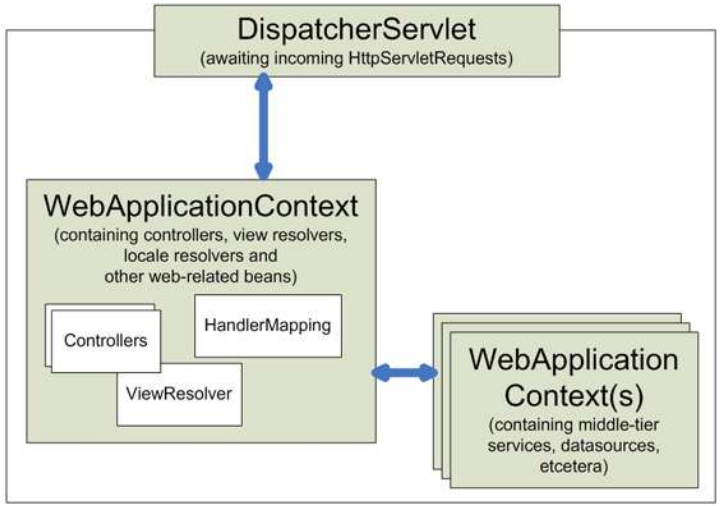

- Spring Framework - Introduction
- Dispatcher Servlet
- Flow - Rest calls in Spring
- Annotations for REST
- REST Example with Spring

# Spring Framework

- *The **Spring Framework** is an opensource, lightweight, Aspect based Inversion of Control container for the Java platform*
- helps "wire" different components together.
- Spring's REST support is based upon  Spring's annotation-based MVC framework
- configure your servlet container as you would for a Spring MVC application using Spring's DispatcherServlet.

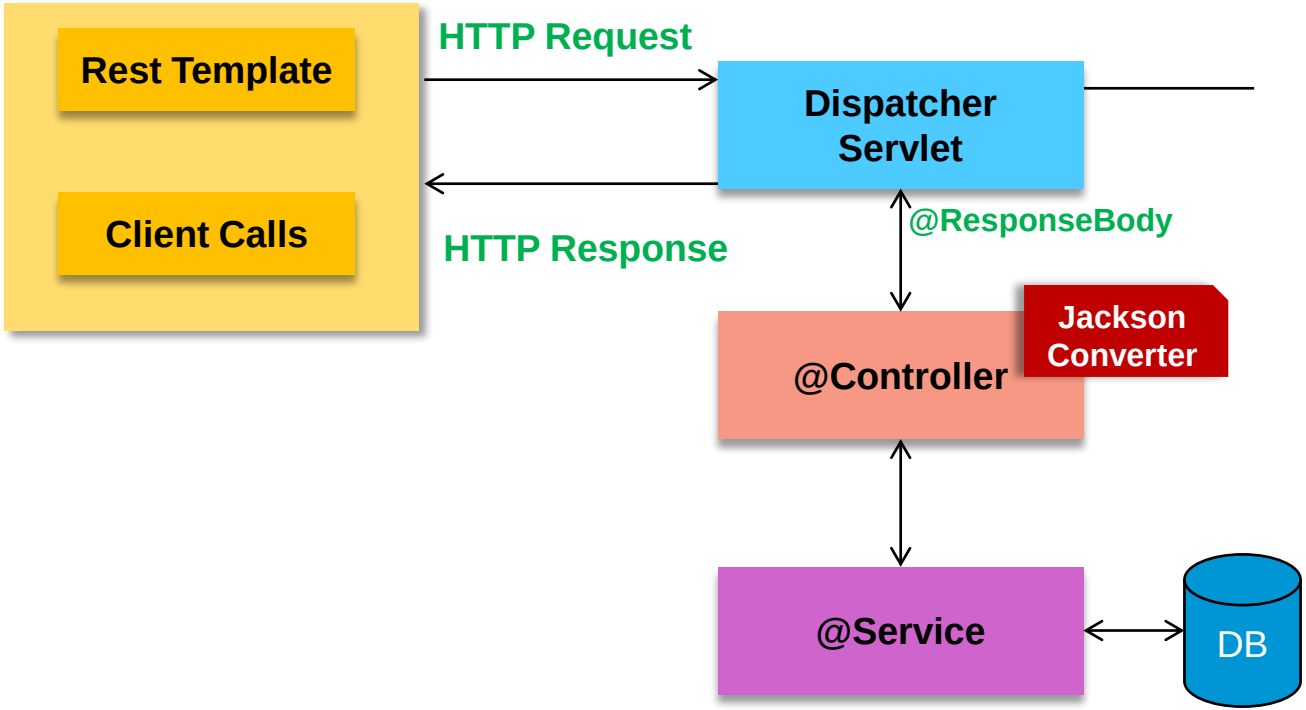
http://static.springsource.org/spring/docs/3.0.0.M3/reference/html/ch18.html

# Dispatcher Servlet




```
<web-app>
    <servlet>
        <servlet-name>rest</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>rest</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>
</web-app>
```

# Flow – REST calls in Spring



# Rest Template

Spring Framework’s RestTemplate provides simple ways to make requests to RESTful services.

Allows all REST METHODS – GET, POST, DELETE, PUT, HEAD, OPTIONS

# Spring Annotations for REST

| Annotations | Usage |
|---|---|
| @Controller | mark the class as a MVC controller |
| @RequestMapping | Maps the request with path |
| @PathVariable | Map variable from the path |
| @RequestBody | unmarshalls the HTTP response body into a Java object injected in the method. |
| @ResponseBody | marshalls return value as HTTP Response |
| @Configuration | Spring Config as a class |

## Example showing Annotations

```
@Controller
@RequestMapping(value = "/ilo")
public class iLOController
{
    @RequestMapping(value = "/server/{id}", method = RequestMethod.GET)
    public @ResponseBody Book getServer(@PathVariable String id) {
        System.out.println("------Gettting Server ------"+id);
    }
        ……
        ……..
}
```

# JSON

Javascript Object Notation

- Lightweight Data Interchange Format – No tags
- Easy to parse and create
- Supports objects and Arrays
- We use **Jackson Framework** for JSON conversions

**JSON Example:**

```
{
  “servers": [
      { “name":“server1" , “category":“Blade" },
      { “name":“Server2" , “category":“DL360" },
      { “name":“Server3" , “category":“DL480" }
  ]
}
```

**Array**

**Object**

# We are Done!