**Spring Microservices**

**Introduction to webservices**

A service which is delivered over the web.

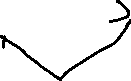
**Webservice – W3C definition**

*Software system designed to support* ***interoperable machine-to-machine interaction over a network****.*

**3 Keys:**

* Designed for machine-to-machine (or application-to-application) interaction.
* Should be interoperable – not platform dependent.
* Should allow communication over the network.

How does data exchange between applications takes place?



Application A

Webservice

Request

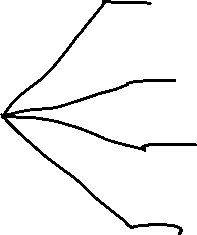
Response

If Application A needs some data or needs to perform some action on webservice, then it will send a request. Webservice then process it and generates response for the Application A.



Webservices are platform independent. So applications built in different languages should be able to make requests and response to webservice. In order for the webservice platform independent then request and response should be platform independent as well. There are two commonly used formats. XML and JSON.

How does the application know what request to be sent, where to send, what is the format of response?



Service Definition

Request/Response format

Request structure

Response structure

Endpoint

The solution to that is Service definition. Every webservice provides service definition. Service definition gives the details of how the request/response format should ex: xml, json etc, it gives info on request structure, response structure and endpoint details to where the request need to be sent.

**Key Terminology:**

* **Request**: Request is input to the webservice.
* **Response**: Response is output from the webservice.
* **Message Exchange format**
* XML and JSON
* **Service provider or server**
* **Service consumer or client**
* **Service definition** is the contract between service provider and service consumer.
* **Transport**: how the service is called. Transport indicates whether a webservice is exposed over the internet (HTTP) or over the queue (MQ).

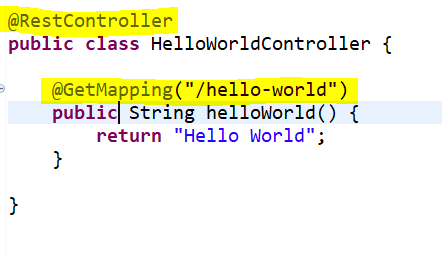
Restful webservices using Spring Boot

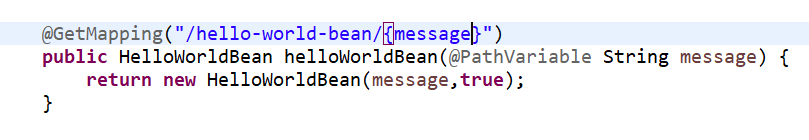
Generate a basic starter project with spring initializer (start.spring.io) and import into your eclipse ide. Rest stands for Representational state transfer.

If we want to tell a class handles rest requests, then we have annotate it with @RestController.

Then we need to create mapping. We have @RequestMapping(method=RequestMethod.GET, path=“/hello-world”)

But instead of using this we can use @GetMapping(“/hello-world”), @PostMapping() etc





We can pass something called path parameter in the uri. We have to use @PathVariable annotation.

Here we can also return some bean. And bean response will be automatically converted to JSON by Jackson data binding jar. And this is all being taken care by Spring Boot. But we have to make sure that the object has both setter and getter methods defined in it.

**What is dispatcher servlet?**

Dispatcher servlet handles all the requests and it would be the that gets the request first. This is also called front controller. It determines the right controller based on HTTP method and URI

**Who is configuring dispatcher servlet?**

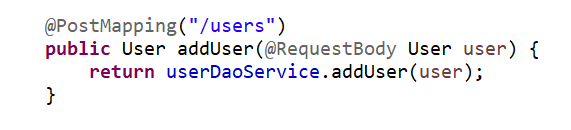
**What does dispatcher servlet do?**

**How does HelloWorldBean object converted to json?**

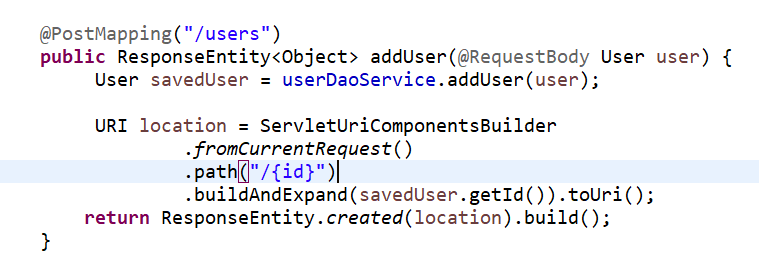
**Who is configuring the error mapping?**

Answer to all these is **Spring Boot auto configuration.**

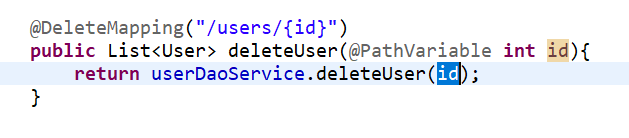
Now there are two more annotations, @PostMapping and @DeleteMapping



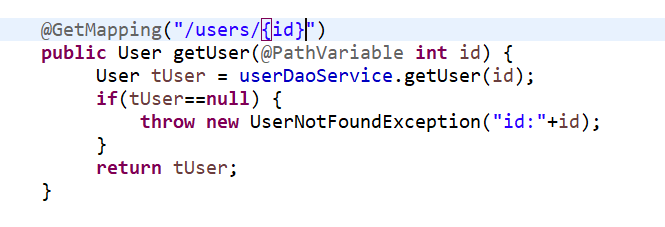
Here for @PostMapping we are using @RequestBody to get user object from the input.

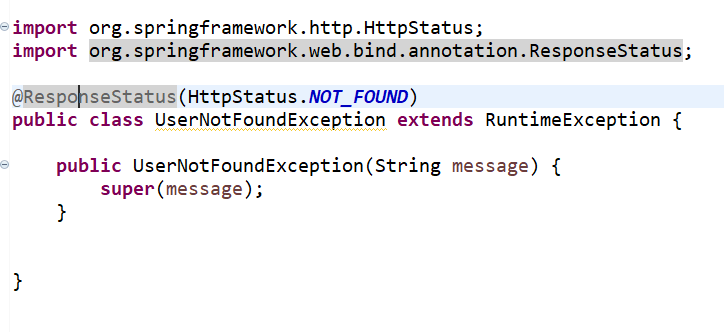


Here we have to fix the return value of addUser method, since this is a post method, we have to return ResponseEntity, so that we will send correct Response code.



**Implementing exception handling – 404 resource not found**





MICROSERVICES

Small autonomous services that work together.

Important things are

* Service that are exposed by Rest
* Small well choose deployable units
* Cloud enabled (this means if needed microservices should be scalable in both directions)

Challenges

* Bounded Context (Instead of creating a single monolith application, we create multiple microservices. But how do we define boundaries between microservices is challenging).
* Configuration Management (100s of microservices and many environments and tons of configuration to manage.)
* Dynamic scale up and down
* Visibility (the functionality is distributed among many microservices and if we want to check a bug where will find. So we want to common place to check logs)
* Fault tolerance (eg: how to prevent microservices going down because of one microservice etc.)

Spring Cloud

Spring cloud has various components that provide solutions to the challenges of microservices.

Spring Cloud provides tools for developers to quickly build some of the common patterns in distributed systems (e.g. configuration management, service discovery, circuit breakers, intelligent routing, micro-proxy, control bus, one-time tokens, global locks, leadership election, distributed sessions, cluster state). Coordination of distributed systems leads to boiler plate patterns, and using Spring Cloud developers can quickly stand up services and applications that implement those patterns. They will work well in any distributed environment, including the developer’s own laptop, bare metal data centres, and managed platforms such as Cloud Foundry.