Application A 🡪 can be a .net applicationc

Application B🡪can be a php c ------------------------------🡪 Webservice

Application C 🡪can be a Java application

Application 🡪 Request -🡪 webservice

<---Response<---

So the request and response should be Platform Independent

Popular Formats:

1. XML
2. JSON (Javascript object Notation) – popular data transfer

How does the application A know the format of webservice?

Solution is the **Service Definition** -🡪Response/Request format (json/xml /..)

🡪 Request Structure (format of the request)

🡪 Response Structure (structure of response returned by the service

-🡪 End point (end point url)

1. How can we make web services platform independent?

Like formats like MXL / JSON

2. How does the Application A know the format of Request and Response?

By providing the Service Definition

Service provider is the one which hosts the webservice

Application A is the consumer of the webservice

Transport -🡪 defines how a service is called either HTTP or MQ

REST 🡪 Restful Web Services

Representation State Transfer

HTTP Request methods(Get,PUT,POST)

REST

* Data Exchange Format
  + No Restriction. JSON is popular
* Transport
  + Only HTTP
* Service Definition
  + No Standard. WADL/Swagger/..

What is happening in the background?

Enable debug logging

1. How are our requests handled?

* Mapping servlets: dispatcherServlet urls=[/] 🡪 all the requests goes to the dispatcher servlet
* Auto Configuration(DispatchServletAutoConfiguration) Spring boot automatically configures the dispatchservlet

1. How does the HelloWorldBean object get converted to JSON
   1. @ResponseBody + JacksonHttpMessageConverters default configuration
      1. Auto Configuration(JacksonHttpMessageConvertersConfiguration) This default configuration is automatically configured for REST API for Spring Boot
2. Who is configuring the error mapping?
   1. Auto Configuration(ErrorMvcAutoConfiguration)
3. How are all jars available(Spring, Spring MVC, Jackson, Tomcat)?
   1. Starter Projects – Spring Boot Starter Web (spring-webmvc, spring-web, spring-boot-starter-tomcat, spring-boot-starter-json)

All these are autoconfigures because of two important projects of spring boot ..Starter Projects and Auto Configurations

Request methods for REST API

* GET – Retrieve details of a resource
* POST – Create a new resource
* PUT – Update an existing resource
* PATCH – Update part of a resource
* DELETE – Delete a resource

Response status for REST API

Return the correct response status

* Resource is not found => 404
* Server Exception => 500
* Validation error => 400

Important Response statuses

* 200 -success
* 201 – created
* 204 – No Content
* 401 – Unauthorized (when authorization fails)
* 404 – Resource Not Found
* 500 – Server Error

When we create POST method for create Users:

@PostMapping(“/users”)

Public ResponseEntity<User> createUser(@RequestBody User user){

User saveUser = service.save(user);

URI location = ServletUriComponentsBuilder.fromCurrentRequest()

.path(“{id}”)

.buildAndExpand(saveUser.getId())

.toUri();

Return ResponseEntity.created(location).build();

So when we make a post request to create a user. Once the user got created , then it will return a response status code of 201 and also the uri of the newly created users (