Voice Recognition

> Goal : to build a voice-controlled virtual assistant using Plivo's automatic speech recognition (ASR) feature in Java using Spring Boot

- `@RequestMapping` is used at the class level while `@GetMapping` is used to connect the methods
- The default value for `interimResults` is false

```
@RequestMapping(value = "/virtual_assistant/firstbranch/", produces = {"application/xml"}, method = RequestMethod.POST)
//@RequestMapping maps a specific request path or pattern onto a controller
//value method is an alias for path method , for example @RequestMapping("/foo") is equivalent to @RequestMapping(path="/foo")
//The params element of the @RequestMapping annotation further helps to narrow down request mapping
//You can then apply additional method-level annotations to make mappings more specific to handler methods
//it is capable of handling HTTP request methods, such as GET, PUT, POST, DELETE, and PATCH

public Response firstbranch(HttpServletRequest request, @RequestParam("Speech") final String speech, @RequestParam("From") final String fro
//@RequestMapping allows easy mapping of URL parameters with the @RequestParam annotation
//@RequestParam is used to bind a web request parameter to a method parameter
//In Spring MVC, the @RequestParam annotation is used to read the form data and bind it automatically to the parameter present in the provi
//it can extract values from the query string
throws PlivoXmlException, PlivoValidationException {
//Used within a method (or constructor)
//The throws keyword in Java is used to declare exceptions that can occur during the execution of a program.
////PlivoXmlException, PlivoValidationException : are the exception list
// ---> if we don't use throws we'll get an error saying unreported exception XXX must be caught or declared to be thrown
System.out.println("Speech Input is:" + speech);


    String hostName = request.getRequestURL().toString();
//getRequestURL -> to get the entire URL (that the client used to make the request)
//output-> The returned URL contains a protocol, server name, port number, and server path, but it does not include query string parameters
//.toString() -> A string representing the object
    final Response response = new Response(); //constant
//Response -> Defines the contract between a returned instance and the runtime when an application needs to provide meta-data to the runtim
    if (speech.equals("sales")) {
//condition "input=sales"
      response.children(new Dial().callerId(fromNumber).action(hostName + "action/").method("POST").redirect(false).children(new Number("<n
    }
//During an active call Dial() verb to connect the current caller to another party
//When you use Dial in your response to Plivo's inbound call request, the dialed party sees the inbound caller's number as the caller ID
//Action attribute accepts a URL as its argument
//Method used to send HTTP request to the action URL
//Response.Redirect(URL,false): The client is redirected to a new page and the current page on the server will keep processing ahead
//In Xml we can use the Redirect element to transfer control of a call to a different URL
//callerID : If set to a string, caller number will be set to this string value

    else if (speech.equals("support"))
//condition "input=support"
{response.children(new Dial().callerId(fromNumber).action(hostName+"action/").method("POST").redirect(false).children(new Number("<number_2
    //same here
}
   else
     {
       response.children(new Speak(wrongInput));
//Plivo will read the wrongInput message when the caller inputs a wrong digit
    }
    System.out.println(response.toXmlString());
//toXmlString -> Creates and returns an XML string that represents the object
    return response;
  }
}
```