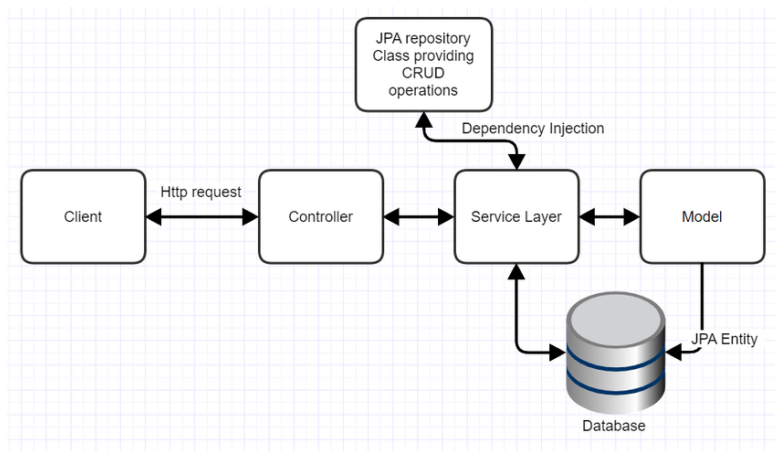


## Sporty Shoes Flow Chart (Spring Boot Project)

As we or client fire localhost: 8081/ > home.html page

Client request to >>>>>by html pages>>>> **ShoeController** with help of html pages > **ShoeController** request to **ShoeService** layer which will be extending **JpaRepository** which will be collecting data from client with help of **ShoeEntity** Class and send to again **ShoeService** layer. **ShoeService** will connect with database with help of **application.properties** and store data and to get **Data collect information case** **ShoeService** will request to **model Object** and **model object** will request to data base and collect required data in object form and return to **service layer** and **service layer** send data to **ShoeController** and **ShoeController** send data to client with help of **index.html** page



This is how Projects will look like

The screenshot shows the Spring Tool Suite IDE with the Sporty Shoes project open. The Package Explorer on the left displays the project structure, including the src/main/java directory with the following packages:

- com.example.demo
- com.example.demo.Controller
- com.example.demo.Entity
- com.example.demo.Repository
- com.example.demo.Service
- com.example.demo.static
- com.example.demo.target
- com.example.demo.target.Hibernate
- com.example.demo.target.mvnw
- com.example.demo.target.mvnw.cmd

The main editor displays the `ShoeController.java` file, which contains the following code:

```
32 model.addAttribute("shoeEntityList", shoeEntity);
33 return "index";
34 }
35
36 @GetMapping("/addshoes")
37 public String addshoesForm() {
38     return "addshoes";
39 }
40
41 @PostMapping("/register")
42 public String shoeRegister(@ModelAttribute ShoeEntity shoeEntity, HttpSession session) {
43     System.out.println(shoeEntity);
44     shoeService.addShoe(shoeEntity);
45     session.setAttribute("msg", "Shoes Added Successfully..!");
46     return "redirect:/showlist";
47 }
48
49 @GetMapping("/EditShoeDetail/{srno}")
50 public String ShoeDetailEdit(@PathVariable int srno, Model model) {
51     ShoeEntity shoeEntity = shoeService.getShoeDetailByID(srno);
52     model.addAttribute("shoeEntityDetail", shoeEntity);
53     return "EditShoeDetail";
54 }
```

The Console window at the bottom shows the application logs, including the startup of the LiveReload server, Tomcat, and the Spring Boot application. The logs indicate that the application is running successfully on port 8081.