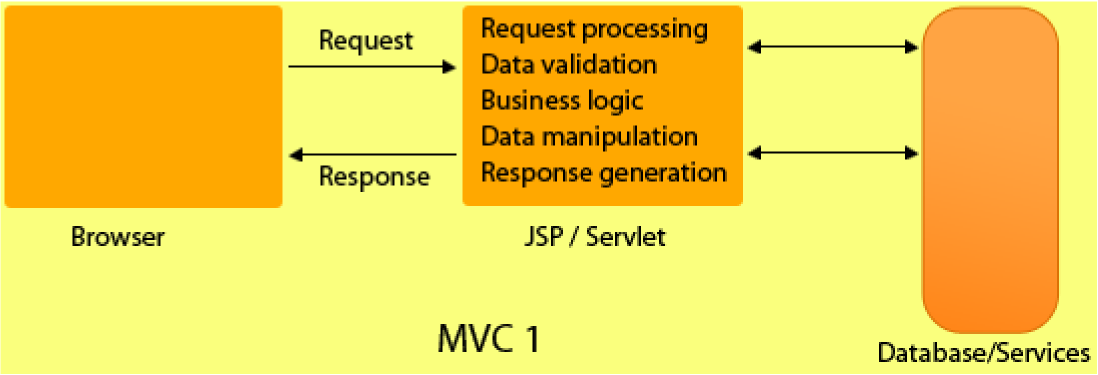
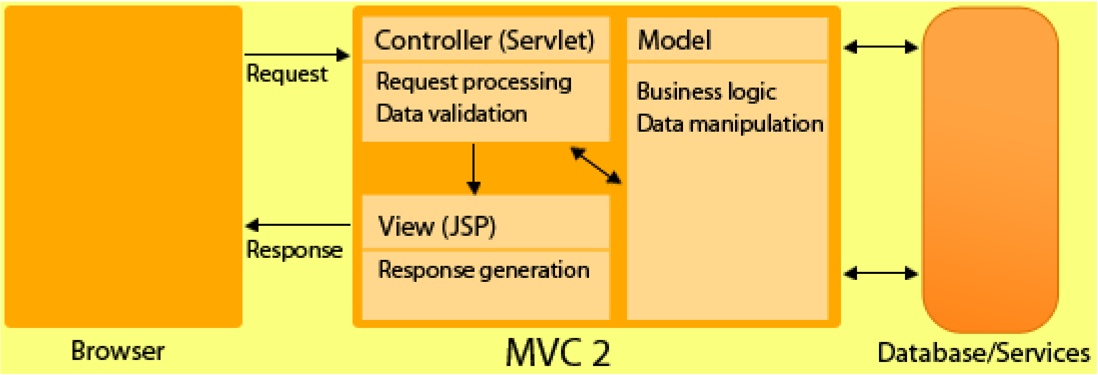
Spring MVC

Presented by

Sagar
Java Consultant

MVC Architectures

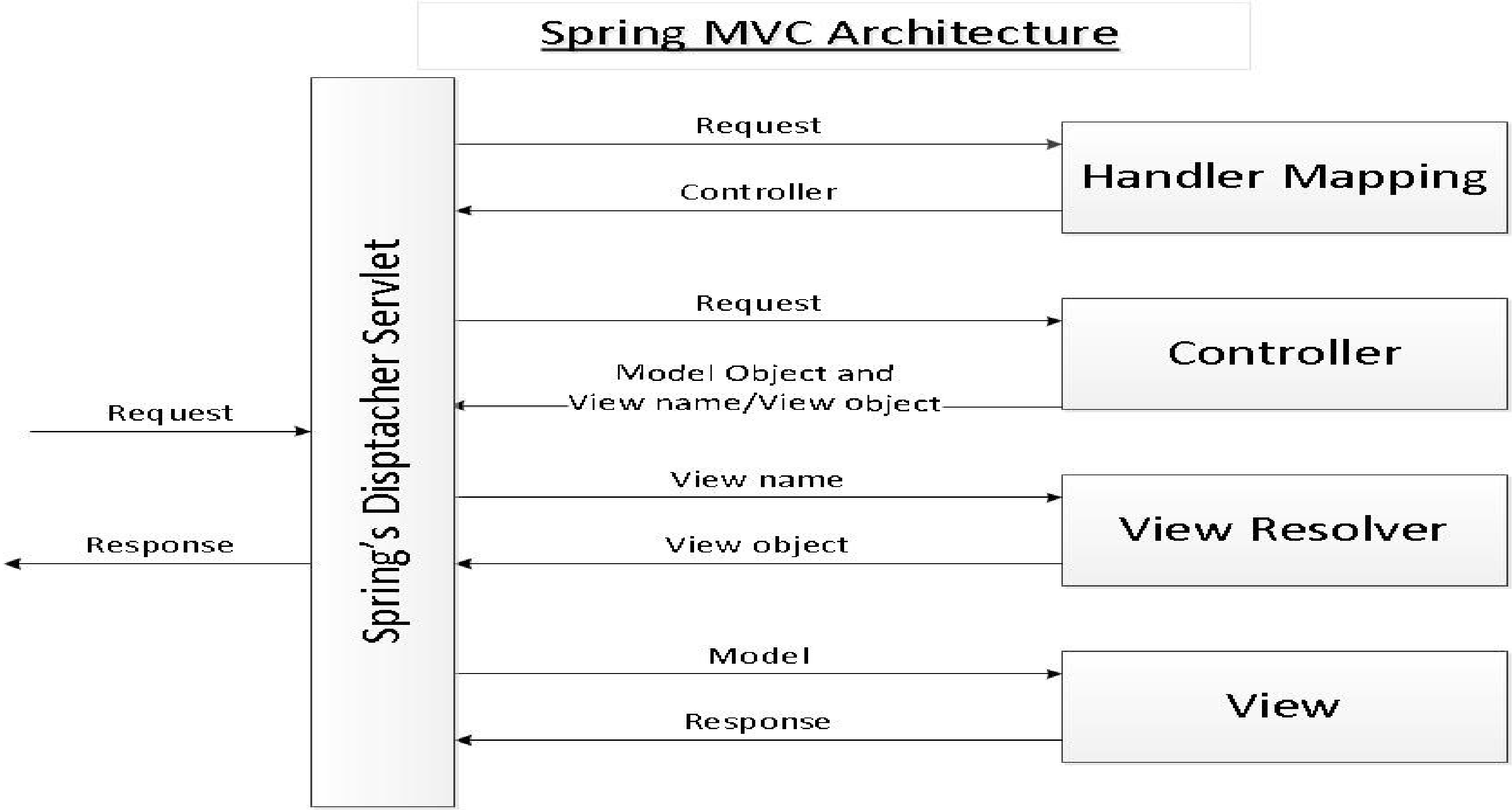




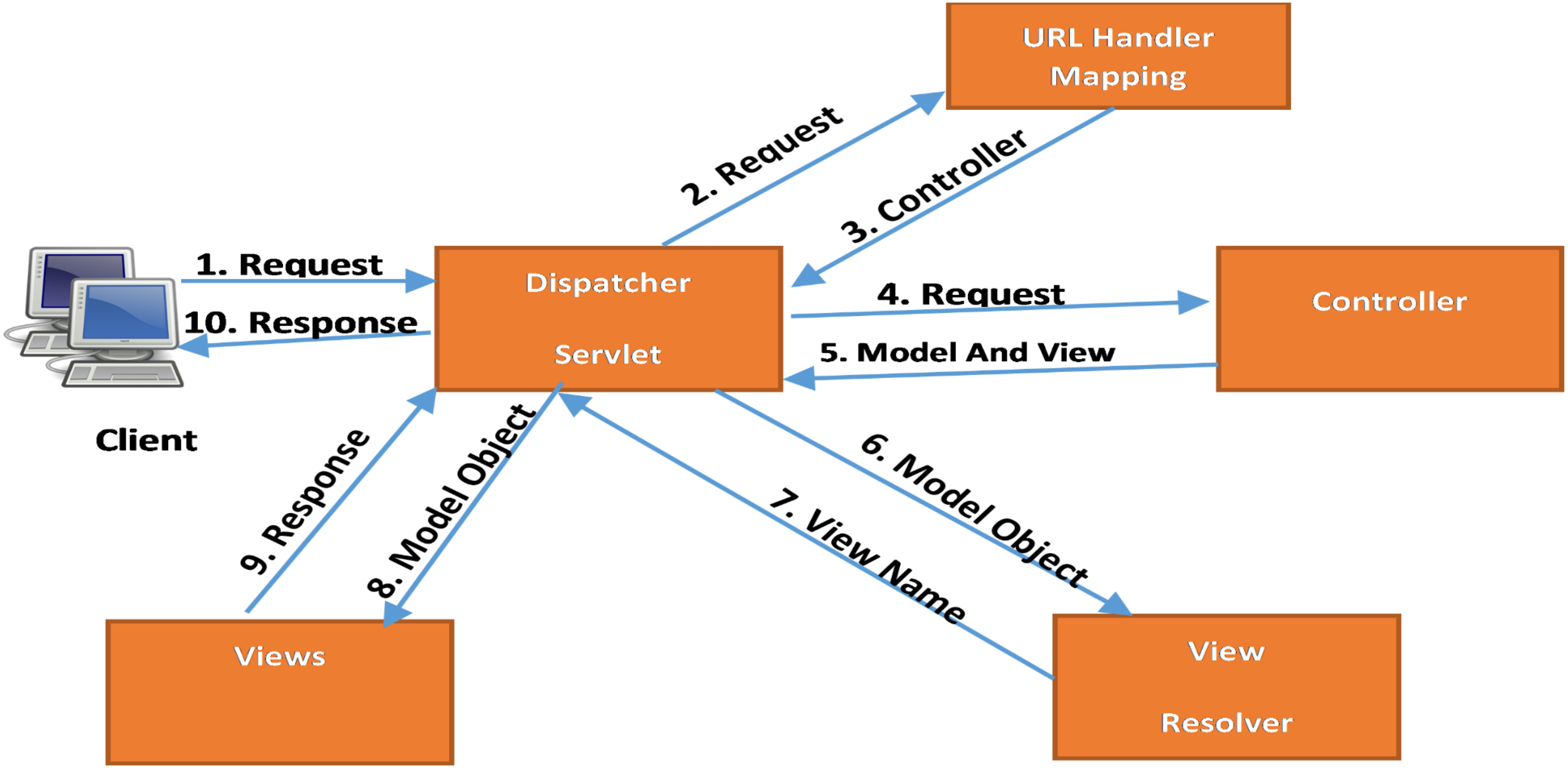
Spring MVC

- Spring provides a very clean division between
- Controllers, Java Bean Models and Views
- Spring's MVC is flexible
- Encourages 2 or more application services in a single controller, i.e., one method for one service.
- *In Servlets - one servlet for one service per request*
- Unlike Struts in Spring MVC there are no action classes. It binds directly to the domain objects.
- In Spring MVC, **DispatcherServlet** class works as the front controller which controllers other controller classes.

Spring MVC …Architecture



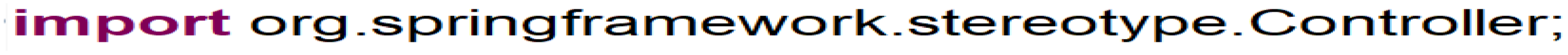
Spring MVC …Architecture



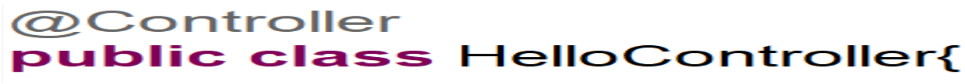
spring MVC

Spring MVC – Handler Mapping

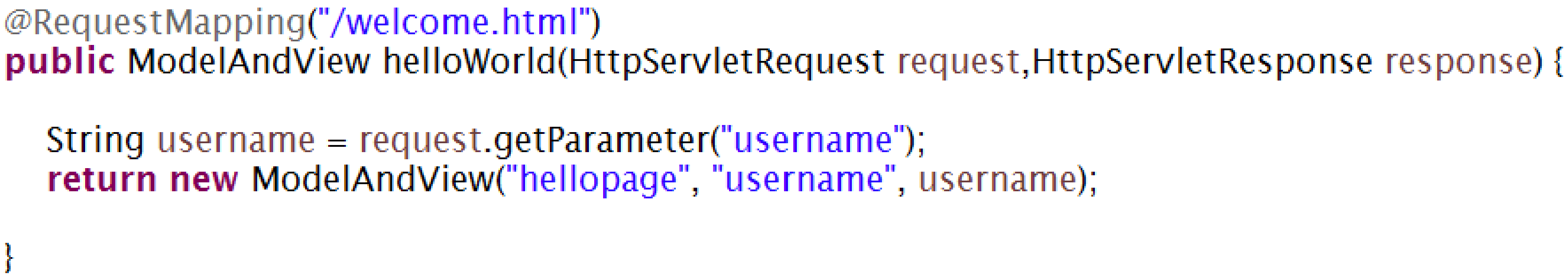
- HandlerMapping is an interface
- Its implementing classes will map the request to the corresponding method of a controller, internally.



```java
import org.springframework.stereotype.Controller;
```



```java
@Controller
public class HelloController{
```



```java
@RequestMapping("/welcome.html")
public ModelAndView helloWorld(HttpServletRequest request,HttpServletResponse response) {

    String username = request.getParameter("username");
    return new ModelAndView("hellopage", "username", username);

}
```

Spring MVC … Request Flow

1. After receiving an HTTP request, *DispatcherServlet* consults the *HandlerMapping* to call the appropriate *Controller*.

2. The *Controller* takes the request and calls the appropriate service methods based on used GET or POST method.

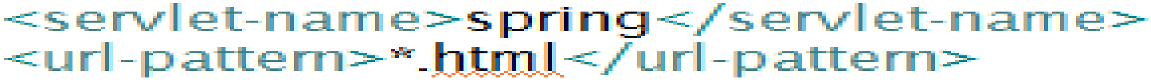
3. The service method will set model data based on defined business logic and returns view name to the *DispatcherServlet*.

4. The *DispatcherServlet* will take help from *ViewResolver* to pickup the defined view for the request.

5. Once view is finalized, The *DispatcherServlet* passes the model data to the view which is finally rendered on the browser.

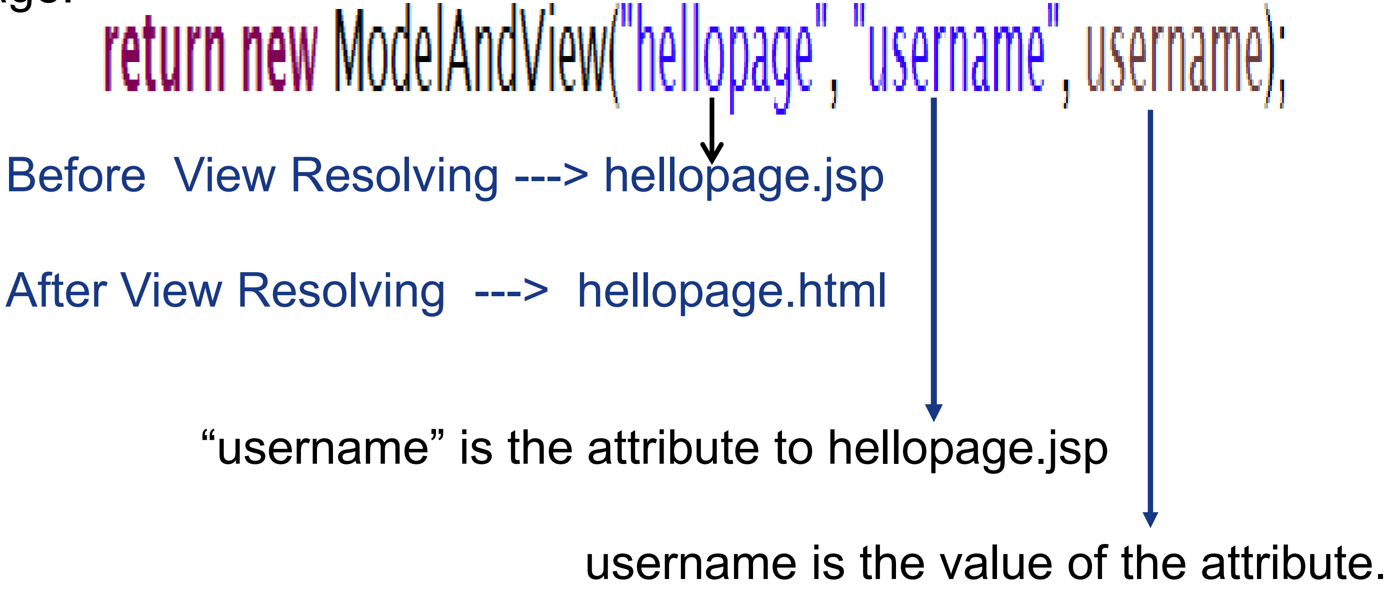
Spring MVC – View Resolver

Models are rendered in a browser through View Resolver (Dynamic to static Content). This is because of



```
<servlet-name>spring</servlet-name>
<url-pattern>*.html</url-pattern>
```

Web page will become light weight after conversion of dynamic page into static page.



```
return new ModelAndView("hellopage", "username", username);
```

Before View Resolving ---> hellopage.jsp

After View Resolving ---> hellopage.html

“username” is the attribute to hellopage.jsp

username is the value of the attribute.

**Demo

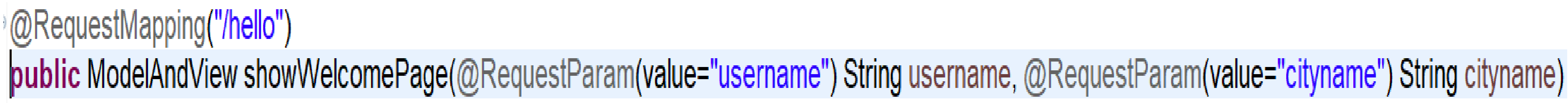
Spring MVC – @RequestParam

@RequestParam is used to retrieve the URL parameter and map it to the method argument.

In index.jsp:

```
 UserName:<input type="text" name="username"/><br/>
City:<input type="text" name="cityname"/>
```

In the controller:



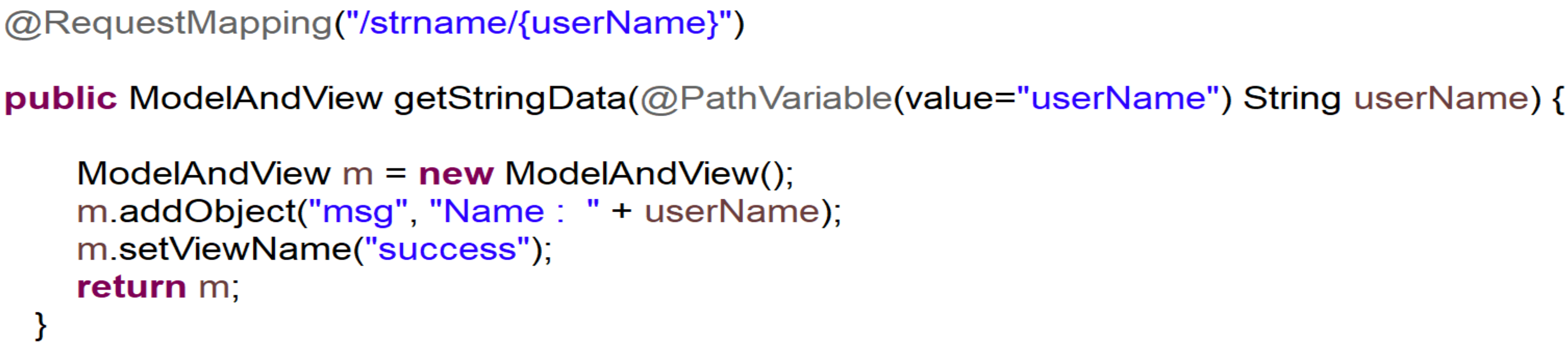
```
@RequestMapping("/hello")
public ModelAndView showWelcomePage(@RequestParam(value="username") String username, @RequestParam(value="cityname") String cityname)
```

Spring MVC – @PathVariable

@PathVariable : Used to extract the value from the URI. Spring MVC allows to use multiple @PathVariable annotations in the same method.

Ex:

<a id="str" href="http://localhost:8087/maven-springmvc-pathvariable/str/Sagar">PathVariable using String</a>



```
@RequestMapping("/strname/{userName}")

public ModelAndView getStringData(@PathVariable(value="userName") String userName) {

    ModelAndView m = new ModelAndView();
    m.addObject("msg", "Name :  " + userName);
    m.setViewName("success");
    return m;
 }
```

Spring MVC – @ModelAttribute

@ModelAttribute binds the value from jsp fields to Pojo class to perform logic in controller class.

Ex:

In index.jsp - Form parameter :

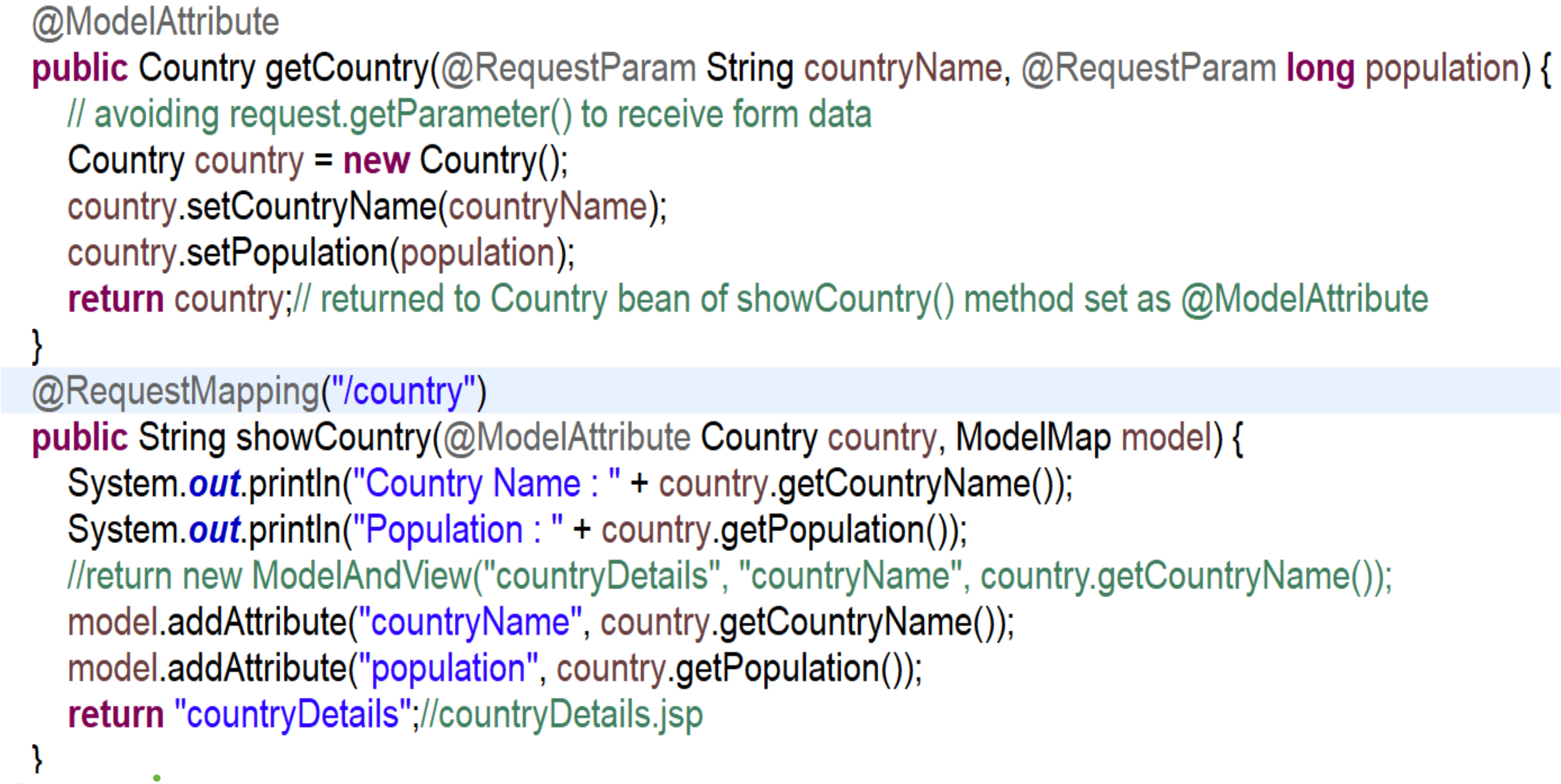
```
<input type="text" name=“countryName”/>
<input type="text" name=“population”/>
```

In the bean:

```
public class Country {
        String countryName;
        long population;
        // getters and setters
}
```

Spring MVC – @ModelAttribute

In the controller :



```
@ModelAttribute
public Country getCountry(@RequestParam String countryName, @RequestParam long population) {
   // avoiding request.getParameter() to receive form data
   Country country = new Country();
   country.setCountryName(countryName);
   country.setPopulation(population);
   return country;// returned to Country bean of showCountry() method set as @ModelAttribute
}
@RequestMapping("/country")
public String showCountry(@ModelAttribute Country country, ModelMap model) {
   System.out.println("Country Name : " + country.getCountryName());
   System.out.println("Population : " + country.getPopulation());
   //return new ModelAndView("countryDetails", "countryName", country.getCountryName());
   model.addAttribute("countryName", country.getCountryName());
   model.addAttribute("population", country.getPopulation());
   return "countryDetails";//countryDetails.jsp
}
```

spring
MVC