Spring REST

Created by :
Sangeeta Joshi

Create a resource controller

```
package hello;

import java.util.concurrent.atomic.AtomicLong;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GreetingController {

    private static final String template = "Hello, %s!";
    private final AtomicLong counter = new AtomicLong();

    @RequestMapping("/greeting")
    public Greeting greeting(@RequestParam(value="name", defaultValue="World") String name) {
        return new Greeting(counter.incrementAndGet(),
                            String.format(template, name));
    }
}
```

- The above example does not specify GET vs. PUT, POST, and so forth, because @RequestMapping maps all HTTP operations by default.
- Use @RequestMapping(method=GET) to narrow this mapping.

- A key difference between a traditional MVC controller and the RESTful web service controller is :
  - the way that the HTTP response body is created.
  - Rather than relying on a view technology to perform server-side rendering of the greeting data to HTML, RESTful web service controller simply populates and returns a Greeting object.
  - The object data will be written directly to the HTTP response as JSON.

- We use Spring 4's new ***@RestController*** annotation, which marks the class as a controller
- And every method returns a ***domain object*** instead of a view.
- It's shorthand for ***@Controller*** and ***@ResponseBody*** rolled together.

- The Greeting object must be converted to JSON.
- With Spring's HTTP ***message converter support***, we don't need to do this conversion manually.
- When Jackson 2 is on the classpath, Spring's Mapping Jackson2HttpMessageConverter is automatically chosen to convert the Greeting instance to JSON.

- Status codes in the 100x range (from 100-199) are informational, and describe the processing for the request.
- Status codes in the 200x range (from 200-299) indicate the action requested by the client was received, understood, accepted and processed successfully
- Status codes in the 300x range (from 300-399) indicate that the client must take additional action to complete the request, such as following a redirect
- Status codes in the 400x range (from 400-499) is intended for cases in which the client seems to have erred and must correct the request before continuing. The aforementioned 404 is an example of this.
- Status codes in the 500x range (from 500-599) is intended for cases where the server failed to fulfill an apparently valid request.