@Controller vs. @RestController

The key difference between @Controller vs. @RestController is how the HTTP response is getting created.

In Spring MVC, @RestController = **@Controller** + **@ResponseBody**

Any method annotated with @ResponseBody, spring automatically convert the return value and written into HTTP response body

@Controller is to create a Map of model object and find a view

```
@Controller
@RequestMapping("customer")
public class Customer {
   Customer customer = new Customer ();
   @RequestMapping(value = "/{name}", method = RequestMethod.GET, produce
s = "application/json")
   public @ResponseBody Customer getCustomer(@PathVariable String name) {
     customer.setName(name);
     customer.setEmail("test@mycompany.com");
   return customer;
   }
}
```

```
@Target(value=TYPE)
@Retention(value=RUNTIME)
@Documented
@Component
public @interface Controller
```

Using @RestController

@RestController simply returns the object and object data is directly written into HTTP response as JSON or XML.

Spring v4.0 onwards

```
@RestController
@RequestMapping("customer")
```

```
public class Customer {
   Customer customer = new Customer ();
   @RequestMapping(value = "/{name}", method = RequestMethod.GET, produce
s = "application/json")
   public  Customer getCustomer(@PathVariable String name) {
      customer.setName(name);
      customer.setEmail("test@mycompany.com");
   return customer;
   }
}
```

Declaration of @RestController looks like

```
@Target(value=TYPE)
@Retention(value=RUNTIME)
@Documented
@Controller
@ResponseBody
public @interface RestController
```