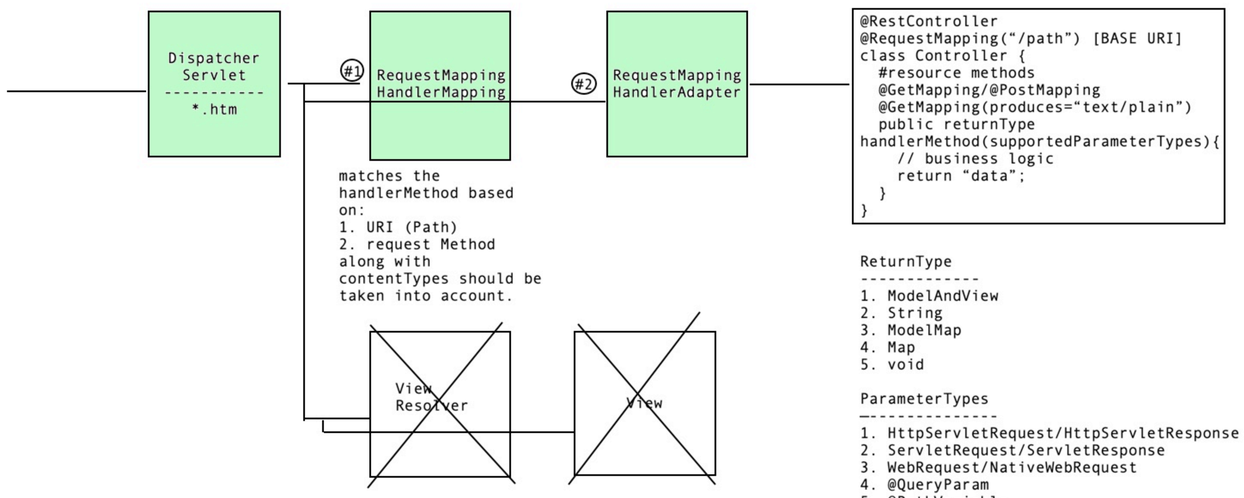
MessageConverter




ReturnType
---
1. ModelAndView
2. String
3. ModelMap
4. Map
5. void

ParameterTypes
---
1. HttpServletRequest/HttpServletResponse
2. ServletRequest/ServletResponse
3. WebRequest/NativeWebRequest
4. @QueryParam
5. @PathVariable
6. HttpSession
7. SessionStatus
8. @ModelAttribute Object
9. BindingResult
10. @SessionAttribute

Request
1. URI
|-query parameters - @RequestParam
|-path parameters - @PathVariable
|-matrix parameters - @MatrixVariable
programmatic api (jaxrs api)|
URIInfo | ServletRequest
PathSegment |
---
2. HEADERS
|-headers - @HeaderParam |
|-cookies - @CookieParam | ServletRequest
programmatic api (jaxrs api) |
HttpHeaders
---
3. Body
|-any data
(data exchange formats like: json/xml/ yaml)

Content Handlers

```
@Path(“/product”)
class ProductResource {
  @POST
  @Consumes(“application/json”)
  @Produces(“application/json”)
  public ProductDetails addProduct(Product product) {
    // extract the data from json format (productName, manufacturer, price, color)
    // convert into java object
    // business logic
    // persistence logic
    // output object
    // converting return value object into json format
    return stringjson;
  }
}
```

Custom Content Handlers
MessageBodyReaders
MessageBodyWriters

```
@RestController
@RequestMapping(“/product”)
class ProductApiController {

  @PostMapping(consumes = {MediaType.TEXT_PLAIN_VALUE},
               produces= {MediaType.TEXT_PLAIN_VALUE})
  public String addProduct(@RequestBody Product product) {
  }
}
```

@ModelAttribute = is used for binding the request body data into model object (bean object). but it works only when the request body contains the data as www-form-url-encoded format, then only it extracts the request body and binds to the attributes of the form object
for eg..

```
public String addProduct(@ModelAttribute(“productForm”) ProductForm form) {}
```

@RequestBody
Incase of restful resource, the data that is sent aspart of the request body could be of anyType (json, yaml, xml, plain/text etc). Inorder to indicate read whatever the request data and bind to the parameter of the resource method we need to use @RequestBody.

The parameterType into which it can extract the requestBody data and bind cannot be anything, it should acceptable parameterTypes like
1. String
etc

This way for receiving the requestbody data is called “raw representation format”.

```
interface HttpMessageConverter {
  boolean canRead(Class<?> clazz, @Nullable MediaType mediaType);
  boolean canWrite(Class<?> clazz, @Nullable MediaType mediaType);
  List<MediaType> getSupportedMediaTypes();

  default List<MediaType> getSupportedMediaTypes(Class<?> clazz) {
    return !this.canRead(clazz, (MediaType)null) && !this.canWrite(clazz, (MediaType)null) ? Collections.emptyList() : this.getSupportedMediaTypes();
  }

  T read(Class<? extends T> clazz, HttpInputMessage inputMessage) throws IOException, HttpMessageNotReadableException;
  void write(T t, @Nullable MediaType contentType, HttpOutputMessage outputMessage) throws IOException, HttpMessageNotWritableException;
}
```

Write our own implementation of HttpMessageConverter with logic for converting json into object and object into json, then register with DispatcherServlet.

```
class WebMvcConfig implements WebMvcConfigurer {
  public void configureMessageConverters(List<HttpMessageConverter<?>> messageConverters) {
    messageConverters.add(new JsonHttpMessageConverterImpl());
  }
}
```

How to customize the response and dispatch it to the client?
By default when the resource/controller methods are turning the return value, the RMHA or dispatcherservlet will constructs an Http Response with default status: 200 , with http standard headers along with response body as returnValue we returned and dispatches to the client.

But sometimes we want to customize the response we want to dispatch to the client, how can we customize the response from a resource method?
ResponseEntity class.

```
@GetMapping(value=“/{accountNo}/balance”, produces={MediaType.TEXT_PLAIN_VALUE}
public ResponseEntity balance(@PathVariable(“accountNo”) String accountNo) {
  return ResponseEntity.ok(“9283.3”).header(“h1”, “v1”);
}
```