**HandlerAdapter**

The HandlerAdapter is basically an interface which facilitates the handling of HTTP requests in a very flexible manner in Spring MVC.

It's used in conjunction with the HandlerMapping, which maps a method to a specific URL.

The DispatcherServlet then uses a HandlerAdapter to invoke this method. The servlet doesn't invoke the method directly – it basically serves as a bridge between itself and the handler objects, leading to a loosely coupling design.

**AnnotationMethodHandlerAdapter**

This adapter class is used to execute the methods that are annotated with @RequestMapping annotation. It is used to map the methods based on HTTP methods and HTTP paths.

The mapping class for this adapter is DefaultAnnotationHandlerMapping, which is used to process the @RequestMapping annotation at the type level and AnnotationMethodHandlerAdaptor is used to process at a method level.

These two classes are already registered by the framework when the DispatcherServlet is initialized. However, if the other handler adapters are already defined, then we need to define it as well in the configuration file.

## Model

Let's start with the most basic concept here – the Model.

Simply put, the model can supply attributes used for rendering views.

To provide a view with usable data, we simply add this data to its Model object. Additionally, maps with attributes can be merged with Model instances:

@GetMapping("/showViewPage")

**public** String **passParametersWithModel**(Model model) {

Map<String, String> map = **new** HashMap<>();

map.put("spring", "mvc");

model.addAttribute("message", "Baeldung");

model.mergeAttributes(map);

**return** "viewPage";

}

## ****4.****ModelMap

Just like the Model interface above, ModelMap is also used to pass values to render a view.

The advantage of ModelMap is it gives us the ability to pass a collection of values and treat these values as if they were within a Map:

@GetMapping("/printViewPage")

**public** String **passParametersWithModelMap**(ModelMap map) {

map.addAttribute("welcomeMessage", "welcome");

map.addAttribute("message", "Baeldung");

**return** "viewPage";

}

## ****5.****ModelAndView

The final interface to pass values to a view is the ModelAndView.

This interface allows us to pass all the information required by Spring MVC in one return:

@GetMapping("/goToViewPage")

**public** ModelAndView **passParametersWithModelAndView**() {

ModelAndView modelAndView = **new** ModelAndView("viewPage");

modelAndView.addObject("message", "Baeldung");

**return** modelAndView;

}