Spring RequestMapping Supported Method Arguments

The following are the supported method arguments:

- Request or response objects (Servlet API) Choose any specific request or response type, for example `ServletRequest` or `HttpServletRequest` .
- Session object (Servlet API) of type `HttpSession` . An argument of this type enforces the presence of a corresponding session. As a consequence, such an argument is never `null` .

  

  Note

  Session access may not be thread-safe, in particular in a Servlet environment. Consider setting the `RequestMappingHandlerAdapter` 's "synchronizeOnSession" flag to "true" if multiple requests are allowed to access a session concurrently.

- `org.springframework.web.context.request.WebRequest` or `org.springframework.web.context.request.NativeWebRequest` . Allows for generic request parameter access as well as request/session attribute access, without ties to the native Servlet/Portlet API.
- `java.util.Locale` for the current request locale, determined by the most specific locale resolver available, in effect, the configured `LocaleResolver` in a Servlet environment.
- `java.io.InputStream / java.io.Reader` for access to the request's content. This value is the raw InputStream/Reader as exposed by the Servlet API.
- `java.io.OutputStream / java.io.Writer` for generating the response's content. This value is the raw OutputStream/Writer as exposed by the Servlet API.
- `org.springframework.http.HttpMethod` for the HTTP request method.
- `java.security.Principal` containing the currently authenticated user.
- `@PathVariable` annotated parameters for access to URI template variables. See the section called "URI Template Patterns".
- `@MatrixVariable` annotated parameters for access to name-value pairs located in URI path segments. See the section called "Matrix Variables".
- `@RequestParam` annotated parameters for access to specific Servlet request parameters. Parameter values are converted to the declared method argument type. See the section called "Binding request parameters to method parameters with @RequestParam".
- `@RequestHeader` annotated parameters for access to specific Servlet request HTTP headers. Parameter values are converted to the declared method argument type.
- `@RequestBody` annotated parameters for access to the HTTP request body. Parameter values are converted to the declared method argument type using `HttpMessageConverter` s. See the section called "Mapping the request body with the **@RequestBody** annotation".
- `@RequestPart` annotated parameters for access to the content of a "multipart/form-data" request part. See Section 16.11.5, "Handling a file upload request from programmatic clients" and Section 16.11, "Spring's multipart (file upload) support".
- `HttpEntity<?>` parameters for access to the Servlet request HTTP headers and contents. The request stream will be converted to the entity body using `HttpMessageConverter` s. See HttpEntity">the section called "Using HttpEntity".
- `java.util.Map / org.springframework.ui.Model / org.springframework.ui.ModelMap` for enriching the implicit model that is exposed to the web view.
- `org.springframework.web.servlet.mvc.support.RedirectAttributes` to specify the exact set of attributes to use in case of a redirect and also to add flash attributes (attributes stored temporarily on the server-side to make them available to the request after the redirect). `RedirectAttributes` is used instead of the implicit model if the method returns a "redirect:" prefixed view name or `RedirectView` .
- Command or form objects to bind request parameters to bean properties (via setters) or directly to fields, with customizable type conversion, depending on `@InitBinder` methods and/or the HandlerAdapter configuration. See the `webBindingInitializer` property on `RequestMappingHandlerAdapter` . Such command objects along with their validation results will be exposed as model attributes by default, using the command class class name - e.g. model attribute "orderAddress" for a command object of type "some.package.OrderAddress". The `ModelAttribute` annotation can be used on a method argument to customize the model attribute name used.
- `org.springframework.validation.Errors / org.springframework.validation.BindingResult` validation results for a preceding command or form object (the immediately preceding method argument).
- `org.springframework.web.bind.support.SessionStatus` status handle for marking form processing as complete, which triggers the cleanup of session attributes that have been indicated by the `@SessionAttributes` annotation at the handler type level.
- `org.springframework.web.util.UriComponentsBuilder` a builder for preparing a URL relative to the current request's host, port, scheme, context path, and the literal part of the servlet mapping.

The `Errors` or `BindingResult` parameters have to follow the model object that is being bound immediately as the method signature might have

more that one model **object** and Spring will create a separate `BindingResult` **instance** for each of them so the following sample won't work:

BindingResult and **@ModelAttribute"**

Invalid ordering of BindingResult and @ModelAttribute.

```
@RequestMapping(method = RequestMethod.POST)
public String processSubmit(@ModelAttribute("pet") Pet pet, Model model, BindingResult result) { ... }
```

Note, that there is a `Model` parameter in between `Pet` and `BindingResult` . To get this working you have to reorder the parameters as follows:

```
@RequestMapping(method = RequestMethod.POST)
public String processSubmit(@ModelAttribute("pet") Pet pet, BindingResult result, Model model) { ... }
```