Server-Creation DSL

RestExpress server = new RestExpress()...

Server Config

.setName(String) - sets the service suite name displayed to the console at startup time.

.setBaseUrl(String) - sets the protocol, host, port used to return links.

.setPort(int) - use ‘bind(int)’ instead in the Server Execution section below.

.setExecutorThreadCount(int) - sets maximum number of threads in the executor pool (maximum number of simultaneous blocking operations).

.setIoThreadCount(int) - sets the number of non-blocking, front-end i/o worker threads (defaults to 2 x #cores, which should be acceptable)

Serialization

.putResponseProcessor(String format, ResponseProcessor) - add a new serializer (ResponseProcessor) for a given .{format} type.

.setDefaultFormat(String) - set the default returned format. RestExpress default is ‘json’.

.noJson() - do not support JSON serialization or responses.

.noXml() - do not support XML serialization or response.

.supportJson() - support JSON as the default response (this is the default).

.supportJson(boolean isDefault) - support JSON, optionally setting it as the default response (if isDefault is true).

.supportXml() - support XML as the default response format.

.supportXml(boolean isDefault) - support XML, optionally setting it as the default response (if isDefault is true).

Socket-Level Config

.setConnectTimeoutMillis(int)

.setKeepAlive(boolean)
.setReceiveBufferSize(int)
.setReuseAddress(boolean)
.setSoLinger(int)
.setUseTcpNoDelay(boolean)

Processors/Observers/Plugins

.addPreprocessor(Preprocessor) - add a preprocessor to the preprocessor chain. Preprocessors are called before the controller method is called. Preprocessors are called in the order in which they are added. If a preprocessor throws an exception, the rest of the chain is aborted and the controller method is not called. An error response is returned to the client immediately.

.addPostprocessor(Postprocessor) - add a postprocessor to the postprocessor chain. Preprocessors are called after the controller method is called and only if the controller method completes successfully (does not throw an exception). If an exception is throw from within one postprocessor, the rest of the postprocessor chain does not process and an error response is returned to the client immediately, if applicable).

.addFinallyProcessor(Postprocessor) - add a postprocessor to the finally-processor chain. Finally processors are executed whether exceptions occur in the upstream chain or not. They will always execute and if one throws an exception, the rest of the finally processors will be executed anyway.

.addMessageObserver(MessageObserver) - add an observer to the message observer chain. MessageObserver instances allow you to get in the game during several times during request/response processing, to perform actions or simply to log, time, or collect analytics.

Exception Handling

.mapException(Class<T> fromException, Class<U> toException) - when fromException is thrown, instead throw toException (e.g. throw a RestExpress-related exception instead of a checked exception).

.setExceptionMap(ExceptionMapping) - Used instead of mapException(), above. This method allows you to create your own ExceptionMapping and assign it at once, instead of line by line calling mapException().

Server Execution

.bind() - start the RestExpress server listening on the port declared in the Server-Creation DSL. no recommended. Use bind(int) instead.

.bind(int) - start the RestExpress server listening on the specified port.

.awaitShutdown() - wait for control-C or a termination signal to be received before shutting down. Executes a JVM shutdown hook. You can create your own shutdown hook, if desired, calling server.shutdown() when the shutdown hook executes.

Example:

```
RestExpress server = new RestExpress()
        .setName("Sample Service")
        .setBaseUrl("http://localhost:8081")
        .setDefaultFormat(Format.JSON)
        .putResponseProcessor(Format.JSON, ResponseProcessors.json())
        .putResponseProcessor(Format.XML, ResponseProcessors.xml())
        .putResponseProcessor(Format.WRAPPED_JSON, ResponseProcessors.wrappedJson())
        .putResponseProcessor(Format.WRAPPED_XML, ResponseProcessors.wrappedXml())
        .addPreprocessor(new AuthenticationPreprocessor())
        .addPreprocessor(new AuthorizationPreprocessor())
        .addPostprocessor(new LastModifiedHeaderPostprocessor())
        .addMessageObserver(new SimpleConsoleLogMessageObserver());
```

Route-Definition DSL

server.uri(String pattern, Object controller) or
server.regex(String regex, Object controller)

where:

pattern = a string defining the resource URI path, containing optional parameters and optional format parameter after a period (‘.’). The parameters are alpha-numeric strings surrounded by curly braces (e.g. “{personId}”).

An example URI pattern is: *“/users/{userId}/enrollments.{format}”*

Route DSL methods:

.method(HttpMethod...) - denote the HTTP method(s) this route supports. In the controller, GET maps to read(), POST maps to create(), PUT maps to update() and DELETE maps to delete().

.action(String controllerMethod, HttpMethod) - use this form when an HTTP method maps to a non-standard operation mapped via method() (e.g. .action(“readAll”, HttpMethod.GET)).

.name(String) - a unique name for this route, used to retrieve the URL pattern in the controller to create hypermedia links.

.alias(String pattern) - uri() method only. Support an additional URL pattern for this route.

.baseUrl(String) - Same as Server-Creation DSL. Sets protocol, host, port returned on links for this route.

.defaultFormat(String) - sets the default format for responses on this route.

.noSerialization() - do not serialize responses (controller for appropriately format the response).

.performSerialization() - the default. Opposite of noSerialization().

.flag(String) - set a flag on the request with this route is called (for example a ‘public’ not-authenticate route). This flag is available in preprocessors, controllers via Request.isFlagged(String).

.parameter(String, Object) - set a name/value pair on the request when this route is called. This parameter is available via Request.getParameter(String).

Example:

```
server.uri("/samples.{format}", config.getSampleController())
	.method(HttpMethod.POST)
	.action("readAll", HttpMethod.GET)
	.name(Constants.Routes.SAMPLE_COLLECTION);

server.uri("/samples/{sampleId}.{format}", config.getSampleController())
	.method(HttpMethod.GET, HttpMethod.PUT, HttpMethod.DELETE)
	.name(Constants.Routes.SINGLE_SAMPLE);
```