org.springframework.web.client

Class RestTemplate

java.lang.Object
    org.springframework.http.client.support.HttpAccessor
        org.springframework.http.client.support.InterceptingHttpAccessor
            org.springframework.web.client.RestTemplate

All Implemented Interfaces:

RestOperations

```
public class RestTemplate
extends InterceptingHttpAccessor
implements RestOperations
```

Spring's central class for synchronous client-side HTTP access. It simplifies communication with HTTP servers, and enforces RESTful principles. It handles HTTP connections, leaving application code to provide URLs (with possible template variables) and extract results.

Note: by default the RestTemplate relies on standard JDK facilities to establish HTTP connections. You can switch to use a different HTTP library such as Apache HttpComponents, Netty, and OkHttp through the `HttpAccessor.setRequestFactory(org.springframework.http.client.ClientHttpRequestFactory)` property.

The main entry points of this template are the methods named after the six main HTTP methods:

HTTP method	RestTemplate methods
DELETE	`delete(java.lang.String, java.lang.Object...)`
GET	`getForObject(java.lang.String, java.lang.Class<T>, java.lang.Object...)`
	`getForEntity(java.lang.String, java.lang.Class<T>, java.lang.Object...)`
HEAD	`headForHeaders(java.lang.String, java.lang.Object...)`
OPTIONS	`optionsForAllow(java.lang.String, java.lang.Object...)`
POST	`postForLocation(java.lang.String, java.lang.Object, java.lang.Object...)`
	`postForObject(java.lang.String, java.lang.Object, java.lang.Class<T>, java.lang.Object...)`
PUT	`put(java.lang.String, java.lang.Object, java.lang.Object...)`
any	`exchange(java.lang.String, org.springframework.http.HttpMethod, org.springframework.http.HttpEntity<?>, java.lang.Class<T>, java.lang.Object...)`
	`execute(java.lang.String, org.springframework.http.HttpMethod, org.springframework.web.client.RequestCallback, org.springframework.web.client.ResponseExtractor<T>, java.lang.Object...)`

In addition the `exchange` and `execute` methods are generalized versions of the above methods and can be used to support additional, less frequent combinations (e.g. HTTP PATCH, HTTP PUT with response body, etc.). Note however that the underlying HTTP library used must also support the desired combination.

For each HTTP method there are three variants: two accept a URI template string and URI variables (array or map) while a third accepts a `URI`. Note that for URI templates it is assumed encoding is necessary, e.g. `restTemplate.getForObject("http://example.com/hotel list")` becomes `"http://example.com/hotel%20list"`. This also means if the URI template or URI variables are already encoded, double encoding will occur, e.g. `http://example.com/hotel%20list` becomes `http://example.com/hotel%2520list`). To avoid that use a `URI` method variant to provide (or re-use) a previously encoded URI. To prepare such an URI with full control over encoding, consider using `UriComponentsBuilder`.

Internally the template uses `HttpMessageConverter` instances to convert HTTP messages to and from POJOs. Converters for the main mime types are registered by default but you can also register additional converters via `setMessageConverters(java.util.List<org.springframework.http.converter.HttpMessageConverter<?>>)`.

This template uses a `SimpleClientHttpRequestFactory` and a `DefaultResponseErrorHandler` as default strategies for creating HTTP connections or handling HTTP errors, respectively. These defaults can be overridden through `HttpAccessor.setRequestFactory(org.springframework.http.client.ClientHttpRequestFactory)` and `setErrorHandler(org.springframework.web.client.ResponseErrorHandler)` respectively.

Since:

3.0

Author:

Arjen Poutsma, Brian Clozel, Roy Clarkson, Juergen Hoeller

See Also:

`HttpMessageConverter`, `RequestCallback`, `ResponseExtractor`, `ResponseErrorHandler`, `AsyncRestTemplate`

Field Summary

Fields inherited from class org.springframework.http.client.support.HttpAccessor

`logger`

Constructor Summary

Constructors

Constructor and Description
`RestTemplate()` Create a new instance of the **`RestTemplate`** using default settings.
`RestTemplate(ClientHttpRequestFactory requestFactory)` Create a new instance of the **`RestTemplate`** based on the given **`ClientHttpRequestFactory`**.
`RestTemplate(List<HttpMessageConverter<?>> messageConverters)` Create a new instance of the **`RestTemplate`** using the given list of **`HttpMessageConverter`** to use

Method Summary

All Methods   Instance Methods   Concrete Methods

Modifier and Type	Method and Description
`protected <T> RequestCallback`	`acceptHeaderRequestCallback(Class<T> responseType)` Returns a request callback implementation that prepares the request `Accept` headers based on the given response type and configured **message converters**.
`void`	`delete(String url, Map<String,?> uriVariables)` Delete the resources at the specified URI.
`void`	`delete(String url, Object... uriVariables)` Delete the resources at the specified URI.
`void`	`delete(URI url)` Delete the resources at the specified URL.
`protected <T> T`	`doExecute(URI url, HttpMethod method, RequestCallback requestCallback, ResponseExtractor<T> responseExtractor)`

	Execute the given method on the provided URI.
`<T> ResponseEntity<T>`	`exchange(RequestEntity<?> requestEntity, Class<T> responseType)` Execute the request specified in the given `RequestEntity` and return the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`exchange(RequestEntity<?> requestEntity, ParameterizedTypeReference<T> responseType)` Execute the request specified in the given `RequestEntity` and return the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`exchange(String url, HttpMethod method, HttpEntity<?> requestEntity, Class<T> responseType, Map<String,?> uriVariables)` Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`exchange(String url, HttpMethod method, HttpEntity<?> requestEntity, Class<T> responseType, Object... uriVariables)` Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`exchange(String url, HttpMethod method, HttpEntity<?> requestEntity, ParameterizedTypeReference<T> responseType, Map<String,?> uriVariables)` Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`exchange(String url, HttpMethod method, HttpEntity<?> requestEntity, ParameterizedTypeReference<T> responseType, Object... uriVariables)` Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`exchange(URI url, HttpMethod method, HttpEntity<?> requestEntity, Class<T> responseType)` Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`exchange(URI url, HttpMethod method, HttpEntity<?> requestEntity, ParameterizedTypeReference<T> responseType)` Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.
`<T> T`	`execute(String url, HttpMethod method, RequestCallback requestCallback, ResponseExtractor<T> responseExtractor, Map<String,?> uriVariables)` Execute the HTTP method to the given URI template, preparing the request with the `RequestCallback`, and

	reading the response with a **ResponseExtractor**.
`<T> T`	**execute**(**String** url, **HttpMethod** method, **RequestCallback** requestCallback, **ResponseExtractor**<T> responseExtractor, **Object**... uriVariables) Execute the HTTP method to the given URI template, preparing the request with the **RequestCallback**, and reading the response with a **ResponseExtractor**.
`<T> T`	**execute**(**URI** url, **HttpMethod** method, **RequestCallback** requestCallback, **ResponseExtractor**<T> responseExtractor) Execute the HTTP method to the given URL, preparing the request with the **RequestCallback**, and reading the response with a **ResponseExtractor**.
ResponseErrorHandler	**getErrorHandler**() Return the error handler.
<T> **ResponseEntity**<T>	**getForEntity**(**String** url, **Class**<T> responseType, **Map**<**String**,?> uriVariables) Retrieve a representation by doing a GET on the URI template.
<T> **ResponseEntity**<T>	**getForEntity**(**String** url, **Class**<T> responseType, **Object**... uriVariables) Retrieve an entity by doing a GET on the specified URL.
<T> **ResponseEntity**<T>	**getForEntity**(**URI** url, **Class**<T> responseType) Retrieve a representation by doing a GET on the URL .
`<T> T`	**getForObject**(**String** url, **Class**<T> responseType, **Map**<**String**,?> uriVariables) Retrieve a representation by doing a GET on the URI template.
`<T> T`	**getForObject**(**String** url, **Class**<T> responseType, **Object**... uriVariables) Retrieve a representation by doing a GET on the specified URL.
`<T> T`	**getForObject**(**URI** url, **Class**<T> responseType) Retrieve a representation by doing a GET on the URL .
List<**HttpMessageConverter**<?>>	**getMessageConverters**() Return the message body converters.
UriTemplateHandler	**getUriTemplateHandler**() Return the configured URI template handler.
protected void	**handleResponse**(**URI** url, **HttpMethod** method, **ClientHttpResponse** response) Handle the given response, performing appropriate logging and invoking the **ResponseErrorHandler** if necessary.
protected **ResponseExtractor**<**HttpHeaders**>	**headersExtractor**() Returns a response extractor for **HttpHeaders**.
HttpHeaders	**headForHeaders**(**String** url, **Map**<**String**,?> uriVariables) Retrieve all headers of the resource specified by the URI template.

`HttpHeaders`	`headForHeaders(String url, Object... uriVariables)` Retrieve all headers of the resource specified by the URI template.
`HttpHeaders`	`headForHeaders(URI url)` Retrieve all headers of the resource specified by the URL.
`protected <T> RequestCallback`	`httpEntityCallback(Object requestBody)` Returns a request callback implementation that writes the given object to the request stream.
`protected <T> RequestCallback`	`httpEntityCallback(Object requestBody, Type responseType)` Returns a request callback implementation that writes the given object to the request stream.
`Set<HttpMethod>`	`optionsForAllow(String url, Map<String,?> uriVariables)` Return the value of the Allow header for the given URI.
`Set<HttpMethod>`	`optionsForAllow(String url, Object... uriVariables)` Return the value of the Allow header for the given URI.
`Set<HttpMethod>`	`optionsForAllow(URI url)` Return the value of the Allow header for the given URL.
`<T> T`	`patchForObject(String url, Object request, Class<T> responseType, Map<String,?> uriVariables)` Update a resource by PATCHing the given object to the URI template, and return the representation found in the response.
`<T> T`	`patchForObject(String url, Object request, Class<T> responseType, Object... uriVariables)` Update a resource by PATCHing the given object to the URI template, and return the representation found in the response.
`<T> T`	`patchForObject(URI url, Object request, Class<T> responseType)` Update a resource by PATCHing the given object to the URL, and return the representation found in the response.
`<T> ResponseEntity<T>`	`postForEntity(String url, Object request, Class<T> responseType, Map<String,?> uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the response as `HttpEntity`.
`<T> ResponseEntity<T>`	`postForEntity(String url, Object request, Class<T> responseType, Object... uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the response as `ResponseEntity`.
`<T> ResponseEntity<T>`	`postForEntity(URI url, Object request, Class<T> responseType)` Create a new resource by POSTing the given object to the URL, and returns the response as `ResponseEntity`.
`URI`	`postForLocation(String url, Object request,`

	`Map<String,?> uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the value of the `Location` header.
`URI`	`postForLocation(String url, Object request, Object... uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the value of the `Location` header.
`URI`	`postForLocation(URI url, Object request)` Create a new resource by POSTing the given object to the URL, and returns the value of the `Location` header.
`<T> T`	`postForObject(String url, Object request, Class<T> responseType, Map<String,?> uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the representation found in the response.
`<T> T`	`postForObject(String url, Object request, Class<T> responseType, Object... uriVariables)` Create a new resource by POSTing the given object to the URI template, and returns the representation found in the response.
`<T> T`	`postForObject(URI url, Object request, Class<T> responseType)` Create a new resource by POSTing the given object to the URL, and returns the representation found in the response.
`void`	`put(String url, Object request, Map<String,?> uriVariables)` Creates a new resource by PUTting the given object to URI template.
`void`	`put(String url, Object request, Object... uriVariables)` Create or update a resource by PUTting the given object to the URI.
`void`	`put(URI url, Object request)` Creates a new resource by PUTting the given object to URL.
`protected <T> ResponseExtractor<ResponseEntity<T>>`	`responseEntityExtractor(Type responseType)` Returns a response extractor for `ResponseEntity`.
`void`	`setDefaultUriVariables(Map<String,?> defaultUriVariables)` Configure default URI variable values.
`void`	`setErrorHandler(ResponseErrorHandler errorHandler)` Set the error handler.
`void`	`setMessageConverters(List<HttpMessageConverter<?>> messageConverters)` Set the message body converters to use.
`void`	`setUriTemplateHandler(UriTemplateHandler handler)` Configure the `UriTemplateHandler` to use to expand URI templates.

Methods inherited from class org.springframework.http.client.support.InterceptingHttpAccessor

`getInterceptors, getRequestFactory, setInterceptors`

Methods inherited from class org.springframework.http.client.support.HttpAccessor

`createRequest, setRequestFactory`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

RestTemplate

```
public RestTemplate()
```

Create a new instance of the `RestTemplate` using default settings. Default `HttpMessageConverters` are initialized.

RestTemplate

```
public RestTemplate(ClientHttpRequestFactory requestFactory)
```

Create a new instance of the `RestTemplate` based on the given `ClientHttpRequestFactory`.

Parameters:

`requestFactory - HTTP request factory to use`

See Also:

`SimpleClientHttpRequestFactory, HttpComponentsClientHttpRequestFactory`

RestTemplate

```
public RestTemplate(List<HttpMessageConverter<?>> messageConverters)
```

Create a new instance of the `RestTemplate` using the given list of `HttpMessageConverter` to use

Parameters:

`messageConverters - the list of HttpMessageConverter to use`

Since:

3.2.7

Method Detail

setMessageConverters

```
public void setMessageConverters(List<HttpMessageConverter<?>> messageConverters)
```

Set the message body converters to use.

These converters are used to convert from and to HTTP requests and responses.

getMessageConverters

`public List<HttpMessageConverter<?>> getMessageConverters()`

Return the message body converters.

setErrorHandler

`public void setErrorHandler(ResponseErrorHandler errorHandler)`

Set the error handler.

By default, RestTemplate uses a `DefaultResponseErrorHandler`.

getErrorHandler

`public ResponseErrorHandler getErrorHandler()`

Return the error handler.

setDefaultUriVariables

`public void setDefaultUriVariables(Map<String,?> defaultUriVariables)`

Configure default URI variable values. This is a shortcut for:

```
DefaultUriTemplateHandler handler = new DefaultUriTemplateHandler();
handler.setDefaultUriVariables(...);

RestTemplate restTemplate = new RestTemplate();
restTemplate.setUriTemplateHandler(handler);
```

Parameters:

`defaultUriVariables - the default URI variable values`

Since:

`4.3`

setUriTemplateHandler

`public void setUriTemplateHandler(UriTemplateHandler handler)`

Configure the `UriTemplateHandler` to use to expand URI templates. By default the `DefaultUriTemplateHandler` is used which relies on Spring's URI template support and exposes several useful properties that customize its behavior for encoding and for prepending a common base URL. An alternative implementation may be used to plug an external URI template library.

Parameters:

`handler - the URI template handler to use`

getUriTemplateHandler

`public UriTemplateHandler getUriTemplateHandler()`

Return the configured URI template handler.

getForObject

```
public <T> T getForObject(String url,
                          Class<T> responseType,
                          Object... uriVariables)
                   throws RestClientException
```

Description copied from interface: `RestOperations`
Retrieve a representation by doing a GET on the specified URL. The response (if any) is converted and returned.

URI Template variables are expanded using the given URI variables, if any.

Specified by:
`getForObject` in interface `RestOperations`

Parameters:
`url` - the URL

`responseType` - the type of the return value

`uriVariables` - the variables to expand the template

Returns:
the converted object

Throws:
`RestClientException`

getForObject

```
public <T> T getForObject(String url,
                          Class<T> responseType,
                          Map<String,?> uriVariables)
                   throws RestClientException
```

Description copied from interface: `RestOperations`
Retrieve a representation by doing a GET on the URI template. The response (if any) is converted and returned.

URI Template variables are expanded using the given map.

Specified by:
`getForObject` in interface `RestOperations`

Parameters:
`url` - the URL

`responseType` - the type of the return value

`uriVariables` - the map containing variables for the URI template

Returns:
the converted object

Throws:
`RestClientException`

getForObject

```
public <T> T getForObject(URI url,
                          Class<T> responseType)
                   throws RestClientException
```

Description copied from interface: `RestOperations`
Retrieve a representation by doing a GET on the URL . The response (if any) is converted and returned.

Specified by:

`getForObject` in interface `RestOperations`

Parameters:

`url` - the URL

`responseType` - the type of the return value

Returns:

the converted object

Throws:

`RestClientException`

getForEntity

```
public <T> ResponseEntity<T> getForEntity(String url,
                                          Class<T> responseType,
                                          Object... uriVariables)
                                   throws RestClientException
```

Description copied from interface: `RestOperations`

Retrieve an entity by doing a GET on the specified URL. The response is converted and stored in an `ResponseEntity`.

URI Template variables are expanded using the given URI variables, if any.

Specified by:

`getForEntity` in interface `RestOperations`

Parameters:

`url` - the URL

`responseType` - the type of the return value

`uriVariables` - the variables to expand the template

Returns:

the entity

Throws:

`RestClientException`

getForEntity

```
public <T> ResponseEntity<T> getForEntity(String url,
                                          Class<T> responseType,
                                          Map<String,?> uriVariables)
                                   throws RestClientException
```

Description copied from interface: `RestOperations`

Retrieve a representation by doing a GET on the URI template. The response is converted and stored in an `ResponseEntity`.

URI Template variables are expanded using the given map.

Specified by:

`getForEntity` in interface `RestOperations`

Parameters:

`url` - the URL

`responseType` - the type of the return value

`uriVariables` - the map containing variables for the URI template

Returns:

the converted object

Throws:

RestClientException

getForEntity

```
public <T> ResponseEntity<T> getForEntity(URI url,
                                          Class<T> responseType)
                                   throws RestClientException
```

Description copied from interface: RestOperations

Retrieve a representation by doing a GET on the URL . The response is converted and stored in an ResponseEntity.

Specified by:

getForEntity in interface RestOperations

Parameters:

url - the URL

responseType - the type of the return value

Returns:

the converted object

Throws:

RestClientException

headForHeaders

```
public HttpHeaders headForHeaders(String url,
                                  Object... uriVariables)
                           throws RestClientException
```

Description copied from interface: RestOperations

Retrieve all headers of the resource specified by the URI template.

URI Template variables are expanded using the given URI variables, if any.

Specified by:

headForHeaders in interface RestOperations

Parameters:

url - the URL

uriVariables - the variables to expand the template

Returns:

all HTTP headers of that resource

Throws:

RestClientException

headForHeaders

```
public HttpHeaders headForHeaders(String url,
                                  Map<String,?> uriVariables)
                           throws RestClientException
```

Description copied from interface: RestOperations

Retrieve all headers of the resource specified by the URI template.

URI Template variables are expanded using the given map.

Specified by:

`headForHeaders` in interface `RestOperations`

Parameters:

`url` - the URL

`uriVariables` - the map containing variables for the URI template

Returns:

all HTTP headers of that resource

Throws:

`RestClientException`

headForHeaders

```
public HttpHeaders headForHeaders(URI url)
                           throws RestClientException
```

Description copied from interface: `RestOperations`

Retrieve all headers of the resource specified by the URL.

Specified by:

`headForHeaders` in interface `RestOperations`

Parameters:

`url` - the URL

Returns:

all HTTP headers of that resource

Throws:

`RestClientException`

postForLocation

```
public URI postForLocation(String url,
                           Object request,
                           Object... uriVariables)
                    throws RestClientException
```

Description copied from interface: `RestOperations`

Create a new resource by POSTing the given object to the URI template, and returns the value of the `Location` header. This header typically indicates where the new resource is stored.

URI Template variables are expanded using the given URI variables, if any.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

`postForLocation` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the Object to be POSTed (may be null)

`uriVariables` - the variables to expand the template

Returns:

the value for the Location header

Throws:

RestClientException

See Also:

HttpEntity

postForLocation

```
public URI postForLocation(String url,
                           Object request,
                           Map<String,?> uriVariables)
                    throws RestClientException
```

Description copied from interface: `RestOperations`

Create a new resource by POSTing the given object to the URI template, and returns the value of the `Location` header. This header typically indicates where the new resource is stored.

URI Template variables are expanded using the given map.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

`postForLocation` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the Object to be POSTed (may be null)

`uriVariables` - the variables to expand the template

Returns:

the value for the Location header

Throws:

RestClientException

See Also:

HttpEntity

postForLocation

```
public URI postForLocation(URI url,
                           Object request)
                    throws RestClientException
```

Description copied from interface: `RestOperations`

Create a new resource by POSTing the given object to the URL, and returns the value of the `Location` header. This header typically indicates where the new resource is stored.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

`postForLocation` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the Object to be POSTed (may be null)

Returns:

the value for the Location header

Throws:

RestClientException

See Also:

`HttpEntity`

postForObject

```
public <T> T postForObject(String url,
                           Object request,
                           Class<T> responseType,
                           Object... uriVariables)
                    throws RestClientException
```

Description copied from interface: `RestOperations`

Create a new resource by POSTing the given object to the URI template, and returns the representation found in the response.

URI Template variables are expanded using the given URI variables, if any.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

`postForObject` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the Object to be POSTed (may be null)

`responseType` - the type of the return value

`uriVariables` - the variables to expand the template

Returns:

the converted object

Throws:

`RestClientException`

See Also:

`HttpEntity`

postForObject

```
public <T> T postForObject(String url,
                           Object request,
                           Class<T> responseType,
                           Map<String,?> uriVariables)
                    throws RestClientException
```

Description copied from interface: `RestOperations`

Create a new resource by POSTing the given object to the URI template, and returns the representation found in the response.

URI Template variables are expanded using the given map.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

`postForObject` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the Object to be POSTed (may be null)

`responseType` - the type of the return value uriVariables - the variables to expand the template

Returns:

the converted object

Throws:

RestClientException

See Also:

HttpEntity

postForObject

```
public <T> T postForObject(URI url,
                           Object request,
                           Class<T> responseType)
                    throws RestClientException
```

Description copied from interface: RestOperations

Create a new resource by POSTing the given object to the URL, and returns the representation found in the response.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

postForObject in interface RestOperations

Parameters:

url - the URL

request - the Object to be POSTed (may be null)

responseType - the type of the return value

Returns:

the converted object

Throws:

RestClientException

See Also:

HttpEntity

postForEntity

```
public <T> ResponseEntity<T> postForEntity(String url,
                                           Object request,
                                           Class<T> responseType,
                                           Object... uriVariables)
                                    throws RestClientException
```

Description copied from interface: RestOperations

Create a new resource by POSTing the given object to the URI template, and returns the response as `ResponseEntity`.

URI Template variables are expanded using the given URI variables, if any.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

postForEntity in interface RestOperations

Parameters:

url - the URL

`request` - the Object to be POSTed (may be `null`)

`uriVariables` - the variables to expand the template

Returns:

the converted object

Throws:

`RestClientException`

See Also:

`HttpEntity`

postForEntity

```
public <T> ResponseEntity<T> postForEntity(String url,
                                           Object request,
                                           Class<T> responseType,
                                           Map<String,?> uriVariables)
                                    throws RestClientException
```

Description copied from interface: `RestOperations`

Create a new resource by POSTing the given object to the URI template, and returns the response as `HttpEntity`.

URI Template variables are expanded using the given map.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

`postForEntity` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the Object to be POSTed (may be `null`)

`uriVariables` - the variables to expand the template

Returns:

the converted object

Throws:

`RestClientException`

See Also:

`HttpEntity`

postForEntity

```
public <T> ResponseEntity<T> postForEntity(URI url,
                                           Object request,
                                           Class<T> responseType)
                                    throws RestClientException
```

Description copied from interface: `RestOperations`

Create a new resource by POSTing the given object to the URL, and returns the response as `ResponseEntity`.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

`postForEntity` in interface `RestOperations`

Parameters:

`url` - the URL

request - the Object to be POSTed (may be null)

Returns:

the converted object

Throws:

RestClientException

See Also:

HttpEntity

put

```
public void put(String url,
                Object request,
                Object... uriVariables)
         throws RestClientException
```

Description copied from interface: RestOperations

Create or update a resource by PUTting the given object to the URI.

URI Template variables are expanded using the given URI variables, if any.

The request parameter can be a HttpEntity in order to add additional HTTP headers to the request.

Specified by:

put in interface RestOperations

Parameters:

url - the URL

request - the Object to be PUT (may be null)

uriVariables - the variables to expand the template

Throws:

RestClientException

See Also:

HttpEntity

put

```
public void put(String url,
                Object request,
                Map<String,?> uriVariables)
         throws RestClientException
```

Description copied from interface: RestOperations

Creates a new resource by PUTting the given object to URI template.

URI Template variables are expanded using the given map.

The request parameter can be a HttpEntity in order to add additional HTTP headers to the request.

Specified by:

put in interface RestOperations

Parameters:

url - the URL

request - the Object to be PUT (may be null)

uriVariables - the variables to expand the template

Throws:

`RestClientException`

See Also:

`HttpEntity`

put

```
public void put(URI url,
                Object request)
         throws RestClientException
```

Description copied from interface: `RestOperations`

Creates a new resource by PUTting the given object to URL.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

Specified by:

`put` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the Object to be PUT (may be null)

Throws:

`RestClientException`

See Also:

`HttpEntity`

patchForObject

```
public <T> T patchForObject(String url,
                            Object request,
                            Class<T> responseType,
                            Object... uriVariables)
                     throws RestClientException
```

Description copied from interface: `RestOperations`

Update a resource by PATCHing the given object to the URI template, and return the representation found in the response.

URI Template variables are expanded using the given URI variables, if any.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

NOTE: The standard JDK HTTP library does not support HTTP PATCH. You need to use the Apache HttpComponents or OkHttp request factory.

Specified by:

`patchForObject` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the object to be PATCHed (may be null)

`responseType` - the type of the return value

`uriVariables` - the variables to expand the template

Returns:

the converted object

Throws:

`RestClientException`

See Also:

`HttpEntity`,
`HttpAccessor.setRequestFactory(org.springframework.http.client.ClientHttpRequestFactory)`,
`HttpComponentsAsyncClientHttpRequestFactory`, `OkHttp3ClientHttpRequestFactory`

patchForObject

```
public <T> T patchForObject(String url,
                            Object request,
                            Class<T> responseType,
                            Map<String,?> uriVariables)
                     throws RestClientException
```

Description copied from interface: `RestOperations`
Update a resource by PATCHing the given object to the URI template, and return the representation found in the response.

URI Template variables are expanded using the given map.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

NOTE: The standard JDK HTTP library does not support HTTP PATCH. You need to use the Apache HttpComponents or OkHttp request factory.

Specified by:

`patchForObject` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the object to be PATCHed (may be null)

`responseType` - the type of the return value

`uriVariables` - the variables to expand the template

Returns:

the converted object

Throws:

`RestClientException`

See Also:

`HttpEntity`,
`HttpAccessor.setRequestFactory(org.springframework.http.client.ClientHttpRequestFactory)`,
`HttpComponentsAsyncClientHttpRequestFactory`, `OkHttp3ClientHttpRequestFactory`

patchForObject

```
public <T> T patchForObject(URI url,
                            Object request,
                            Class<T> responseType)
                     throws RestClientException
```

Description copied from interface: `RestOperations`
Update a resource by PATCHing the given object to the URL, and return the representation found in the response.

The `request` parameter can be a `HttpEntity` in order to add additional HTTP headers to the request.

NOTE: The standard JDK HTTP library does not support HTTP PATCH. You need to use the Apache HttpComponents or OkHttp request factory.

Specified by:

`patchForObject` in interface `RestOperations`

Parameters:

`url` - the URL

`request` - the object to be PATCHed (may be null)

`responseType` - the type of the return value

Returns:

the converted object

Throws:

`RestClientException`

See Also:

`HttpEntity`,
`HttpAccessor.setRequestFactory(org.springframework.http.client.ClientHttpRequestFactory)`,
`HttpComponentsAsyncClientHttpRequestFactory`, `OkHttp3ClientHttpRequestFactory`

delete

```
public void delete(String url,
                   Object... uriVariables)
            throws RestClientException
```

Description copied from interface: `RestOperations`

Delete the resources at the specified URI.

URI Template variables are expanded using the given URI variables, if any.

Specified by:

`delete` in interface `RestOperations`

Parameters:

`url` - the URL

`uriVariables` - the variables to expand in the template

Throws:

`RestClientException`

delete

```
public void delete(String url,
                   Map<String,?> uriVariables)
            throws RestClientException
```

Description copied from interface: `RestOperations`

Delete the resources at the specified URI.

URI Template variables are expanded using the given map.

Specified by:

`delete` in interface `RestOperations`

Parameters:

`url` - the URL

`uriVariables` - the variables to expand the template

Throws:

`RestClientException`

delete

```
public void delete(URI url)
            throws RestClientException
```

Description copied from interface: RestOperations
Delete the resources at the specified URL.

Specified by:
`delete` in interface `RestOperations`

Parameters:
`url` - the URL

Throws:
`RestClientException`

optionsForAllow

```
public Set<HttpMethod> optionsForAllow(String url,
                                       Object... uriVariables)
                                throws RestClientException
```

Description copied from interface: RestOperations
Return the value of the Allow header for the given URI.

URI Template variables are expanded using the given URI variables, if any.

Specified by:
`optionsForAllow` in interface `RestOperations`

Parameters:
`url` - the URL

`uriVariables` - the variables to expand in the template

Returns:
the value of the allow header

Throws:
`RestClientException`

optionsForAllow

```
public Set<HttpMethod> optionsForAllow(String url,
                                       Map<String,?> uriVariables)
                                throws RestClientException
```

Description copied from interface: RestOperations
Return the value of the Allow header for the given URI.

URI Template variables are expanded using the given map.

Specified by:
`optionsForAllow` in interface `RestOperations`

Parameters:
`url` - the URL

`uriVariables` - the variables to expand in the template

Returns:
the value of the allow header

Throws:

RestClientException

optionsForAllow

```
public Set<HttpMethod> optionsForAllow(URI url)
                                throws RestClientException
```

Description copied from interface: RestOperations

Return the value of the Allow header for the given URL.

Specified by:

optionsForAllow in interface RestOperations

Parameters:

url - the URL

Returns:

the value of the allow header

Throws:

RestClientException

exchange

```
public <T> ResponseEntity<T> exchange(String url,
                                      HttpMethod method,
                                      HttpEntity<?> requestEntity,
                                      Class<T> responseType,
                                      Object... uriVariables)
                               throws RestClientException
```

Description copied from interface: RestOperations

Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as ResponseEntity.

URI Template variables are expanded using the given URI variables, if any.

Specified by:

exchange in interface RestOperations

Parameters:

url - the URL

method - the HTTP method (GET, POST, etc)

requestEntity - the entity (headers and/or body) to write to the request may be null)

responseType - the type of the return value

uriVariables - the variables to expand in the template

Returns:

the response as entity

Throws:

RestClientException

exchange

```
public <T> ResponseEntity<T> exchange(String url,
                                      HttpMethod method,
                                      HttpEntity<?> requestEntity,
                                      Class<T> responseType,
                                      Map<String,?> uriVariables)
                               throws RestClientException
```

Description copied from interface: `RestOperations`
Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.

URI Template variables are expanded using the given URI variables, if any.

Specified by:
`exchange` in interface `RestOperations`

Parameters:
`url` - the URL

`method` - the HTTP method (GET, POST, etc)

`requestEntity` - the entity (headers and/or body) to write to the request (may be null)

`responseType` - the type of the return value

`uriVariables` - the variables to expand in the template

Returns:
the response as entity

Throws:
`RestClientException`

exchange

```
public <T> ResponseEntity<T> exchange(URI url,
                                      HttpMethod method,
                                      HttpEntity<?> requestEntity,
                                      Class<T> responseType)
                               throws RestClientException
```

Description copied from interface: `RestOperations`
Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as `ResponseEntity`.

Specified by:
`exchange` in interface `RestOperations`

Parameters:
`url` - the URL

`method` - the HTTP method (GET, POST, etc)

`requestEntity` - the entity (headers and/or body) to write to the request (may be null)

`responseType` - the type of the return value

Returns:
the response as entity

Throws:
`RestClientException`

exchange

```
public <T> ResponseEntity<T> exchange(String url,
                                      HttpMethod method,
                                      HttpEntity<?> requestEntity,
                                      ParameterizedTypeReference<T> responseType,
                                      Object... uriVariables)
                               throws RestClientException
```

Description copied from interface: RestOperations

Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as ResponseEntity. The given ParameterizedTypeReference is used to pass generic type information:

```
ParameterizedTypeReference<List<MyBean>> myBean = new ParameterizedTypeReference<List<MyBea
ResponseEntity<List<MyBean>> response = template.exchange("http://example.com",HttpMethod.G
```

Specified by:

exchange in interface RestOperations

Parameters:

url - the URL

method - the HTTP method (GET, POST, etc)

requestEntity - the entity (headers and/or body) to write to the request (may be null)

responseType - the type of the return value

uriVariables - the variables to expand in the template

Returns:

the response as entity

Throws:

RestClientException

exchange

```
public <T> ResponseEntity<T> exchange(String url,
                                 HttpMethod method,
                                 HttpEntity<?> requestEntity,
                                 ParameterizedTypeReference<T> responseType,
                                 Map<String,?> uriVariables)
                          throws RestClientException
```

Description copied from interface: RestOperations

Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as ResponseEntity. The given ParameterizedTypeReference is used to pass generic type information:

```
ParameterizedTypeReference<List<MyBean>> myBean = new ParameterizedTypeReference<List<MyBea
ResponseEntity<List<MyBean>> response = template.exchange("http://example.com",HttpMethod.G
```

Specified by:

exchange in interface RestOperations

Parameters:

url - the URL

method - the HTTP method (GET, POST, etc)

requestEntity - the entity (headers and/or body) to write to the request (may be null)

responseType - the type of the return value

uriVariables - the variables to expand in the template

Returns:

the response as entity

Throws:

RestClientException

exchange

```
public <T> ResponseEntity<T> exchange(URI url,
                                      HttpMethod method,
                                      HttpEntity<?> requestEntity,
                                      ParameterizedTypeReference<T> responseType)
                               throws RestClientException
```

Description copied from interface: RestOperations

Execute the HTTP method to the given URI template, writing the given request entity to the request, and returns the response as ResponseEntity. The given ParameterizedTypeReference is used to pass generic type information:

```
ParameterizedTypeReference<List<MyBean>> myBean = new ParameterizedTypeReference<List<MyBea
ResponseEntity<List<MyBean>> response = template.exchange("http://example.com",HttpMethod.G
```

Specified by:

exchange in interface RestOperations

Parameters:

url - the URL

method - the HTTP method (GET, POST, etc)

requestEntity - the entity (headers and/or body) to write to the request (may be null)

responseType - the type of the return value

Returns:

the response as entity

Throws:

RestClientException

exchange

```
public <T> ResponseEntity<T> exchange(RequestEntity<?> requestEntity,
                                      Class<T> responseType)
                               throws RestClientException
```

Description copied from interface: RestOperations

Execute the request specified in the given RequestEntity and return the response as ResponseEntity. Typically used in combination with the static builder methods on RequestEntity, for instance:

```
MyRequest body = ...
RequestEntity request = RequestEntity.post(new URI("http://example.com/foo")).accept(MediaT
ResponseEntity<MyResponse> response = template.exchange(request, MyResponse.class);
```

Specified by:

exchange in interface RestOperations

Parameters:

requestEntity - the entity to write to the request `responseType` - the type of the return value

Returns:

the response as entity

Throws:

`RestClientException`

exchange

```
public <T> ResponseEntity<T> exchange(RequestEntity<?> requestEntity,
                                      ParameterizedTypeReference<T> responseType)
                               throws RestClientException
```

Description copied from interface: `RestOperations`

Execute the request specified in the given `RequestEntity` and return the response as `ResponseEntity`. The given `ParameterizedTypeReference` is used to pass generic type information:

```
MyRequest body = ...
RequestEntity request = RequestEntity.post(new URI("http://example.com/foo")).accept(MediaT
ParameterizedTypeReference<List<MyResponse>> myBean = new ParameterizedTypeReference<List<M
ResponseEntity<List<MyResponse>> response = template.exchange(request, myBean);
```

Specified by:

`exchange` in interface `RestOperations`

Parameters:

`requestEntity` - the entity to write to the request

`responseType` - the type of the return value

Returns:

the response as entity

Throws:

`RestClientException`

execute

```
public <T> T execute(String url,
                     HttpMethod method,
                     RequestCallback requestCallback,
                     ResponseExtractor<T> responseExtractor,
                     Object... uriVariables)
              throws RestClientException
```

Description copied from interface: `RestOperations`

Execute the HTTP method to the given URI template, preparing the request with the `RequestCallback`, and reading the response with a `ResponseExtractor`.

URI Template variables are expanded using the given URI variables, if any.

Specified by:

`execute` in interface `RestOperations`

Parameters:

`url` - the URL

`method` - the HTTP method (GET, POST, etc)

`requestCallback` - object that prepares the request `responseExtractor` - object that extracts the return value from the response

`uriVariables` - the variables to expand in the template

Returns:

an arbitrary object, as returned by the `ResponseExtractor`

Throws:

`RestClientException`

execute

```
public <T> T execute(String url,
                     HttpMethod method,
                     RequestCallback requestCallback,
                     ResponseExtractor<T> responseExtractor,
                     Map<String,?> uriVariables)
              throws RestClientException
```

Description copied from interface: `RestOperations`

Execute the HTTP method to the given URI template, preparing the request with the `RequestCallback`, and reading the response with a `ResponseExtractor`.

URI Template variables are expanded using the given URI variables map.

Specified by:

`execute` in interface `RestOperations`

Parameters:

`url` - the URL

`method` - the HTTP method (GET, POST, etc)

`requestCallback` - object that prepares the request

`responseExtractor` - object that extracts the return value from the response

`uriVariables` - the variables to expand in the template

Returns:

an arbitrary object, as returned by the `ResponseExtractor`

Throws:

`RestClientException`

execute

```
public <T> T execute(URI url,
                     HttpMethod method,
                     RequestCallback requestCallback,
                     ResponseExtractor<T> responseExtractor)
              throws RestClientException
```

Description copied from interface: `RestOperations`

Execute the HTTP method to the given URL, preparing the request with the `RequestCallback`, and reading the response with a `ResponseExtractor`.

Specified by:

`execute` in interface `RestOperations`

Parameters:

`url` - the URL

`method` - the HTTP method (GET, POST, etc)

`requestCallback` - object that prepares the request `responseExtractor` - object that extracts the return value from the response

Returns:

an arbitrary object, as returned by the `ResponseExtractor`

Throws:

`RestClientException`

doExecute

```
protected <T> T doExecute(URI url,
                          HttpMethod method,
                          RequestCallback requestCallback,
                          ResponseExtractor<T> responseExtractor)
                   throws RestClientException
```

Execute the given method on the provided URI.

The `ClientHttpRequest` is processed using the `RequestCallback`; the response with the `ResponseExtractor`.

Parameters:

`url` - the fully-expanded URL to connect to

`method` - the HTTP method to execute (GET, POST, etc.)

`requestCallback` - object that prepares the request (can be null)

`responseExtractor` - object that extracts the return value from the response (can be null)

Returns:

an arbitrary object, as returned by the `ResponseExtractor`

Throws:

`RestClientException`

handleResponse

```
protected void handleResponse(URI url,
                              HttpMethod method,
                              ClientHttpResponse response)
                       throws IOException
```

Handle the given response, performing appropriate logging and invoking the `ResponseErrorHandler` if necessary.

Can be overridden in subclasses.

Parameters:

`url` - the fully-expanded URL to connect to

`method` - the HTTP method to execute (GET, POST, etc.)

`response` - the resulting `ClientHttpResponse`

Throws:

`IOException` - if propagated from `ResponseErrorHandler`

Since:

4.1.6

See Also:

`setErrorHandler(org.springframework.web.client.ResponseErrorHandler)`

acceptHeaderRequestCallback

```
protected <T> RequestCallback acceptHeaderRequestCallback(Class<T> responseType)
```

Returns a request callback implementation that prepares the request `Accept` headers based on the given response type and configured message converters.

httpEntityCallback

```
protected <T> RequestCallback httpEntityCallback(Object requestBody)
```

Returns a request callback implementation that writes the given object to the request stream.

httpEntityCallback

```
protected <T> RequestCallback httpEntityCallback(Object requestBody,
                                                 Type responseType)
```

Returns a request callback implementation that writes the given object to the request stream.

responseEntityExtractor

```
protected <T> ResponseExtractor<ResponseEntity<T>> responseEntityExtractor(Type responseType)
```

Returns a response extractor for `ResponseEntity`.

headersExtractor

```
protected ResponseExtractor<HttpHeaders> headersExtractor()
```

Returns a response extractor for `HttpHeaders`.