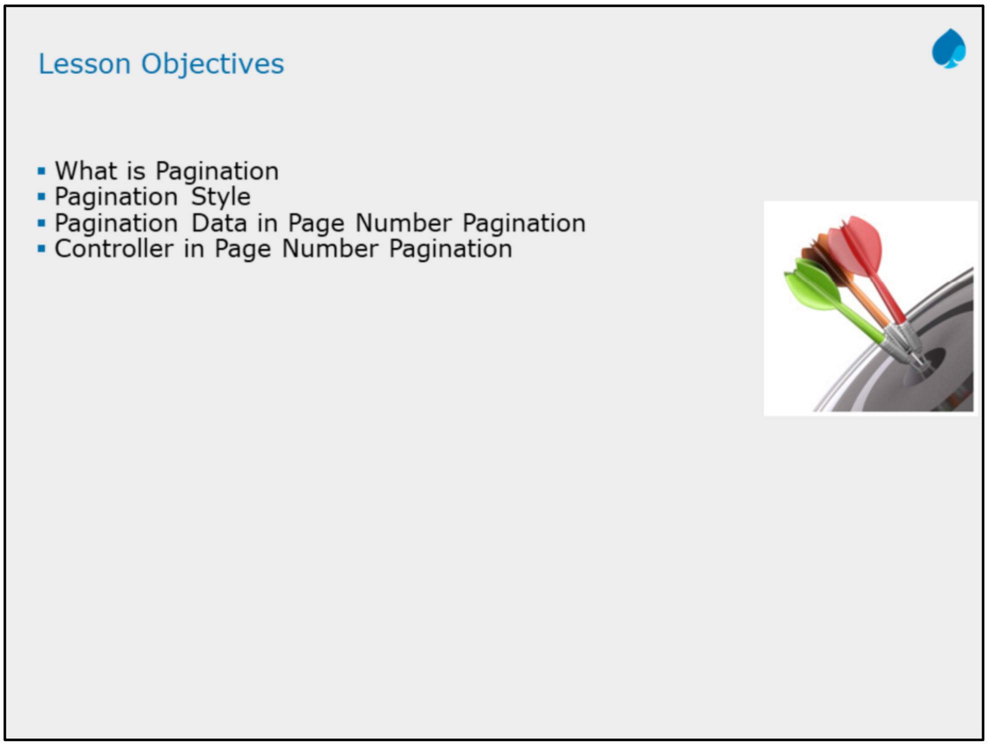
Instructor Notes:

Add instructor notes here.

RESTFul Web Services with Spring

Paging

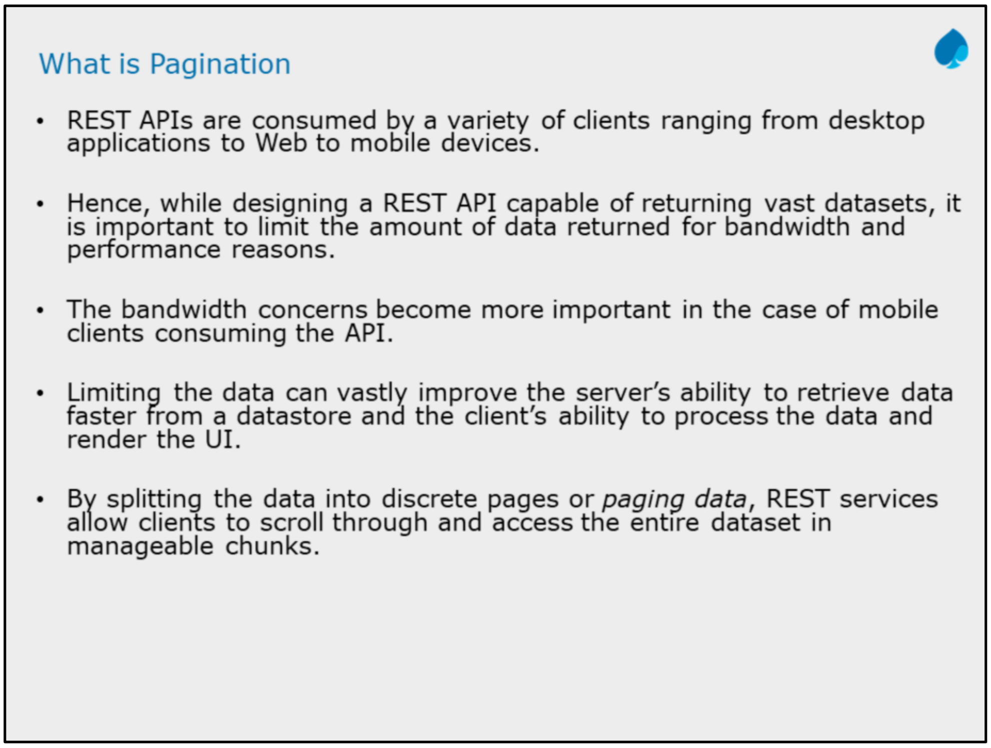
Instructor Notes:



Lesson Objectives

- What is Pagination
- Pagination Style
- Pagination Data in Page Number Pagination
- Controller in Page Number Pagination

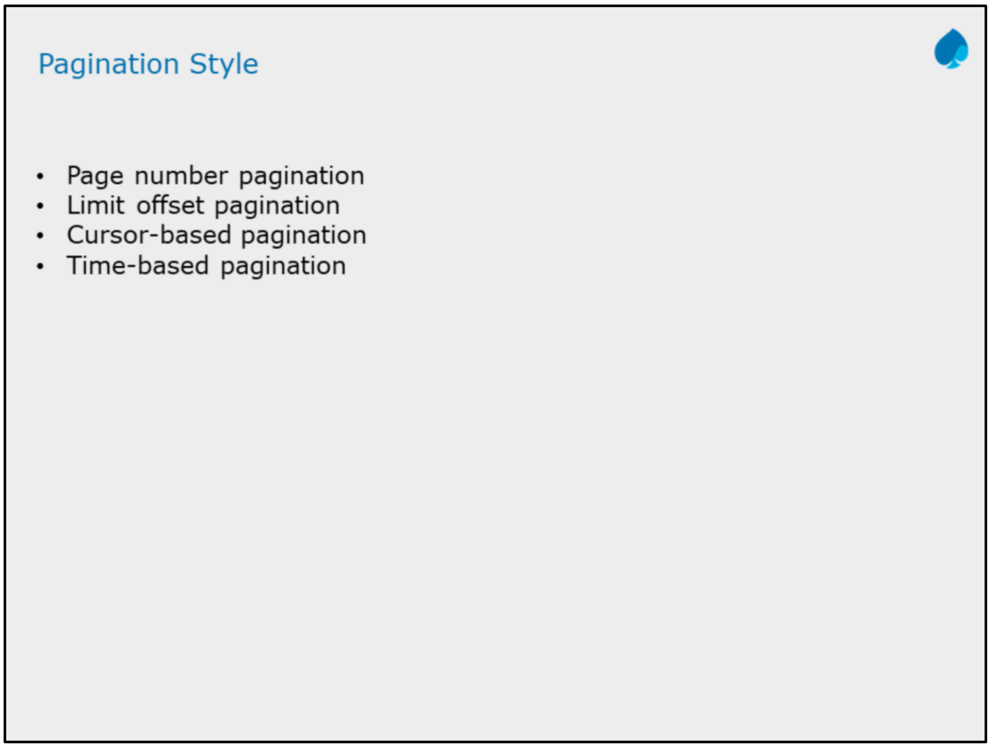
Instructor Notes:



What is Pagination

- REST APIs are consumed by a variety of clients ranging from desktop applications to Web to mobile devices.
- Hence, while designing a REST API capable of returning vast datasets, it is important to limit the amount of data returned for bandwidth and performance reasons.
- The bandwidth concerns become more important in the case of mobile clients consuming the API.
- Limiting the data can vastly improve the server's ability to retrieve data faster from a datastore and the client's ability to process the data and render the UI.
- By splitting the data into discrete pages or *paging data*, REST services allow clients to scroll through and access the entire dataset in manageable chunks.

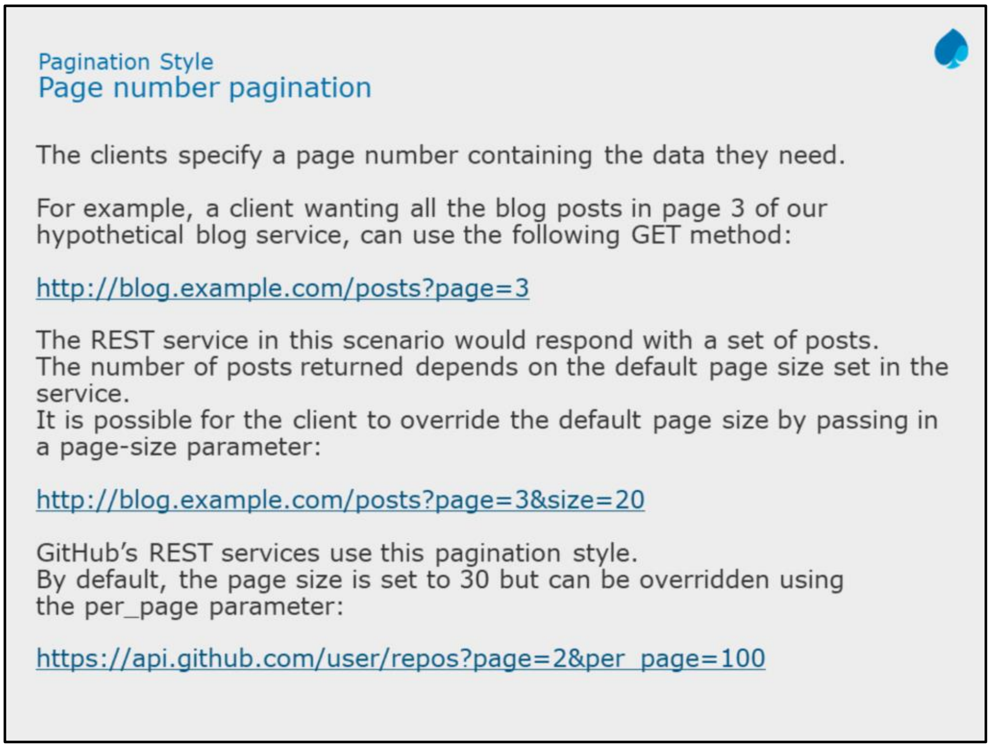
Instructor Notes:



Pagination Style

- Page number pagination
- Limit offset pagination
- Cursor-based pagination
- Time-based pagination

Instructor Notes:



Pagination Style

Page number pagination

The clients specify a page number containing the data they need.

For example, a client wanting all the blog posts in page 3 of our hypothetical blog service, can use the following GET method:

http://blog.example.com/posts?page=3

The REST service in this scenario would respond with a set of posts.
The number of posts returned depends on the default page size set in the service.
It is possible for the client to override the default page size by passing in a page-size parameter:

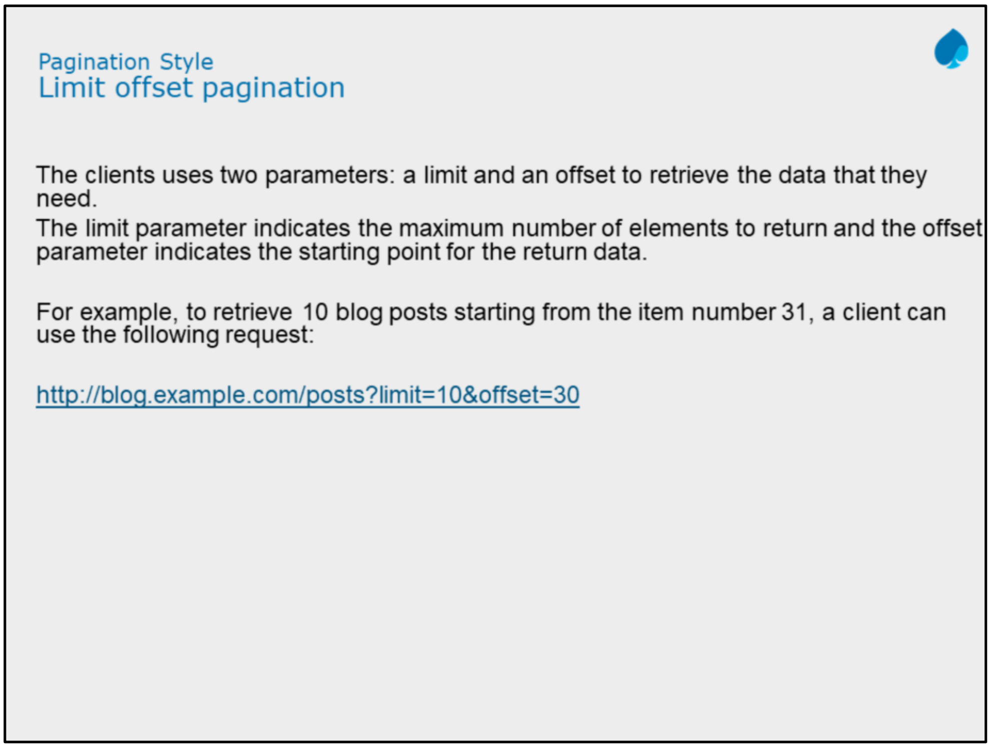
http://blog.example.com/posts?page=3&size=20

GitHub's REST services use this pagination style.
By default, the page size is set to 30 but can be overridden using the per_page parameter:

https://api.github.com/user/repos?page=2&per_page=100

In this pagination style, the clients specify a page number containing the data they need. For example, a client wanting all the blog posts in page 3 of our hypothetical blog service, can use the following GET method:
`http://blog.example.com/posts?page=3`
The REST service in this scenario would respond with a set of posts. The number of posts returned depends on the default page size set in the service. It is possible for the client to override the default page size by passing in a page-size parameter:
`http://blog.example.com/posts?page=3&size=20`
GitHub's REST services use this pagination style. By default, the page size is set to 30 but can be overridden using the `per_page` parameter:
`https://api.github.com/user/repos?page=2&per_page=100`

Instructor Notes:



Pagination Style
Limit offset pagination

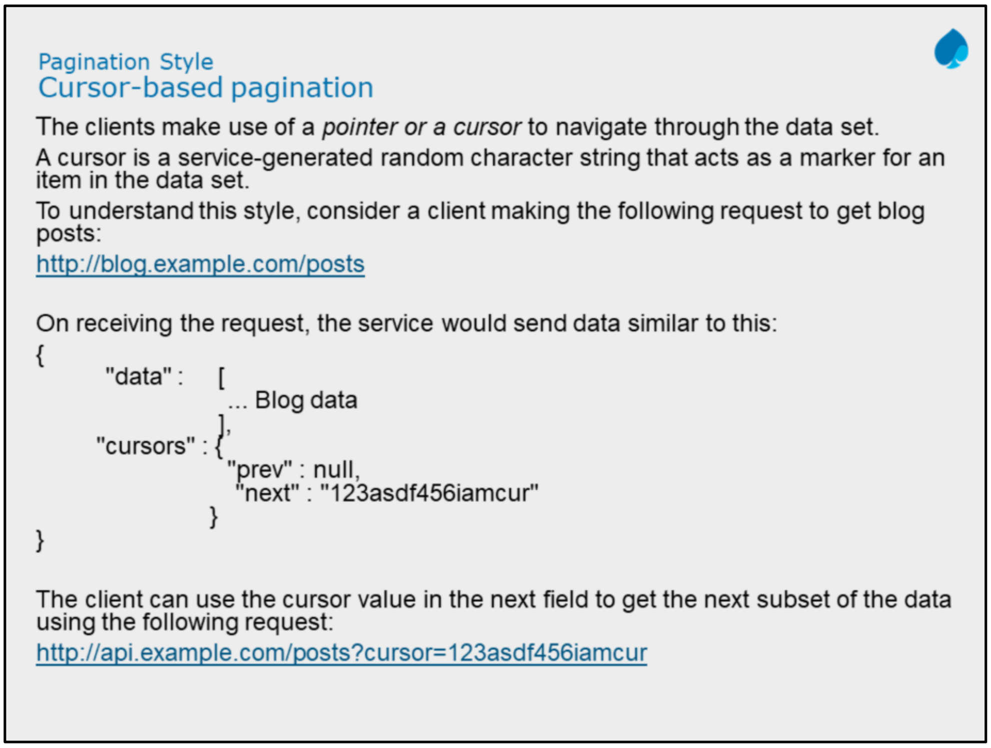
The clients uses two parameters: a limit and an offset to retrieve the data that they need.
The limit parameter indicates the maximum number of elements to return and the offset parameter indicates the starting point for the return data.

For example, to retrieve 10 blog posts starting from the item number 31, a client can use the following request:

http://blog.example.com/posts?limit=10&offset=30

In this pagination style, the clients uses two parameters: a limit and an offset to retrieve the data that they need. The limit parameter indicates the maximum number of elements to return and the offset parameter indicates the starting point for the return data. For example, to retrieve 10 blog posts starting from the item number 31, a client can use the following request: http://blog.example.com/posts?limit=10&offset=30

Instructor Notes:



Pagination Style

Cursor-based pagination

The clients make use of a *pointer or a cursor* to navigate through the data set.

A cursor is a service-generated random character string that acts as a marker for an item in the data set.

To understand this style, consider a client making the following request to get blog posts:

http://blog.example.com/posts

On receiving the request, the service would send data similar to this:

```
{
        "data" :    [
                     ... Blog data
                    ],
       "cursors" : {
                     "prev" : null,
                      "next" : "123asdf456iamcur"
                   }
}
```

The client can use the cursor value in the next field to get the next subset of the data using the following request:

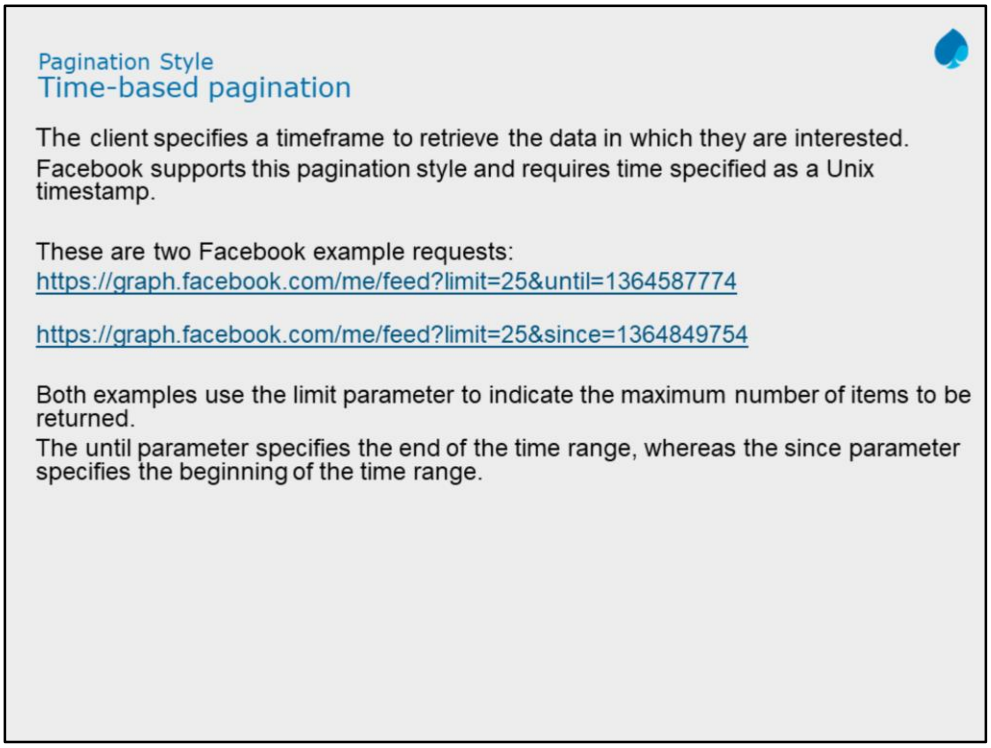
http://api.example.com/posts?cursor=123asdf456iamcur

In this pagination style, the clients make use of a *pointer or a cursor* to navigate through the data set. A cursor is a service-generated random character string that acts as a marker for an item in the data set. To understand this style, consider a client making the following request to get blog posts:
http://blog.example.com/posts
On receiving the request, the service would send data similar to this:

```
{
      "data" :    [
                ... Blog data
                      ],
           "cursors" : {
                  "prev" : null,
                  "next" : "123asdf456iamcur"
                    }
}
```

This response contains a set of blogs representing a subset of the total dataset. The cursors that are part of the response contains a prev field that can be used to retrieve the previous subset of the data. However, because this is the initial subset, the prev field value is empty. The client can use the cursor value in the next field to get the next subset of the data using the following request:
http://api.example.com/posts?cursor=123asdf456iamcur
On receiving this request, the service would send the data along with the prev and next cursor fields. This pagination style is used by applications such as Twitter and Facebook that deal with real-time datasets (tweets and posts) where data changes frequently. The generated cursors typically don't live forever and should be used for short-term pagination purposes only.

Instructor Notes:



Pagination Style

Time-based pagination

The client specifies a timeframe to retrieve the data in which they are interested.
Facebook supports this pagination style and requires time specified as a Unix timestamp.

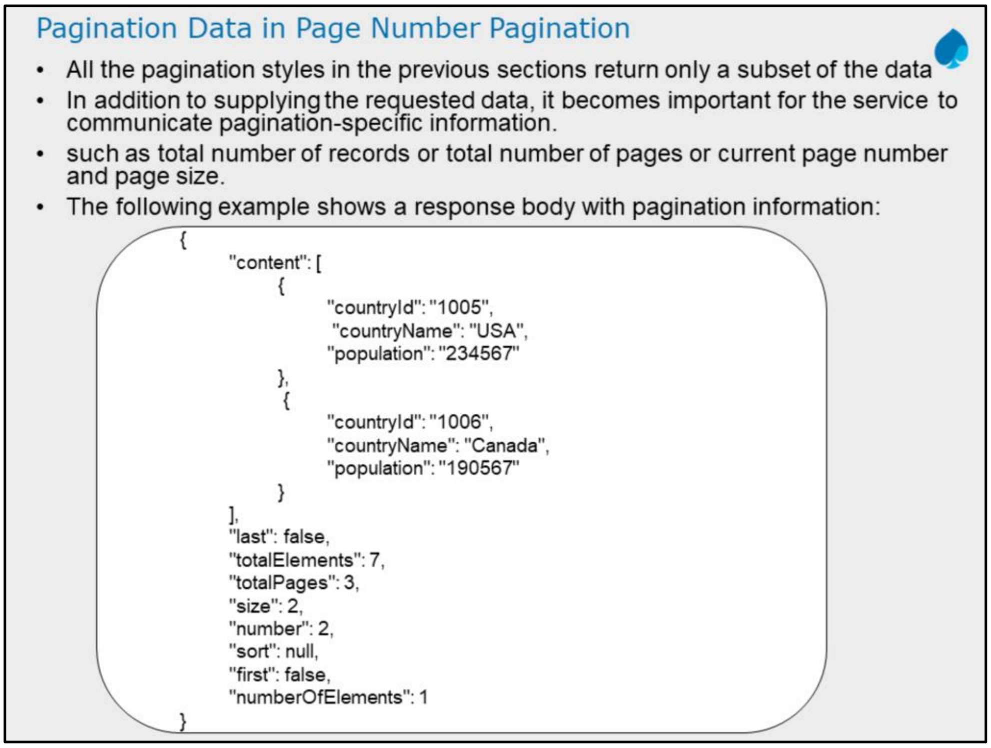
These are two Facebook example requests:
https://graph.facebook.com/me/feed?limit=25&until=1364587774

https://graph.facebook.com/me/feed?limit=25&since=1364849754

Both examples use the limit parameter to indicate the maximum number of items to be returned.
The until parameter specifies the end of the time range, whereas the since parameter specifies the beginning of the time range.

In this style of pagination, the client specifies a timeframe to retrieve the data in which they are interested. Facebook supports this pagination style and requires time specified as a Unix timestamp. These are two Facebook example requests:
https://graph.facebook.com/me/feed?limit=25&until=1364587774
https://graph.facebook.com/me/feed?limit=25&since=1364849754Both examples use the limit parameter to indicate the maximum number of items to be returned. The until parameter specifies the end of the time range, whereas the since parameter specifies the beginning of the time range.

Instructor Notes:



Pagination Data in Page Number Pagination

- All the pagination styles in the previous sections return only a subset of the data
- In addition to supplying the requested data, it becomes important for the service to communicate pagination-specific information.
- such as total number of records or total number of pages or current page number and page size.
- The following example shows a response body with pagination information:

```
{
        "content": [
              {
                      "countryId": "1005",
                       "countryName": "USA",
                      "population": "234567"
              },
               {
                      "countryId": "1006",
                      "countryName": "Canada",
                      "population": "190567"
              }
        ],
        "last": false,
        "totalElements": 7,
        "totalPages": 3,
        "size": 2,
        "number": 2,
        "sort": null,
        "first": false,
        "numberOfElements": 1
}
```

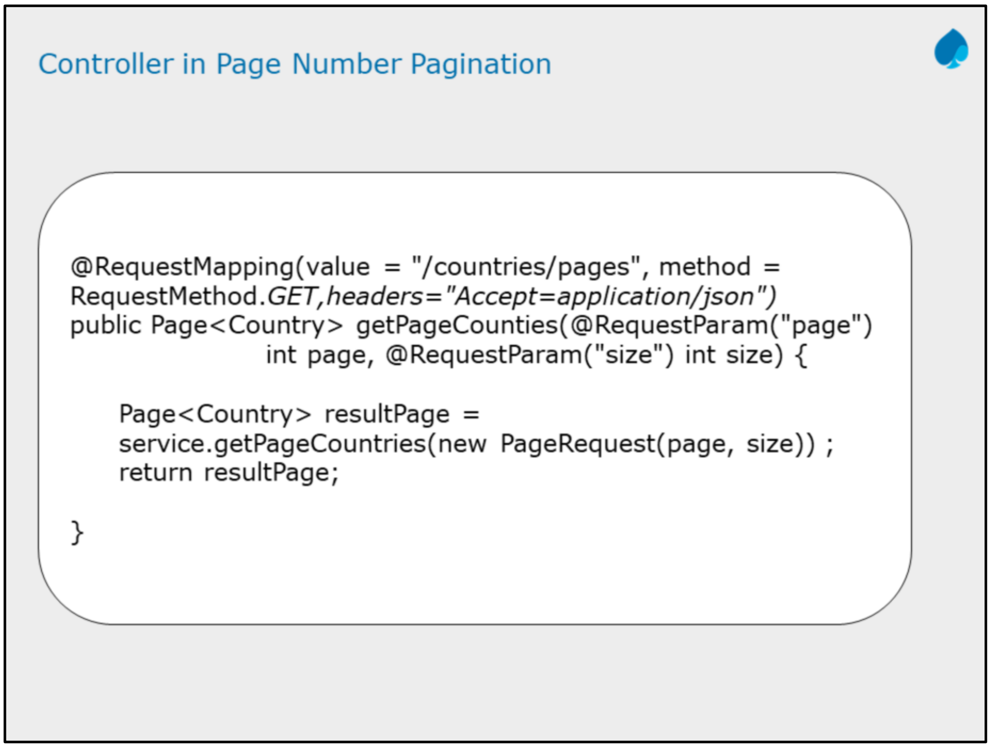
All the pagination styles in the previous sections return only a subset of the data. So, in addition to supplying the requested data, it becomes important for the service to communicate pagination-specific information such as total number of records or total number of pages or current page number and page size. The following example shows a response body with pagination information:

```
{
   "data": [
       ... Blog Data
   ],
   "totalPages": 9,
   "currentPageNumber": 2,
   "pageSize": 10,
   "totalRecords": 90
}
```

Clients can use the pagination information to assess the current state as well as construct URLs to obtain the next or previous datasets. The other technique services employ is to include the pagination information in a special Link header. The Link header is defined as part of RFC 5988(http://tools.ietf.org/html/rfc5988). It typically contains a set of ready-made links to scroll forward and backward. GitHub uses this approach; here is an example of a Link header value:

Link: <https://api.github.com/user/repos?page=3&per_page=100>; rel="next", <https://api.github.com/user/repos?page=50&per_page=100>; rel="last"

Instructor Notes:



Controller in Page Number Pagination

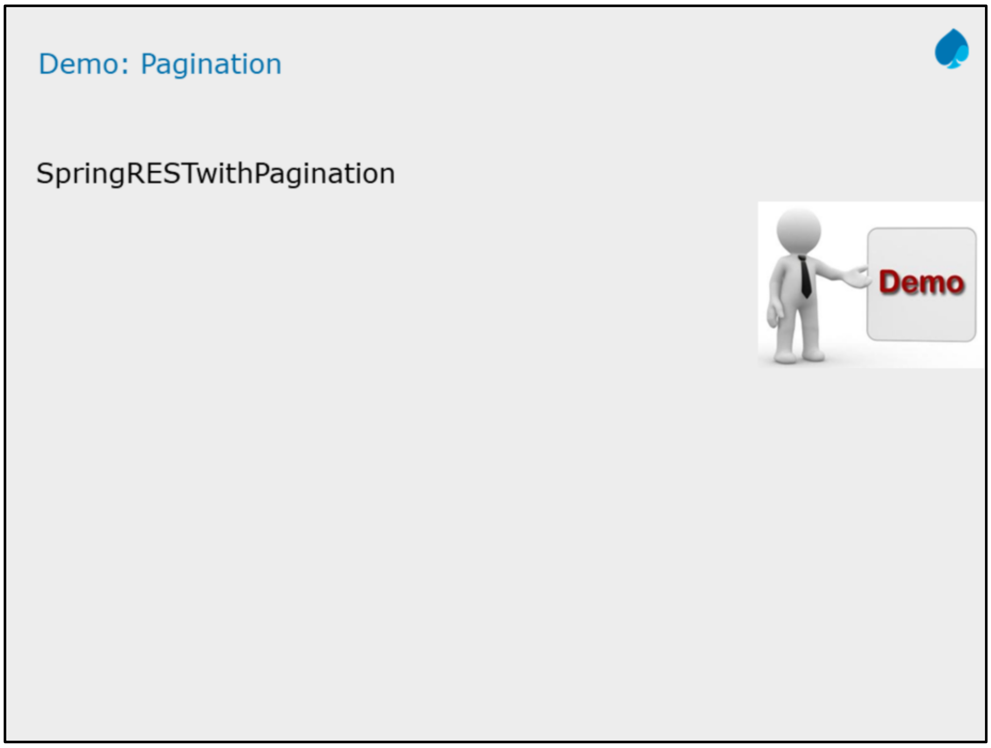
```
@RequestMapping(value = "/countries/pages", method =
RequestMethod.GET,headers="Accept=application/json")
public Page<Country> getPageCounties(@RequestParam("page")
                int page, @RequestParam("size") int size) {

    Page<Country> resultPage =
    service.getPageCountries(new PageRequest(page, size)) ;
    return resultPage;

}
```

Instructor Notes:

These demos can be executed for better understanding



Demo: Pagination

SpringRESTwithPagination

Get the output by below url in Postman:

http://localhost:8081/SpringRESTwithPagination/rest/countries/pages?page=0&size=3
http://localhost:8081/SpringRESTwithPagination/rest/countries/pages?page=1&size=3
http://localhost:8081/SpringRESTwithPagination/rest/countries/pages?page=2&size=3

Instructor Notes:

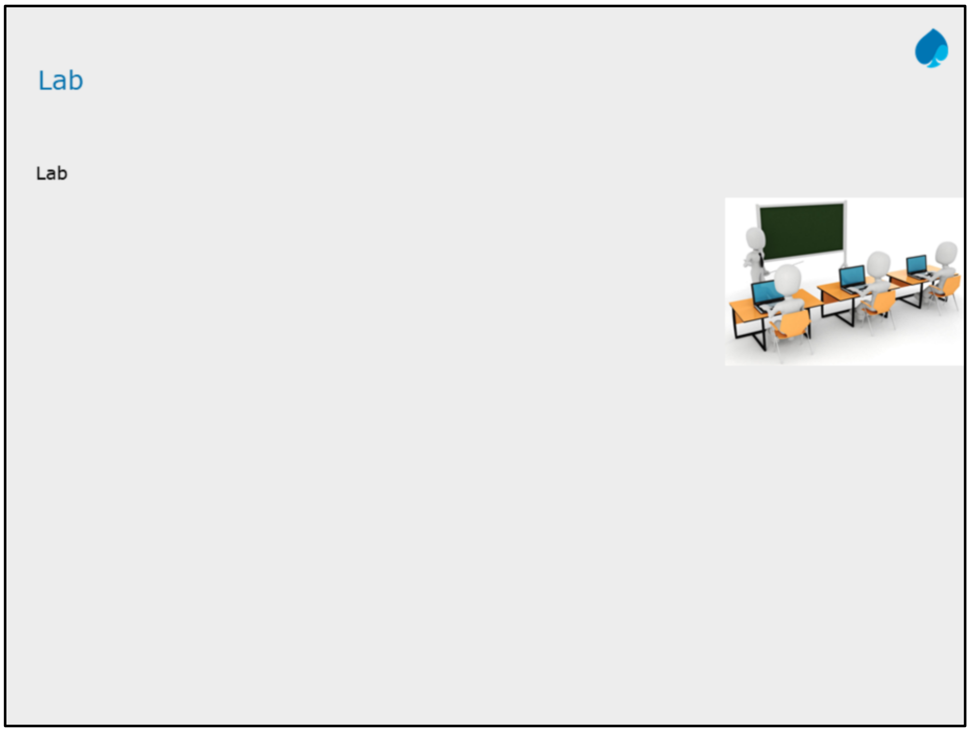
Lesson Objectives

- Introduction to REST web Services
- REST Controllers on the top of MVC

References:
https://www.genuitec.com/spring-frameworkrestcontroller-vs-controller/

Instructor Notes:

Corresponding lab assignment



Lab

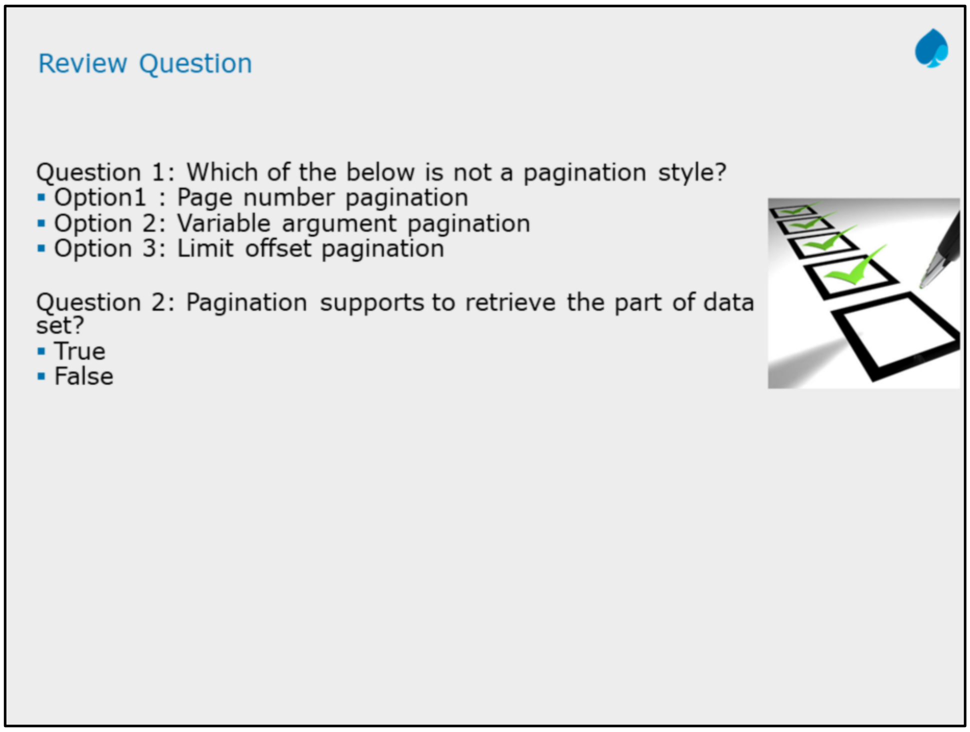
Lab

Lab will be added after lab book creation

Instructor Notes:

Question 1: Option 2

Question 2: True



Review Question

Question 1: Which of the below is not a pagination style?
- Option1 : Page number pagination
- Option 2: Variable argument pagination
- Option 3: Limit offset pagination

Question 2: Pagination supports to retrieve the part of data set?
- True
- False

Add the notes here.