Server-Side Scrolling and Sorting

This chapter covers the following topics:

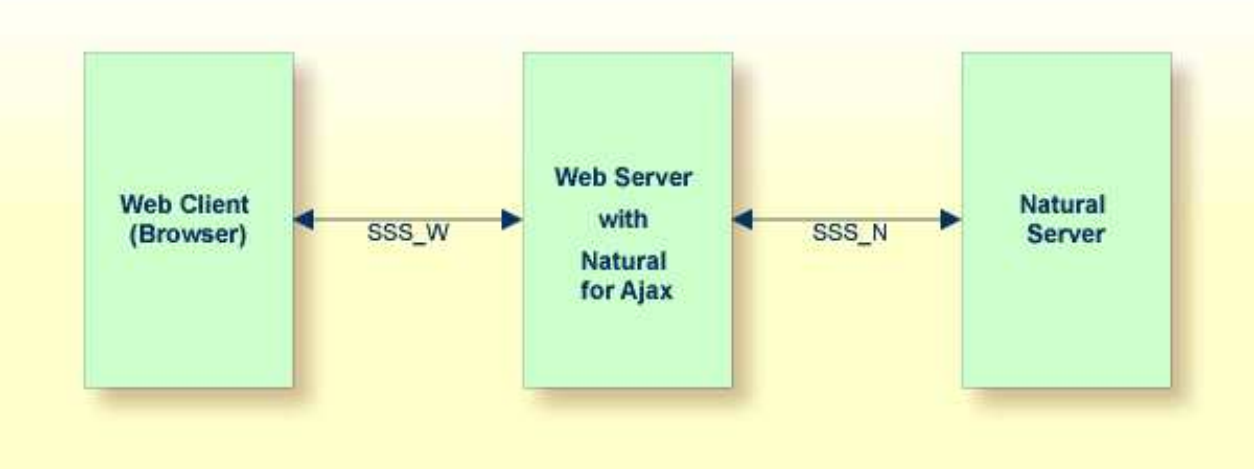
- General Information
- Variants of Server-Side Scrolling and Sorting
- Controls that Support Server-Side Scrolling and Sorting
- Data Structures for Server-Side Scrolling and Sorting
- Server-Side Scrolling and Sorting in Trees
- Events for Server-Side Scrolling and Sorting

General Information

It is often the case that a web application has to display an arbitrary amount of data in a grid control, for instance, the records from a database table. In these cases, it is mostly not efficient to send all data as a whole to the web client. Instead, it will be intended to display a certain amount of data to begin with and to send more data as the user scrolls through the page. To support this, the grid controls in Natural for Ajax support the concept of server-side scrolling and sorting.

Variants of Server-Side Scrolling and Sorting

The following graphic illustrates the different types of server-side scrolling and sorting that are supported by Natural for Ajax.



With respect to server-side scrolling and sorting, the following options can be used:

- **No Server-Side Scrolling and Sorting**
  The Natural application sends the grid data to the web server as a whole. The web server sends the grid data to the web client (browser) as a whole.

Advantage: Neither the web server nor the Natural application are involved in the process of scrolling and sorting. As long as the user only scrolls and sorts, no round trip from the web client to the web server or to the Natural server is necessary.

Disadvantage: A round trip between web server and Natural server that is triggered by other user actions transports the entire grid data.

- **Web Server-Side Scrolling and Sorting (SSS_W)**
  The Natural application sends the grid data to the web server as a whole. The web server sends the grid data to the web client (browser) in portions.

  Advantage: The Natural application is not involved in the process of scrolling and sorting. As long as the user only scrolls and sorts, no round trip from the web server to the Natural server is necessary.

  Disadvantage: A round trip between web server and Natural server that is triggered by other user actions transports the entire grid data.

- **Natural Server-Side Scrolling and Sorting (SSS_N)**
  The Natural application sends the grid data to the web server in portions. The web server sends the grid data to the web client (browser) in portions.

  Advantage: A round trip between web server and Natural application passes only the visible data portion.

  Disadvantage: The Natural application must support the process of scrolling and sorting with a specific application logic.

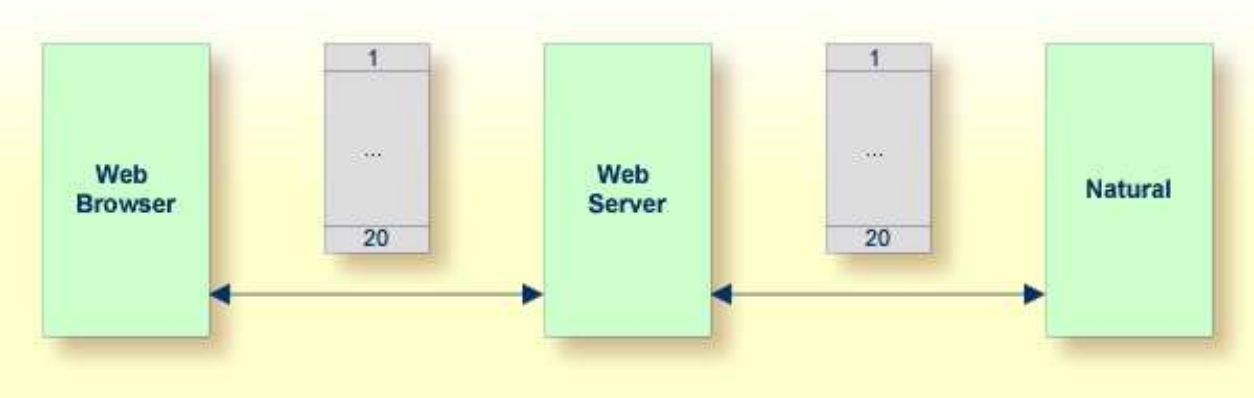
The decision between these options will often depend on the expected data volume. The application can decide dynamically at runtime which option to use.

The following topics show the difference between these three options

- No Server-Side Scrolling and Sorting
- Web Server-Side Scrolling and Sorting
- Natural Server-Side Scrolling and Sorting

No Server-Side Scrolling and Sorting

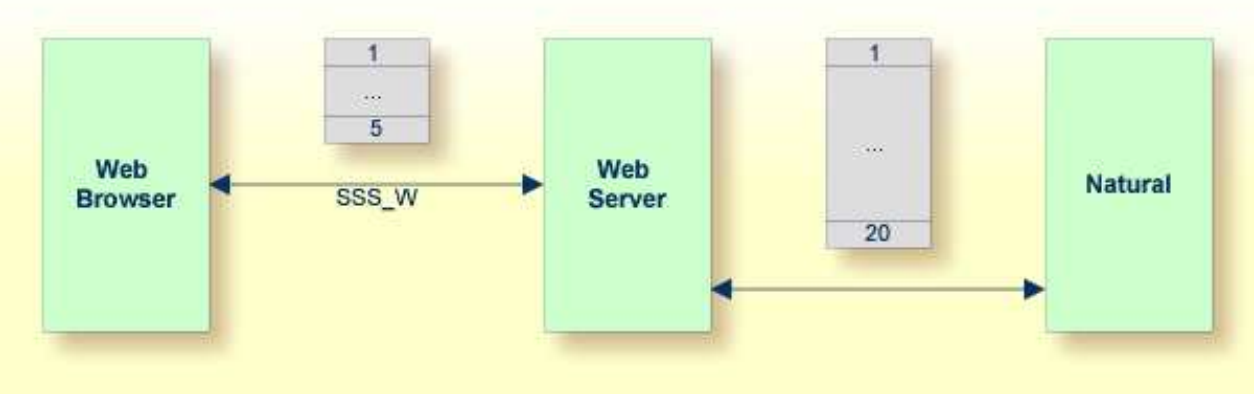
Step 1: The grid is configured at design time to a row count of twenty. The Natural application sends twenty rows and indicates that no further rows are to be expected (`SIZE=0`).



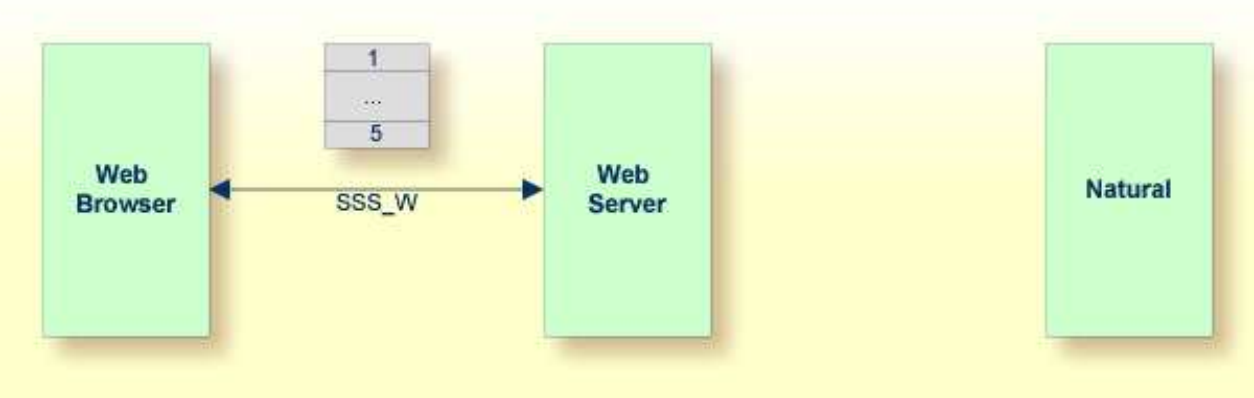
Step 2: When you scroll up and down, no server round trips to the web server or to the Natural application are performed.

Web Server-Side Scrolling and Sorting

Step 1: The grid is configured at design time to a row count of five. The Natural application sends twenty rows and indicates that no further rows are to be expected (`SIZE=0`).

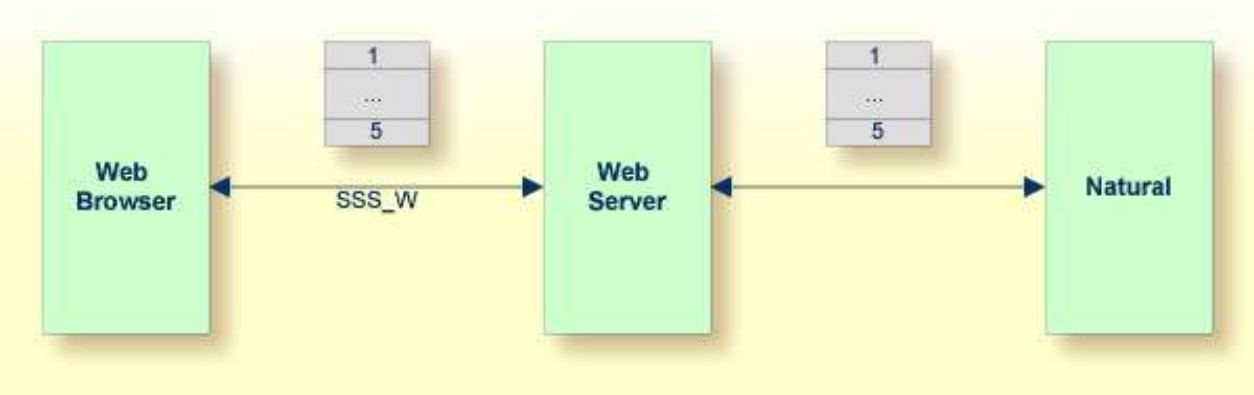


Step 2: When you scroll up and down, the web browser requests additional records from the web server There are no server round trips to Natural.

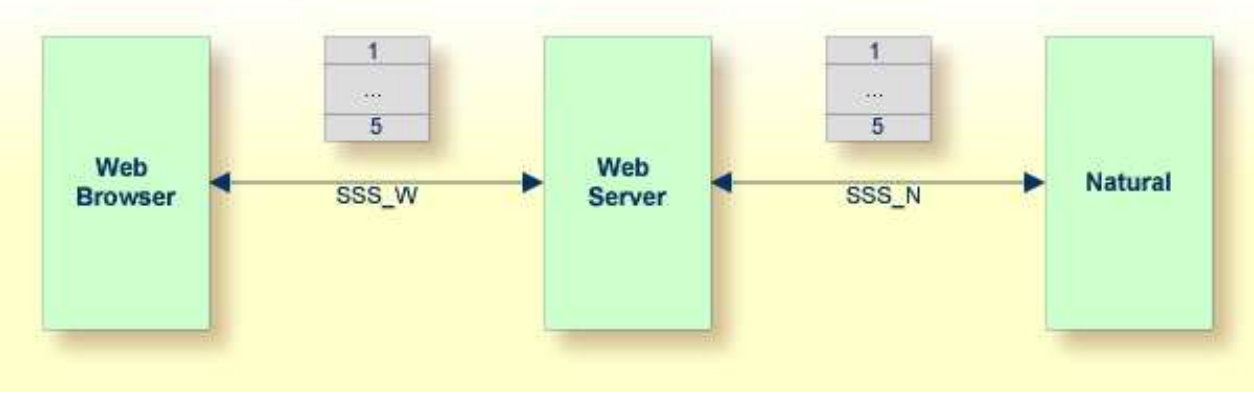


Natural Server-Side Scrolling and Sorting

Step 1: The grid is configured at design time to a row count of five. The Natural application sends five rows and indicates that further rows are to be expected (`SIZE=20`).



Step 2: When you scroll up and down, the web browser requests additional records from the web server. The web server requests additional records from the Natural application.



The Natural application can dynamically decide at runtime which option of server-side scrolling and sorting it wants to use. This can depend on the number of records contained in a search result.

- If the application does not want to use server-side scrolling and sorting at all, it sends as many rows to the web browser as the grid is configured to hold, or it sends fewer rows.
- If the application wants to use web server-side scrolling and sorting, it sends all available rows and sets the `SIZE` parameter to zero in the data structure that represents the grid in the application.
- If the application wants to use Natural server-side scrolling and sorting, it sends only part of the available rows and indicates in the `SIZE` parameter how many rows are to be expected altogether.

Controls that Support Server-Side Scrolling and Sorting

The following controls support server-side scrolling and sorting:

- TEXTGRIDSSS2
- ROWTABLEAREA2
- MGDGRID

Note:
For compatibility reasons with earlier versions of Natural for Ajax, you have to set the `natsss` property of NATPAGE to true in order to activate server-side scrolling and sorting for the controls ROWTABLEAREA2 and MGDGRID. If this property is set to true, for all instances of these grid controls on a page, the necessary data structures are generated into the Natural adapter interface.

Data Structures for Server-Side Scrolling and Sorting

If you use the TEXTGRIDSSS2 control or if you use the ROWTABLEAREA2 or MGDGRID control and have set the property `natsss` to true for the page, the following additional data structure is generated into the adapter interface for each instance of these controls. This data structure is used to control the scroll and sort behavior at runtime.

```
1 LINESINFO
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (U) DYNAMIC
2 TOPINDEX (I4)
```

The name of the data structure is derived from the name of the variable that is bound to the grid. In this example, the variable `LINES` had been bound to the grid. Therefore, the name `LINESINFO` was generated.

With each event that is related to scrolling and sorting, the application receives the information how many rows it should deliver at least (`ROWCOUNT`) and the index of the first record to be delivered (`TOPINDEX`).

In `SORTPROPS`, the application receives the information in which sort sequence the records should be delivered and by which columns the records should be sorted.

On the other hand, the application itself can specify a sort sequence (also using multiple sort criteria) and indicate this sort sequence by filling the structure with the desired sort criteria.

- If web server-side scrolling and sorting is used, the specified sort sequence is automatically created on the web server.
- If Natural server-side scrolling and sorting is used, the application itself must provide the records in the specified sort sequence.
- With the TEXTGRIDSSS2 control, the first three specified sort criteria are automatically indicated in the column headers of the grid.
- With the ROWTABLEAREA2 control, the first specified sort criterion is automatically indicated in the column headers of the grid. If more sort criteria are to be be indicated, the application should provide custom grid headers.

In `SIZE`, the application can indicate whether the delivered amount of rows represents all available data (`SIZE=0`, no Natural server-side scrolling), or whether there are more rows to come (`SIZE=`*`total-number-of-records`*, Natural server-side scrolling).

When Natural server-side scrolling is used, the application will, for instance, hold the available rows (mostly the result of a database search) in an X-array, sort this X-array as requested and deliver the requested portion of rows. However, other implementations and optimizations are possible, depending on the needs and possibilities of the application.

Server-Side Scrolling and Sorting in Trees

The ROWTABLEAREA2 control can also be configured as a tree control, where each row represents a tree node. In this case, the data structure that supports server-side scrolling contains one more field, `DSPINDEXFIRST`.

```
1 LINESINFO
2 DSPINDEXFIRST (I4)
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (U) DYNAMIC
2 TOPINDEX (I4)
```

The need for this additional control field comes from the fact that a tree can contain hidden items.

The rows sent by the Natural application must always start with an item at level one. The additional field `DSPINDEXFIRST` is provided because the visible part of the tree can start at a node with a level greater than one (a subnode). In `DSPINDEXFIRST`, the application must indicate the index of the first visible row within the rows sent from Natural.

Example

The top nodes of the tree are open and the user scrolls down as shown below:

The Natural application is supposed to send data starting with a top node. In our example, this is the node named **toptext_0**. But the first visible child node would be **childtext_0.2**. This means that among the sent items, the first three items are hidden. The application sets the value for `DSPINDEXFIRST` to "3" when sending the data.

Events for Server-Side Scrolling and Sorting

In order to support server-side scrolling and sorting, an application must handle a number of related events properly. The events are described with the corresponding controls. Examples on how to handle the events are provided in the library `SYSEXNJX`.