



# Sequential attributes

## Pattern #4

A common requirement in enterprise applications is generating sequential codes that may need to follow certain composition rules. Although we could be tempted to use the auto-number feature of an entity's attribute, these numbers are not guaranteed to be sequential, only unique.

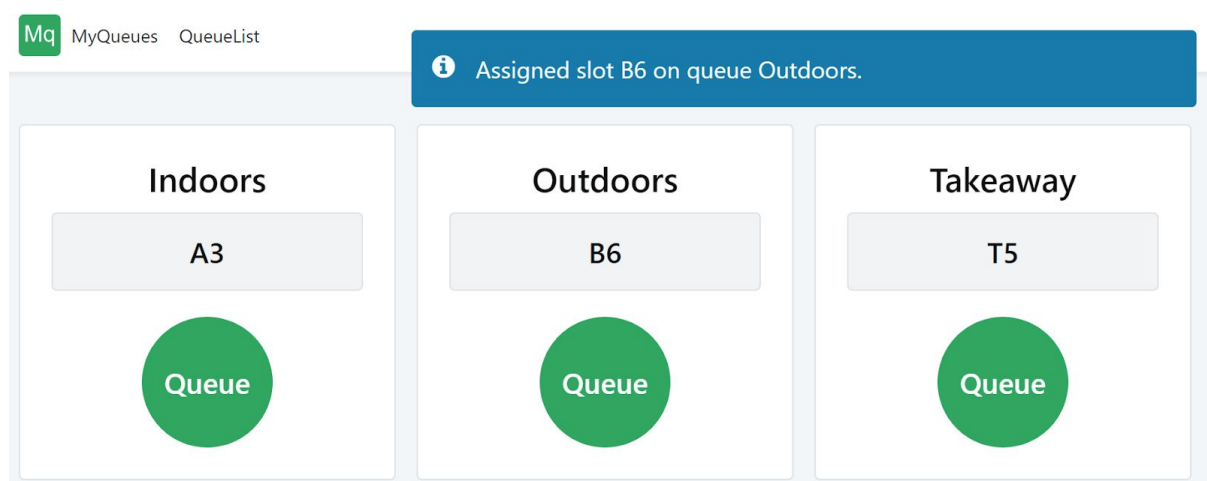
For this exercise we will create a new application that allows users to generate queue numbers and automatically assigns their number. First we will set up a sequential number mechanism for queues, afterwards we extend the queues numbers so the sequence is independent per queue.

**KEYWORDS:** data-model, unique index, scaffolding, sequential attribute, gallery, automatic attribute, round button

## Creating a Queueing system

### PART I - Generate a sequential number

We will be creating our initial version of the Queueing system. The user will be able to see the available queues and request a slot for them. The following is an example of what user interaction could be like, where the user just pressed the Queue button on the Outdoors queue:



1. Create a new application called **MyQueues\_<your initials>** and add a Reactive Web App Module with the same name
2. Define the queue concept
  - a. Create the **Queue** entity, with attributes the following mandatory attributes:
    - A **Text** attribute named **Name**, that can hold up to **20** characters
    - A **Text** attribute named **Code**, with **2** characters
  - b. Add a unique index to the newly created entity
    - There cannot be two queues with the same **Code** value
3. Define the queue slot concept

- a. Create the **QueueSlot** entity, with the following attributes
  - A mandatory *Queue Identifier* attribute named **QueueId**
  - An mandatory *Integer* attribute named **Position**
  - A *Date Time* attribute named **IssuedOn**, with default value **CurrDateTime()**

---

**NOTE:** By providing a default value of **CurrDateTime()** we don't need to assign it when creating a new record, it will automatically be set to the current instant

---

- b. Add a unique index to the newly created entity
    - There cannot be two slots with the same *Position* value
4. Use scaffolding to create the screens to list and edit Queues
5. Add a wrapper **QueueSlotCreate** server action to create a new queue slot and assign it a unique sequence number
  - a. Obtain the highest currently assigned *Position*

---

**NOTE:** You may want to consider using the *Max()* aggregated function

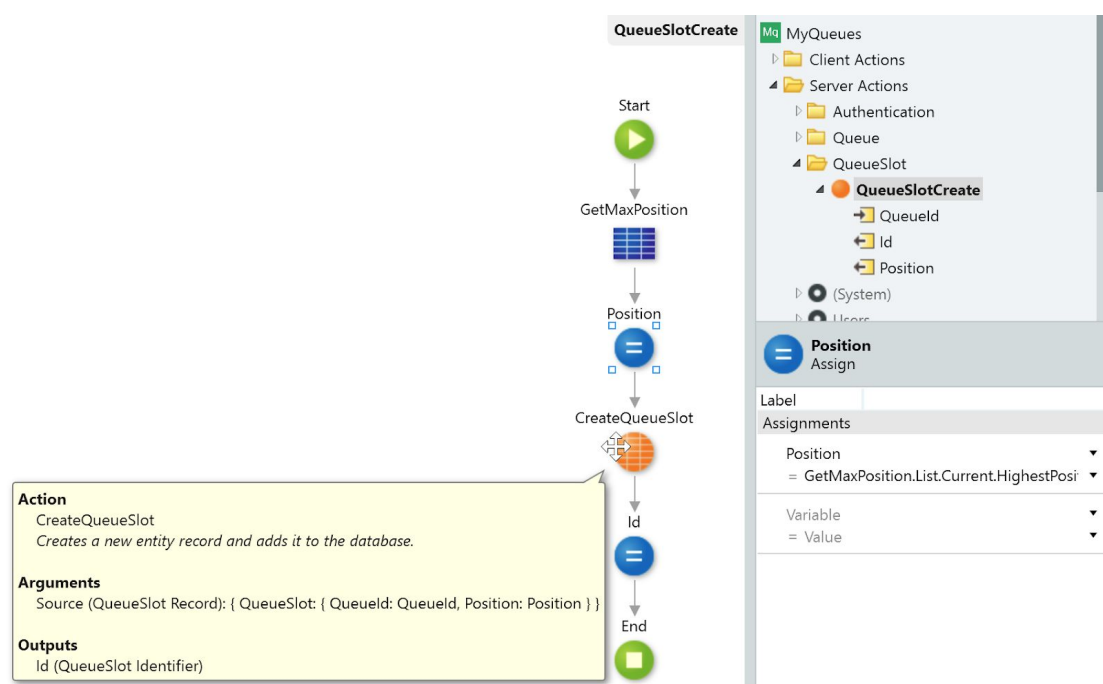
---

- b. Assign to the record's *Position* the next number in sequence
    - c. Create the **QueueSlot** record in the database
    - d. Return not only the **Id** of the newly created record but also the **Position** assigned

---

**NOTE:** Since we are assigning automatically values and the only information we need is the Queue Identifier, that can be the single input parameter of the wrapper action, no need to receive a **QueueSlot** record

---



6. Create a **Home** screen that displays the last assigned position of each queue and allows users to request a new queue slot. Don't forget to make it the *Default Screen* of the module

a. Obtain last position assigned to each queue

- Create an Aggregate that gets **Queue** information, as well as the maximum value for the Position attribute of its associated **QueueSlots**.

**NOTE:** You can use the *Group by* functionality of Aggregates, along with the *Max()* aggregated function to get the desired outcome.

Remember a *Queue* may not have any *QueueSlot* records yet, so the Join should be **With or Without**

GetQueuesWithQueueSlots

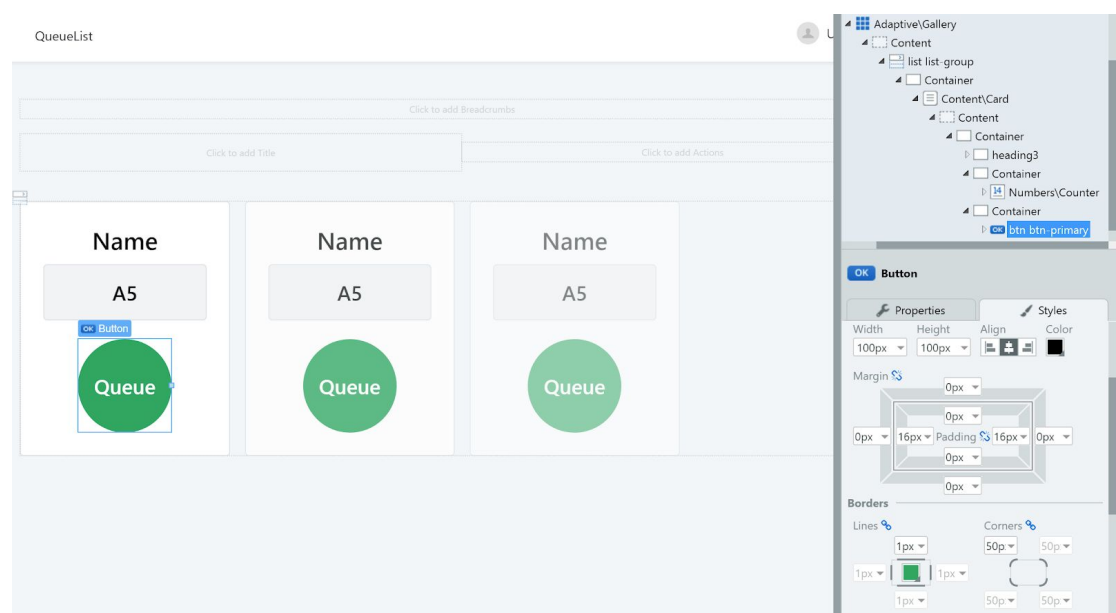
SOURCES: Queue, QueueSlot

JOINS: Queue (Queue.Id = QueueSlot.QueueId) With or Without QueueSlot

Group of Id	Group of Name	Group of Code	Max of Position	QueueSlot	QueueSlot
Id	Name	Code	LastPosition	Position	IssuedOn
1	Indoors	A	3	1	2020-03-19 06:19:24
				3	2020-03-19 06:19:29
2	Outdoors	B	6	2	2020-03-19 06:19:27
				6	2020-03-19 06:19:46
3	Takeaway	T	5	4	2020-03-19 06:19:31
				5	2020-03-19 06:19:33

b. Display a grid of Queue cards

**NOTE:** You can use the *Gallery* pattern to display the content of a list evenly distributed, and the *Card* pattern to visually separate each queue's info, like this:



- For each Queue, show its name, last assigned sequence number and a large round button to request a new slot

---

**NOTE:** In order to create a round button, all you have to do is set its *Width*, *Height* style properties to the same value, and the *Corners* property to half of that value.

---

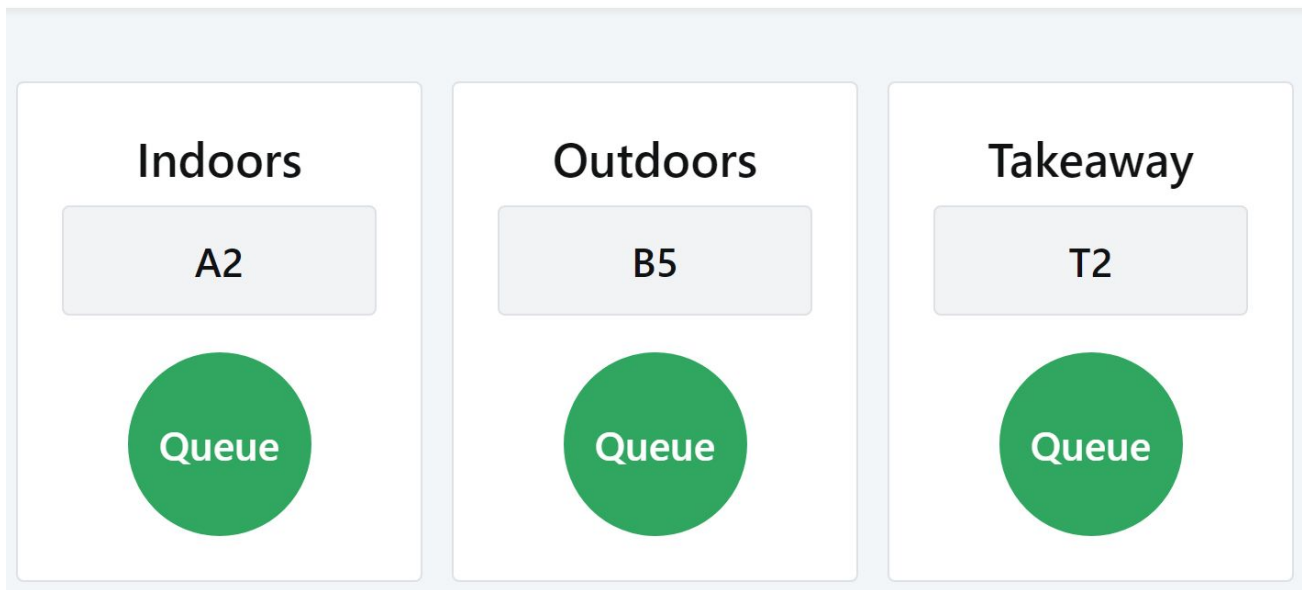
- The button's screen action should call the wrapper **QueueSlotCreate** server action, and give feedback about the queue and position slot assigned, as well as refreshing the screen info for that queue.
7. Publish your application and test it by setting up a couple of queues and adding multiple queue slots to each of them.

END OF PART I

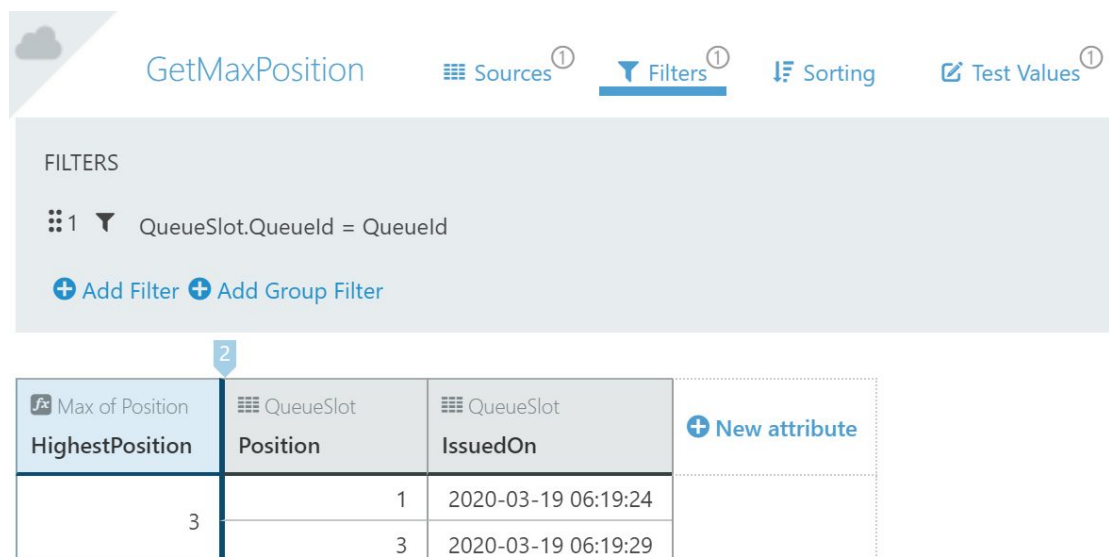
## PARTII - Generate independent sequence numbers per queue

We will now update the Queueing system so queue slots grow independently whenever the user requests a slot for them. The following is an example of what user interaction would be by the end of the exercise

Mq MyQueues QueueList



8. Modify the QueueSlot entity to allow independent sequence numbers
  - a. Change the existing index to include not only the *Position* attribute but also the *QueueId* attribute
9. Modify the wrapper **QueueSlotCreate** server action to assign the a unique sequence number per queue (instead of global)
  - a. Modify the aggregate obtaining the last position assigned to filter it by QueueId



The screenshot shows the configuration for the 'GetMaxPosition' server action. The 'Filters' section contains a single filter: 'QueueSlot.QueueId = QueueId'. Below the filters, there is a table with two columns: 'QueueSlot' and 'QueueSlot'. The first column is labeled 'HighestPosition' and the second is labeled 'IssuedOn'. The table has two rows of data. A blue arrow labeled '2' points to the 'HighestPosition' column.

QueueSlot	QueueSlot
HighestPosition	IssuedOn
3	2020-03-19 06:19:24
3	2020-03-19 06:19:29



## 10. Allow registered users to reset queues

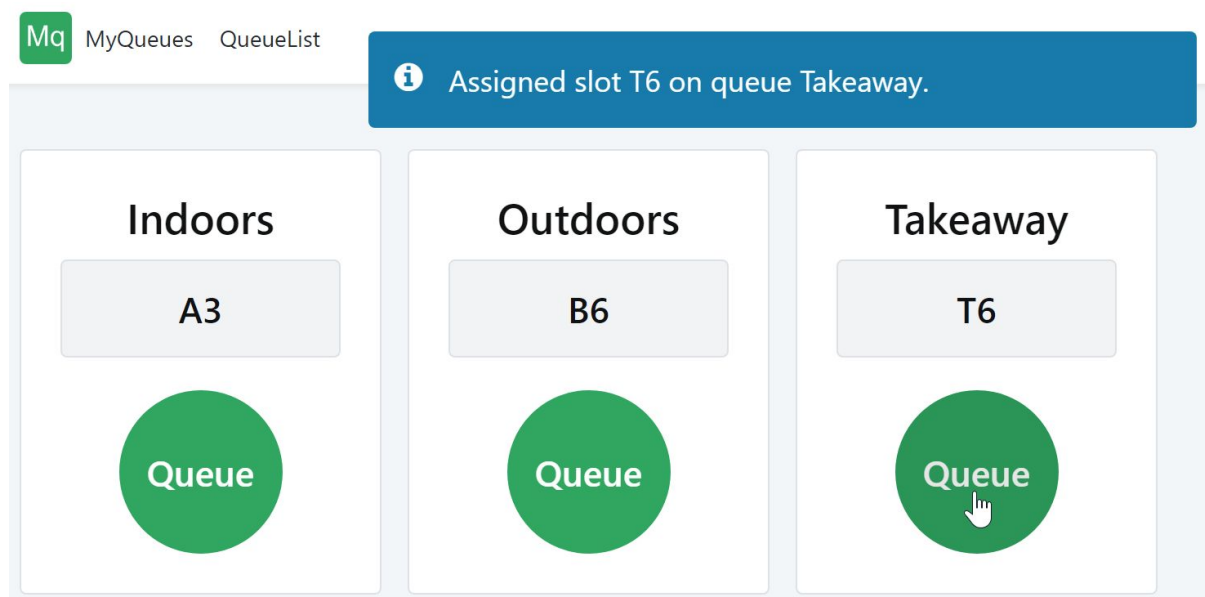
- a. Make sure that the Queue listing and details screens are only accessible to users with role *Registered*
- b. Add a **Reset Queues** button to the *Actions* placeholder of the Queue listing screen
  - The screen action should delete all QueueSlot records

---

**NOTE:** The SQL tool is particularly adequate for this sort of bulk database operations, but can only be called from a Server Action. Consider defining a wrapper **QueueSlotDeleteAll** server action and use it here

---

## 11. 1-Click Publish the application, and check that the Queues now have their sequence numbers grow independently.



## 12. Go to the Queues listing and click on the Reset Queues button. The *Home* screen should now show all queues back to the beginning of the sequence

---

**NOTE:** We have changed the data-model and associated business rules and want to guarantee all queues' sequence numbers are reset to initial values so the correct business rules apply.

---

END OF PART II