

DATA MODELING

Data Page – its Configuration, Usage & Execution

20 comments – March 31, 2020



Written by
OSP Editorial Team

Summary: This article discusses about Data Page, its configuration, usage and execution.

Before getting into the discussion of the data page, let's explore a few key terminologies for better understanding.

Memory

Memory is just like a human brain to store

information/data.

We will have a few friends who will tell us what they did for us at the age of 2 when we are at the age of 30+ 🤔. They tend to have good memory power. On the other hand, we will have a few friends who will forget what we did for them last week 😞.

In the same way, System has two types of memory using which we can access data.

Permanent Memory

This type of memory will store the data in a storage space, which can be retrieved whenever required. Typically, it is a **database** [Oracle, SQL, Postgresql].

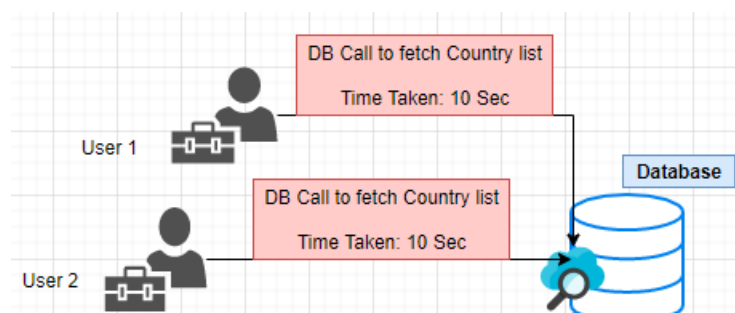
Temporary Memory

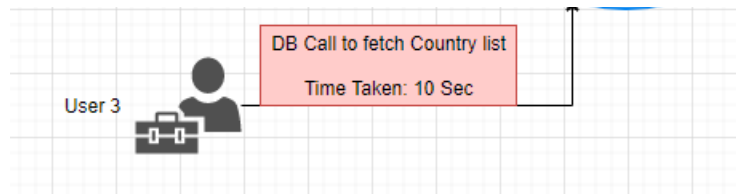
Temporary memory will hold the data only for a specific time period.

Rules can be referred from temporary memory [Rule Cache] & **Data** can be referred from temporary memory [Clipboard].

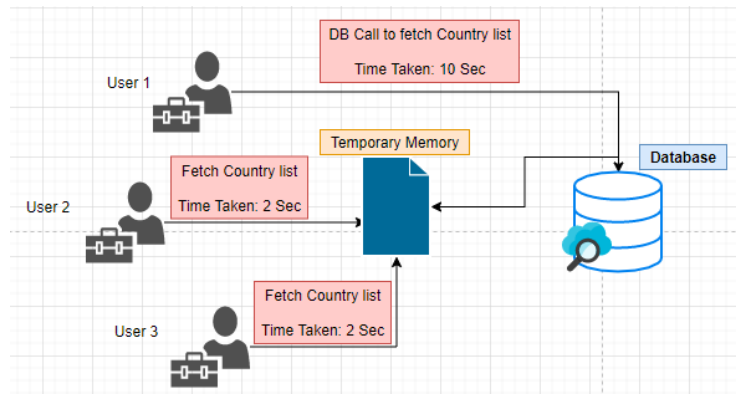
Why Temporary Memory?

Pega uses the Clipboard Page as temporary memory. Let's compare the below two diagrams to see the usage of **temporary memory** [Clipboard] in terms of data retrieval.





When there is **no temporary memory**, every time when information is required, a DB call happens to fetch the data which is very expensive.




When there is a **temporary memory**, the first time DB call happens to fetch the required information and sets the data in the temporary memory [**Clipboard**]. Subsequent calls will refer to the data directly from the clipboard.

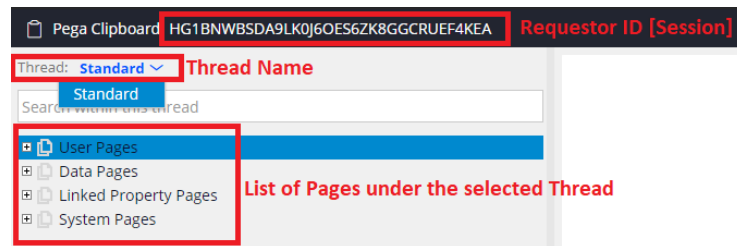
What is a Page?

Let's take a famous book **Ponniyin Selvan** written by Kalki. This book has approximately **2300** pages. Each page in the book has the actual information/data about the story. In the same way, each page in the clipboard has the actual data needed for processing. Pega has categorized pages in the clipboard into different types,

- **User Pages** – Top-level clipboard pages created using activity rule. These pages are specific to thread.
- **Data Pages** – Our hero of the post. We will see

this later 😎.

- **Linked Property Pages** – pages created through [linked properties](#) . These pages are specific to thread.
- **System Pages** – pages created by Pega to maintain system & session related information. We can refer these pages, but, we can't create any System page.



User pages, **Linked property** pages & **system** pages are gone from memory in the following situations,

- When the user logs-off the session.
- When the requestor session times out.
- When the thread is closed or becomes in-active [not applicable for System pages].

Why Data Page?

Few data in the application will be required more frequently. It will be good if we maintain a single copy of data that can be referred whenever needed.

Can the User page satisfy this need? –

No 😞

User pages are specific to thread and become inactive when we close the thread. Hence Pega provides **Data Pages**, which can be reused across a thread, a requestor or multiple requestors (node

level) based on the configuration.

What is a Data Page?

- Data page is a clipboard page that can be **shared** across a **thread**, a **requestor** or **multiple requestors** (node level).

• Data page rule is an instance of Rule-Declare

Share:



Menu



Node



Records Explorer.

- Data pages are loaded when it's **first referenced** during execution.
- Data pages can be **parameterized**.
- Data pages can be **reloaded/expired** based on the configuration defined.
- Data Pages always starts with the prefix **"Declare_"** (legacy). Data pages are informally called as **DPage**.

How to Configure a Data Page

- Similar to other rules in Pega, the New Data Page accepts **Name**, **Class** & **Rule Version**.

- Data Page configuration is simple when you know the actual purpose of each of the fields. Let's quickly see the usage of each,

MENU

Home

All Articles

Suggest Topic

Contact

Forum

Follow me:

Type here to search...

Definition

This tab of the Data page rule has 3 main sections,

- Data page definition
- Data sources
- Post load processing


Data page definition

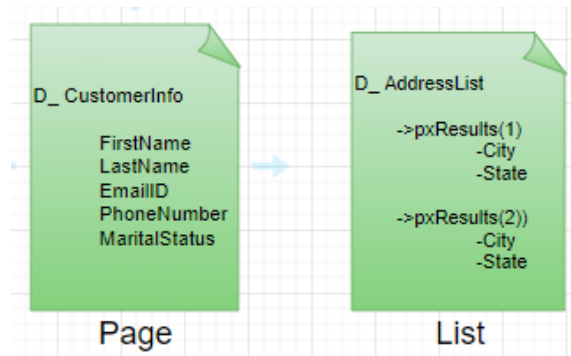
This section of the data page has most of the configurations which describes how the data page is in clipboard.

The screenshot shows the 'Definition' tab of a data page rule configuration. The 'Data page definition' section is active. It contains the following settings:

- Structure:** A dropdown menu set to 'Page'.
- Object type★:** A dropdown menu set to 'OSP-Data-Customer'.
- Mode:** A dropdown menu set to 'Read-Only'.
- Restricted feature set for high throughput:** An unchecked checkbox with a help icon (?) to its right.
- Scope:** A dropdown menu set to 'Thread'.

Structure

- This defines the mode of the data page. Possible modes of a data page can be **Page & List** [Page List].
- The list structure data page provides an additional option to configure keyed access. Keyed page access is one of the [data access patterns](#)  available in Pega which can be seen in detail in a separate post.



Object type

Each page in the clipboard will have a class context. For Example, the **OperatorID** page is of class **Data-Admin-Operator-ID**, **pxRequestor** page is of class **Code-Pega-Requestor**. In the same way, each data page in the clipboard belongs to a class and this field holds that class name.

Property	Value
CustomerID	C-01
EmailID	contact@osp.com
FirstName	OneStop
LastName	Pega
MobileNumber	
pxObjClass	OSP-Data-Customer
pzInsKey	OSP-Data-Customer C-01

Mode

This field decides where exactly the data page will be shown in the clipboard. It can be **Read-Only**, **Editable & Savable**.

- **Read-Only**– The data page of this mode will be used only for referring data. We can't update the information on this data page. It can be modified only during loading or in Post processing activity. Read-Only data pages can be seen under the Data Pages category in the clipboard.

- **Editable** – The data page of this mode can be used for referring the data as well as it can be updated on the fly using Activity, Data Transforms, etc. This mode of data page works similar to User Pages and can be seen under the User pages category in the clipboard.

- **Savable**– The data page of this mode can be used to persist information directly into the database. We can explore the Savable Data page in detail in a separate post. These modes of data pages are similar to User Pages and can be seen under the User pages category in the clipboard.

High Throughput

This option is applicable only if the mode of the data page is **Read-only**.

We can improve clipboard performance in data page loading by enabling this option. Enabling this is recommended when the data is going to be static and used repeatedly. **For Example**, a data page to hold the list of Countries.

When this option is enabled, the below feature will not be supported in Data Page.

- Declarative rules.
- Page messages.
- Complex property references [Property-Ref].
- Saving pages to a database.

Lightweight clipboard mode

This option is applicable only if the mode of the data page is **Editable**.

We can improve clipboard performance in data page loading by using the Lightweight clipboard mode option.

When this option is enabled, the below feature will not be supported in Data Page.

- Properties referencing a data page.
- Linked properties.
- Change tracking for property value updates in the user interface

Scope

Let's see the below architecture before defining scope.

The scope of a data page can be **Node** or **Requestor** or **Thread**.

Node

- If a data page is defined with node-level scope, then **all the requestors** & its related threads (BROWSER, APP, BATCH, PORTAL) running in the same node can share the same data page.
- Node level data pages can only be of mode

Read-Only.

Requestor

- If a data page is defined with requestor level scope, **the threads active** in that requestor can access the same data page.
- Requestor level data pages can be of mode **Read-only, Editable** and **Savable**.


Thread

- Thread level data pages are similar to user pages. If a data page (having case-related information) is defined with thread-level scope, it can be accessed **multiple times** by the **same thread** without having it to reload.
- Thread level data pages can be of mode **Read-only, Editable** and **Savable**.

For Example: Assume that you have launched a Case manager portal and opened multiple cases in a series of tabs. The cases opened in each tab runs in a new thread. If a data page having case-related information is defined with Thread level, It can be reused by the same case multiple times. If a data page is defined with the Requestor level, then all the cases opened in each tab can access the same data page.

Data sources

This section gives us an option to configure the actual source of our data page.

- We can select any of the available sources based on the structure of the page and load the data page.
- We can also have multiple sources for a data page and decide on those conditionally using a when rule.
- Only Activity of type Load Data Page can be used as a source of data page when [Activity](#)  is selected as the source type. The activity type can be changed under the security tab of the activity rule.

Post load processing

- We can't ensure that we always have the required data readily available to us. There might be a situation where we should manipulate information based on the results.
- Post loading activity that we specify here will be used to perform any custom manipulations on the results.

Load Management

- This tab of the data page rule is more important as the configurations available here ensures that the data is up-to-date.

- Load management is only applicable when the mode of the data page is **read-only**.

Page Management

- This option is most likely used by the developers when they want to manually remove/flush the data page from the clipboard.

- We can refer the below-mentioned function in an activity/data transform to remove all instances of the data page from the clipboard irrespective of the parameters.

@(Pega-
RULES:DeclarePages).pzDeleteAllInstancesOfDeclarativePage(tools
"D_XXXXXXXXXX")

Load authorization

- This section is only applicable if the scope of the data page is **Node**.
- Requestor and Thread level data pages use the application context of the logged-in user. Node level data page is outside of the requestor context and hence the application context will be decided

based on the access group provided in this field.

Refresh Strategy

This section enables us to define a mechanism using which the data on the page remains up-to-date.

Reload once per interaction

- Enabling this option will freshly load the data page whenever it is referred. Careful consideration is required before enabling as this may cause a huge performance issue.
- This option is available only when the data page is of scope **Thread** or **Requestor**.
- When selected, the system ignores any values in the **Reload if older than** and **Do not reload when** fields at run time.

Do not reload when

- We can also control the refresh strategy of the data page using a when rule.

With this configuration, during data page reference, the system validates if the customer is in-active. If yes, the data page will not be refreshed else, it will be refreshed.

- This option is available only when the data page is of scope **Thread** or **Requestor**.

Reload if older than

- We can make a data page reload based on the time duration in **Days, Hours, Minutes** and **Seconds**.

- When the data page runs for the first time, it sets the cache with the expiration time based on the configuration.

- The next consecutive data page calls will check if the expiration time has reached or the page is stale. If yes then reloads the page, else fetch the values from the existing page.

Page limits

- This option is most recommended when the data page is of mode **read-only** with scope as

Requestor or node.

Limit to a single data page

- This option is only available when the data page has at least one parameter defined. This can be seen in the later part of the post when we discuss the parameters tab.

Clear pages after non-use

- This option when selected forces the system to delete the clipboard instances of the data page after an interval passes with no access. Subsequent attempts by a requestor to access the data pages cause the pages to reload.
- The time interval until it waits is defined in the DSS ***DeclarePages/DefaultIdleTimeSeconds***. This value will be used to expire the page. By default, DSS value is set to **86400** seconds or **one day**.
- we can adjust the value of the DSS based on our needs.

Parameters

Similar to other rules in Pega, data pages can be parameterized. Parameters defined in the data page define the actual data on the page.

Parameter values can be used in conjunction with the settings ***limit to a single data*** page in the load management tab.

- When the ***limit to a single data page*** is enabled, the system only maintains the latest instance of the data page in the clipboard.
- When the ***limit to a single data page*** is not enabled, the system maintains all the instances of the data page in the clipboard.

Let's create a data page for a real time business scenario to understand it better.

Business Scenario:

Consider OSP Organization has an HR Onboarding application. Candidates can apply for a job via the company's site or via Naukri. Candidate details are available in OSP's database if candidates use Company's site and in the external system if they use the Naukri site to

apply for a job.

Let's now see how we can configure a data page to fetch candidate information conditionally based on the application source.

Implementation

D_CandidateDetail – To get the Candidate detail from either local data table or from an external source via a connector.

Step 1: Goto Records Explorer -> Data Model -> DataPage -> Create

Step 2: We can change the data page definition if needed. As per our requirement, cases are created for each candidate, hence, the Scope should be Thread and Structure should be page. Since we are just going to display the candidate details, we can choose the Mode as Read-only.

Step 3: Configure the data source to get a candidate detail from the candidate table. To do that, we can use the lookup option and provide the class name with the key parameter passed.

Step 4: Once the data page to load data from the Data table is configured, we can add another source to load data from service.

We can add new source by Clicking Add Source option

Step 5: *Create Data Source* provided in each Source is to configure the new Source. In our scenario, we can create a Connect Rest by configuring it. Creating a connect rest will be covered in our future post.

Once the Connector rule is ready, we can choose the connect rule and configure its Request and Response Data transform as mentioned below.

Note the highlighted classes, since the class of

the connector and class of the data page are different, it's mandatory to provide Response data transform to map the data from the int layer to the Data layer.

Execution

Login as HR and select a Candidate ID to view the information. The Source of the profile selected here will be conditionally used in the data page.

Since the selected source is the OSP site, the data page loads the result using a lookup.

Unit Testing

Data page can be unit tested directly by running it stand-alone in the designer studio. We can also trace a data page run to see the DB operations, cache hits, etc.

Now we are at the end of the post. We have

covered most of the basic concepts of data pages in this post. We can explore the below features of the data page in our continuation post.

- Error handling in Data Page.
- Asynchronous Data Page.
- Declarative page cache.
- Savable Data Page.
- Keyed page access in Data Page.

Stay tuned for our continuation post 🕶️

Tagged as
data pages

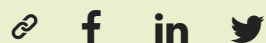
Share:



Written by
OSP Editorial Team

We are a team of 3 CLSA's who strongly believe in the fact that Learning is nothing without sharing.

OneStopPega is purely out of our love and passion for PEGA.



Recent Jobs from our community

Pega developer
job opportunities
in Pegasystems

Hiring #Pega-
Developers #
Work from
Home/Hybrid
#Pan India

Virtusa - Hiring
for Pega
Developers

Pega LSA, SSA -
Morgan Stanley,
Bangalore

[VIEW MORE JOBS ON OSP FORUM](#)

Join the discussion

Feel free to post your questions here
about this topic if any. We will definitely
get back to you **ASAP !!!**
If you have any off-topic questions, Let's
discuss at **OSP Forum** [↗](#)

Comment

Name *

Email *

- ☐ Save my name, email, and website in this browser for the next time I comment.
- ☐ Enable this to get email notification when someone respond back to your comment
- ☐ Enable this to Subscribe for instant email notification on new article

Submit Comment

20 comments

Aditya Kunal

March 31, 2020 at 8:03 PM

Also unit testing means I thought guys u will explain pega it cases. It is misleading guys.

Reply

OSP Editorial Team

March 31, 2020 at 8:16 PM

You can expect a separate post

soon on Pega Unit Testing from us.

Stay tuned and Happy learning
from OSP 😊

Reply

Neeraj

April 1, 2020 at 10:52 AM

Good Explanation

Reply

OSP Editorial Team

April 1, 2020 at 11:14 AM

Thank you so much @Neeraj.

Happy Learning from OSP 😊

Reply

Aditya Kunal

March 31, 2020 at 11:39 PM

Guys it would have been given a different information rather than looking like pdn help if below things were explained guys.

Just my opinion:

- 1) syntax of calling data pages by passing proper parameters from rules like data transform.
- 2) unit test case of data page
- 3) activity methods like load-datapage specific to data page
- 4) what happens in the backend
- 5) whitelist to facilitate offline config for mobile app

- 6) debugging through sma
- 7) talking about access pattern or atleast linking ur previous articles like sor
- 8) how data page syncs between nodes



Reply

OSP Editorial Team

April 1, 2020 at 11:12 AM

1) syntax of calling data pages by passing proper parameters from rules like data transform. – This post focus on how to configure the Data page. This will be covered when we explain in detail about data page usage in our next post.

2) unit test case of data page – We will handle this in a separate post.

3) activity methods like load-datapage specific to data page – This is an asynchronous data page which we said we will explain in our continuation post. Just go through our article completely before commenting.

4) what happens in the backend – We explained about data page cache in this post, not sure if you have seen that. When we explain the asynchronous data page, we will explain in detail on the back-end execution with pool id.

5) whitelist to facilitate offline

config for mobile app – This was never in our scope. These are less realistic topics which are our least priorities. We are planning to bring up guest blogging soon. If you are interested, you can give us an article on this offline capability.

6) debugging through sma – No SMA in v8. Be up-to-date.

7) talking about access pattern or atleast linking ur previous articles like sor – This is already linked in our article. Not sure if you have gone through our article completely.

8) how data page syncs between nodes – Data pages never sync between nodes. Not sure where did you gather this information.

Reply

Ramu

April 1, 2020 at 1:40 AM

Can we change the scope of a Datapage from Requestor to Node, which is already in Production environment. If yes, what are the steps that we need to consider?

Reply

OSP Editorial Team

April 1, 2020 at 11:23 AM

1) Is the data is requestor specific

or most frequently changing data,
then strictly not in Node level.

We would not recommend doing
that in PROD, but if you have a
proper justification then nothing
stops from implementing that.
Please make sure you follow the
below items when doing the
change

- 1) Make sure you have a proper
refresh strategy when you have
scope as node. This ensures the
data is up-to-date.
- 2) Make sure you manually flush all
the existing instances of the data
page in PROD after deployment
[Clear data page in Data Page rule].
- 3) Make sure you clear the
declarative page cache of the
particular data page rule. you can
use the below PegaAPI to clear the
entire declarative cache in a node.
nodes_var_caches_declaratives_9fe8405193f7d8dbdd439c

Reply

Ramu

April 1, 2020 at 12:03 PM

Thank you for the steps. My
intention was to inform that
Datapage is in production
now, not to do the change in
prod 😊
I will try doing these steps in
Dev.

Reply

OSP Editorial Team

April 1, 2020 at 12:13 PM

We even meant the same. Changing the data page in DEV which is already in PROD.

Our bad, for not putting it in proper words.

Make the changes in DEV and let us know of any issues you face.

We will be happy to help you out anytime



Reply

Mahesh

April 9, 2020 at 3:26 PM

Nice post team, very informative.

I have a query below:

What would be the difference between Response Data Transform and Post load processing function, if we want to manipulate the data we can do it any of the above approach right?

Reply

OSP Editorial Team

April 9, 2020 at 8:06 PM

Response Data Transform – Can be used for simple data mapping and other manipulations.

For Eg,

Concatenating First Name, Last Name.

Mapping custom properties to the page/list.

Post Load Processing can be used to perform some complex manipulations that can't be done using Data transform.

For Eg,

Notifying administrators on the execution of the data page.

Mapping a custom property in the page/list which needs to be fetched from an external system.

1) Response Data transform becomes mandatory when Data page object class and data source class are different.

2) Usage of Post Load processing is not recommended until and unless it's really required.

Reply

Praneeth

August 25, 2020 at 11:19 PM

Hi,

For node level scope we need to give access group to load, if that data page is needed by two or more access groups what should we give there.

Reply

OSP Editorial Team

August 26, 2020 at 1:23 PM

We don't think its an ideal scenario.

But then we can have one Batch Access Group created with required applications as built-on and configure it to the node level data page.

Reply

Sayanee

September 3, 2020 at 6:32 PM

Is there any way we can source data type into a text input . I know we can source from data type in a table. The format I am looking for is, in first screen I will give customer id, in second screen , I will populate the customer details from data type based on id.

Reply

Dharma

September 16, 2020 at 7:27 AM

Excellent. I like the example Ponniyan selvan 😊 you guys love the language it seems. appreciated

Reply

DO MINH TUNG

October 14, 2020 at 11:34 AM

From today I become a big fan of Page.
Here have all that I need.
Thank you

Reply

OSP Editorial Team

October 15, 2020 at 9:32 AM

Thank you so much!

Happy Learning from OSP 😊

Reply

Suresh

January 21, 2021 at 12:58 PM

Hi Team,

If we specify different access group for
Node level Data page, and user with
another Access Group can access the
Node level Data page?

Reply

Ram

September 19, 2022 at 7:39 PM

Very good articles.

Reply

JOIN THE DISCUSSIONS AT OSP FORUM

How to install Pega on prem in government space - *3 thoughts*

Job Scheduler - *1 thoughts*

How can we generate PDF in new Tab, from pyAttachStream content. ? - *1 thoughts*

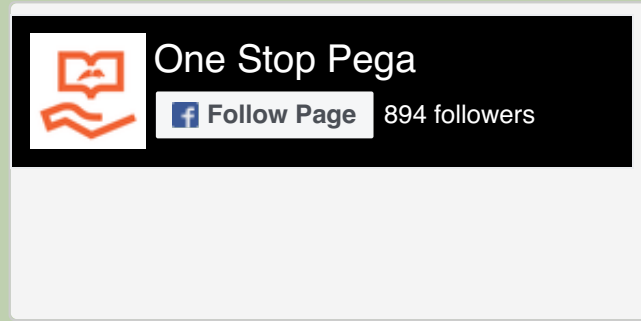
Function/Activity for PGP Encryption - *1 thoughts*

Stop/Start an agent using Pega API in Pega 8x - *1 thoughts*

[Start Discussion](#)

LET'S GET CONNECTED !

Join our OneStopPega social family to get instant updates on articles, news, updates from our Pega ecosystem.



TOPICS

Announcements	2
Background Processing	4
Batch Processing	2
Case Management	6
Contest	2
Data Modeling	9
Decisioning & Declarative	1
Enterprise Design	2
Extensibility	2
Integration	2
Pega Frameworks	1

Performance	4
Reporting	5
Security	1
System Administration	3
Technical	1
User Experience	3

TRENDING ARTICLES



Queue Processor - its
Configuration, Usage & Execution



Job Scheduler - its Configuration,
Usage & Execution



Data Page - its Configuration,
Usage & Execution



Savable Data Page - its
Configuration, Usage & Execution



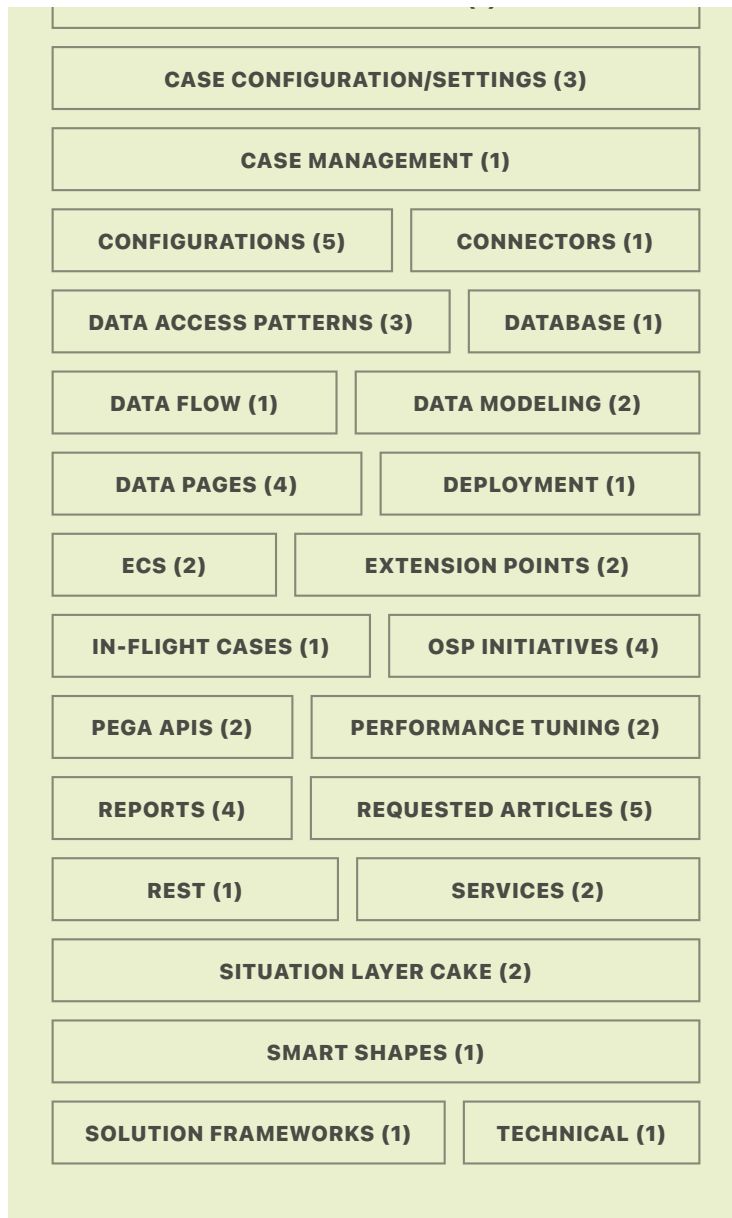
Service REST - its Configuration,
Debugging & Error handling

TAGS

ACTIVITY METHODS (1)

AGENTS (2)

AUTHENTICATION (1)



UPCOMING ARTICLES

Attribute Based Access Control

Business Intelligent Exchange

Campaign execution in Pega Customer
Decision Hub

Report browser in Pega

[Request Article](#)

HOW DO YOU STAY UP TO DATE IN THIS FAST-MOVING INDUSTRY?

*A good start is to subscribe and get notified of new articles by email. We bring the **best content** for our People.*

Email Address

[Subscribe Now](#)

Further reading



DATA MODELING

Aggregate Source in Data Page – its Configuration, Usage & Execution

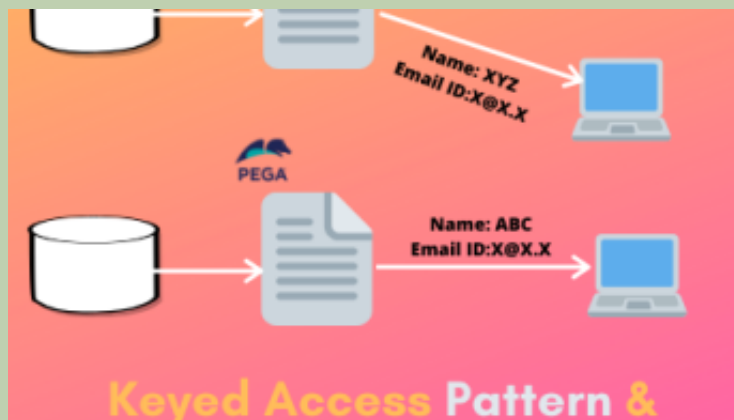
September 13, 2020



DATA MODELING

Savable Data Page – its Configuration, Usage & Execution

April 19, 2020



DATA MODELING

Reference Pattern & Keyed Access Pattern in Pega

April 12, 2020

ABOUT

OneStopPega is a Pega blog that strongly believes in

the power of sharing knowledge. The vision of OSP is to build a single destiny for all Pega thirsts. We as a family can work towards that big milestone. Without you, we are nothing !!!



INITIATIVES

[OSP Forum](#)

[LetsSolveWednesday](#)

[OSP Trivia Contest](#)

[OSP Jobs](#)

ONESTOPPEGA

[About](#)

[Suggest Topic for our next Article](#)

[Sitemap](#)

[Newsletter](#)

[Contact](#)

[Disclaimer](#)

[Privacy Policy](#)

IMPORTANT LINKS

[OSP Forum](#) 

[#LetsSolveWednesday](#) 

[Useful Resources](#) 

[Pega Jobs](#) 

