

RADS

Best Practices on Application Building

<https://github.com/radservice/rad-community/fork>

Prerequisites

1. Understand all major components of RADS.

Content

1. Application Building Best Practices
2. Performance



Chapter I

Application Building Best Practices

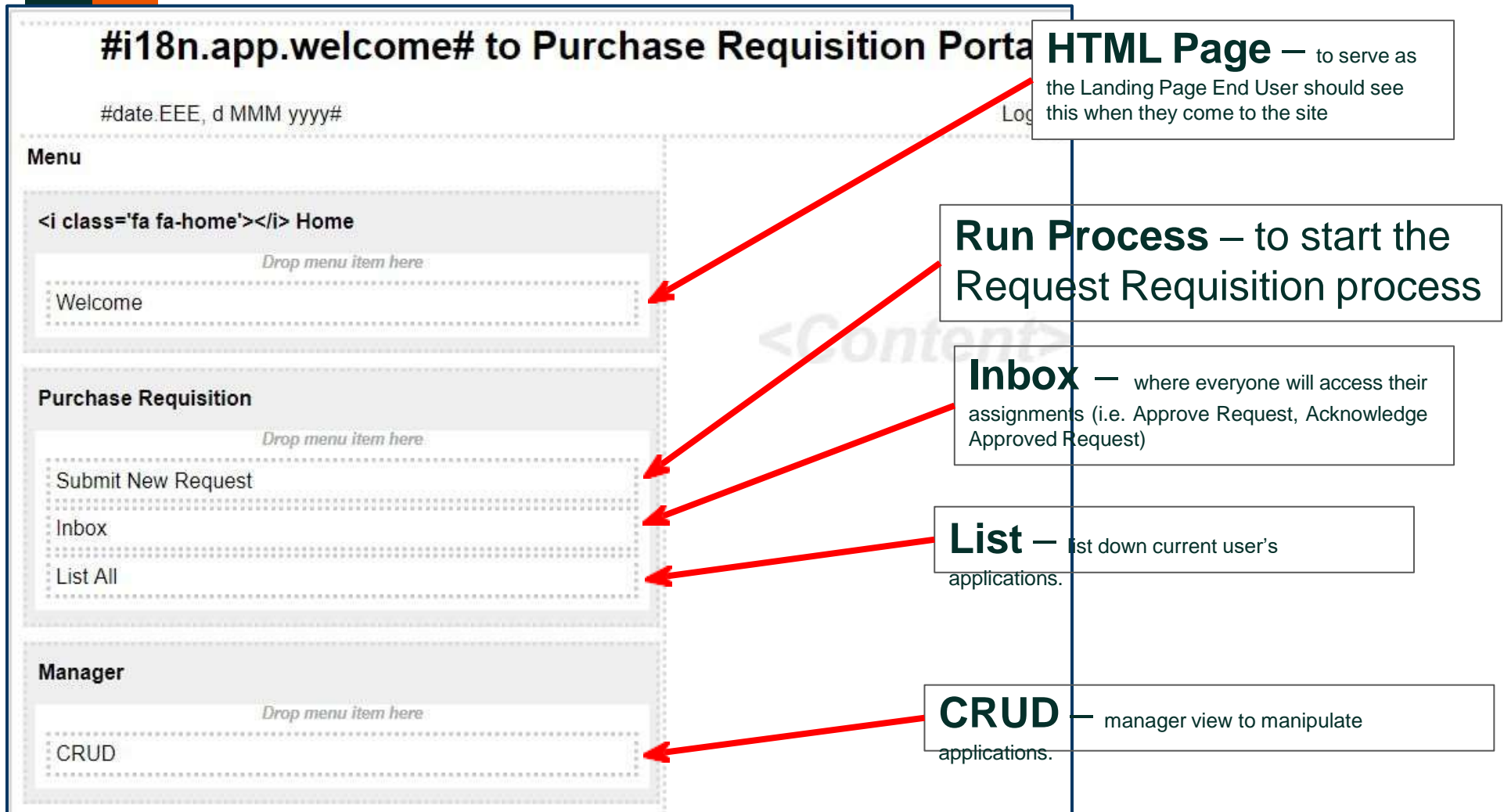
Application Building Best Practices

1. Draw a Sitemap
2. Segregate Userview by Roles
3. Iterative Build Process
4. Presence Indicator
5. Naming Convention
6. Taggings
7. Beanshell Coding
8. App Versioning
9. Process or Data
10. Defaults
11. Notes
12. Comments
13. Chrome Groups

Draw a Sitemap

- Before you start designing your app, try to visualize on all these components would work together.
 - Process
 - Form
 - List
 - Userview
- by using the Userview as the centerpiece of the planning.

Draw a Sitemap - Purchase Requisition App






Segregate Userview by Roles

If you multiple roles and a lot of userview menu elements, it is best to segregate the userview by roles.

Benefits:

- Clear cut Permission Management
- Performance (CRUD / Inbox Count, etc)

USERVIEW

-  App Admin Userview
-  Approver Userview
-  Requester Userview

Iterative Build Process

- Build as you go.
- Establish the most basic layer first.
- Modularize.
- Design, configure and test individual module independently when possible.
- Increase complexity between each iteration.
- Teamwork is good but....
 - Must work in the same server/instance.
 - Must work on separate item/sections at any point of time.
 - Communication and expectation must be set clearly between each team member.
 - Assign a App champion to oversee and delegate tasks.

Iterative Build Process

Incrementally go through each step thoroughly before going to the next...

1. Process Design
 - Verify that all possible routes flow as intended to.
2. Form Design
 - Build and test with and without being part of the process.
 - Define table names with ERD in mind.
3. Form with Process
 - With form and process individually tested and verified, then only add Form layer on top of process layer.
4. Datalist
 - With all Form and Process verified and tested, then only design the Datalist for reporting purpose.
5. Userview
 - Should come last depending on the flow of design you have. With everything tested, one can now add in permission control across all the layers.

Once a cycle is completed, consider versioning it to save your work before moving to next new cycle.

Presence Indicator

- Real-time presence will check to see if someone else is viewing or editing a RADS Component while you have it open
 - Helps avoid conflicts and promote collaboration.

The screenshot displays the FORM BUILDER interface for 'Purchase Requisition v2: Request Form (Published)'. The interface includes a top navigation bar with tabs: DESIGN FORM, PROPERTIES, PREVIEW, SAVE, and GENERATE APP. A red box highlights the presence indicator icons in the top right corner, showing two circular icons with user avatars and names (e.g., 'admin', 'cal').

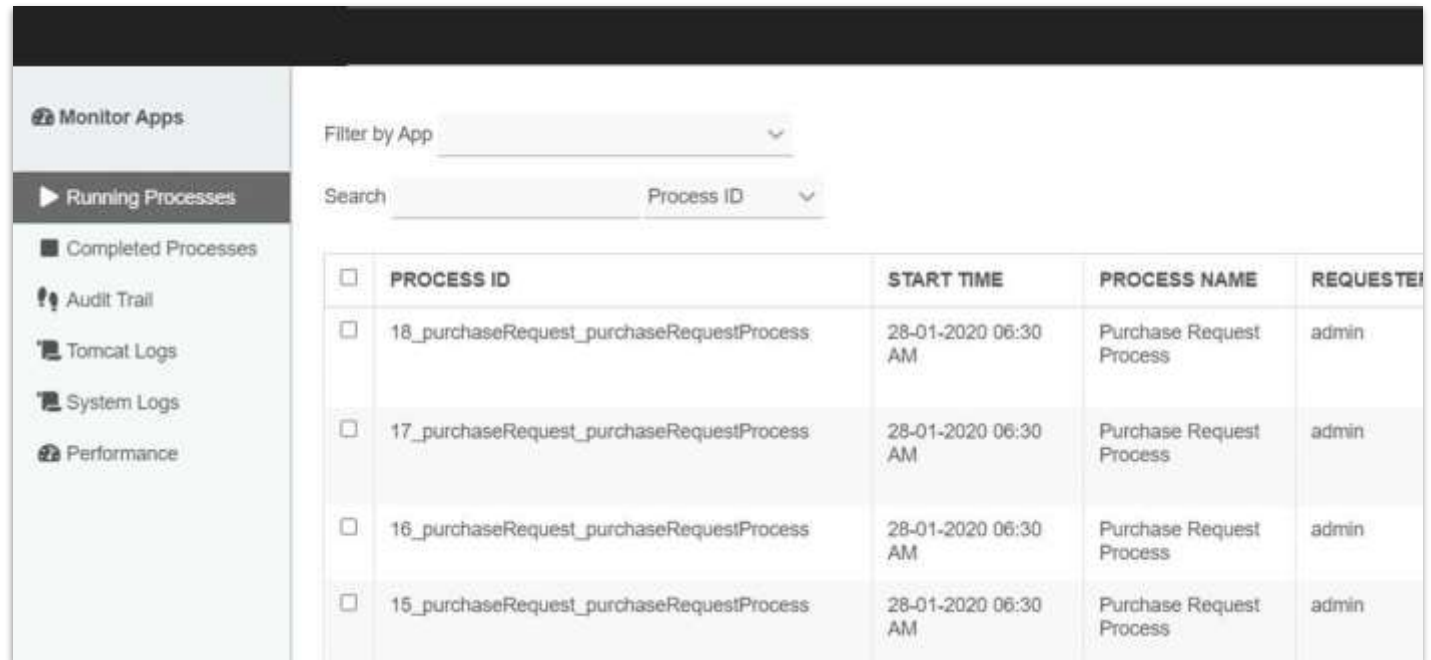
The main form area is titled 'Request Details' and contains the following fields:

- Name:
- Request Date *:
- Category:
- Items: A table with columns Name, Quantity, and Price.
- Remarks:

The left sidebar shows a list of form components: Basic, Currency Field, Hidden Field, Text Field, Password Field, Text Area, Select Box, Check Box, Radio, and Date Picker.

Naming Convention

- Use **camelCase** when appropriate. Particularly:-
 - App ID
 - Process ID
 - Activity ID



The screenshot shows the 'Monitor Apps' interface with the 'Running Processes' tab selected. It displays a table of running processes with columns for Process ID, Start Time, Process Name, and Requester. The process IDs are in camel case, using underscores as separators.

<input type="checkbox"/>	PROCESS ID	START TIME	PROCESS NAME	REQUESTER
<input type="checkbox"/>	18_purchaseRequest_purchaseRequestProcess	28-01-2020 06:30 AM	Purchase Request Process	admin
<input type="checkbox"/>	17_purchaseRequest_purchaseRequestProcess	28-01-2020 06:30 AM	Purchase Request Process	admin
<input type="checkbox"/>	16_purchaseRequest_purchaseRequestProcess	28-01-2020 06:30 AM	Purchase Request Process	admin
<input type="checkbox"/>	15_purchaseRequest_purchaseRequestProcess	28-01-2020 06:30 AM	Purchase Request Process	admin

Because RADS uses underscore (_) as the separator in the naming of Process Instance ID.

Naming Convention

- Use **camelCase** for the rest as well to maintain consistency.
 - Form ID
 - Datalist ID
 - Userview ID
 - Workflow Variable

Naming Convention

- Use **snake_case** when dealing with **database** related fields.
Particularly:-
 - Form Element ID.
 - Form Table Name.
- Define a **prefix** for **Form Table Name**
 - All **apps'** form data are stored in the same database.
 - Prevent other App from writing into other App's set of tables.

Naming Convention

- Not to be confused with Form Table Name on the previous slide, **Form Name** should be named with **process prefix** and **numbering** if it is bounded to a process.
- Example:-

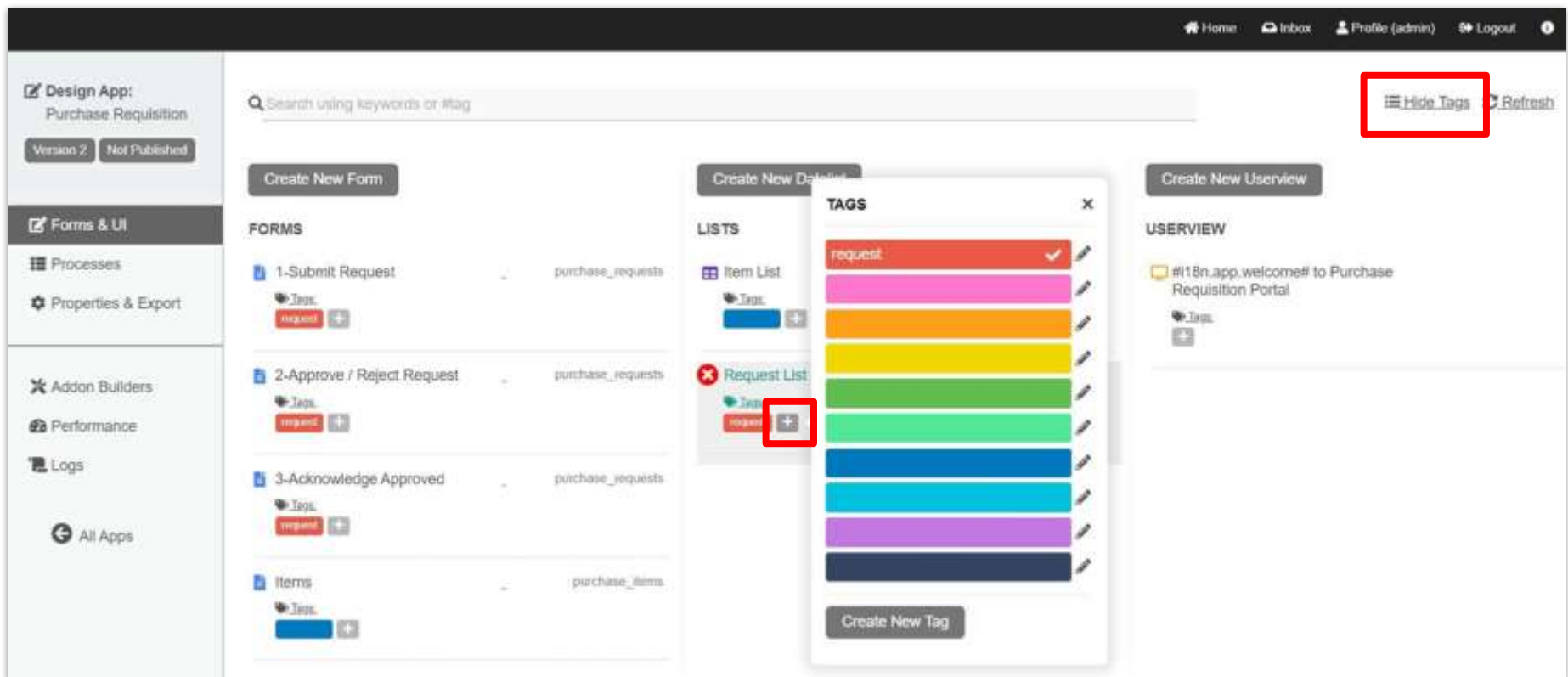
Form Name
R-1-Submit Request
R-2-Approve Request
R-3-Acknowledge Approved Request
QR-1-Submit Quick Request
QR-2-Acknowledge Approved Quick Request

Taggings

- Taggings and Labels are an extra layer of visual data and organization to the RADS components.
- View organized tags at a glance.
- Can also be used in the search filter.

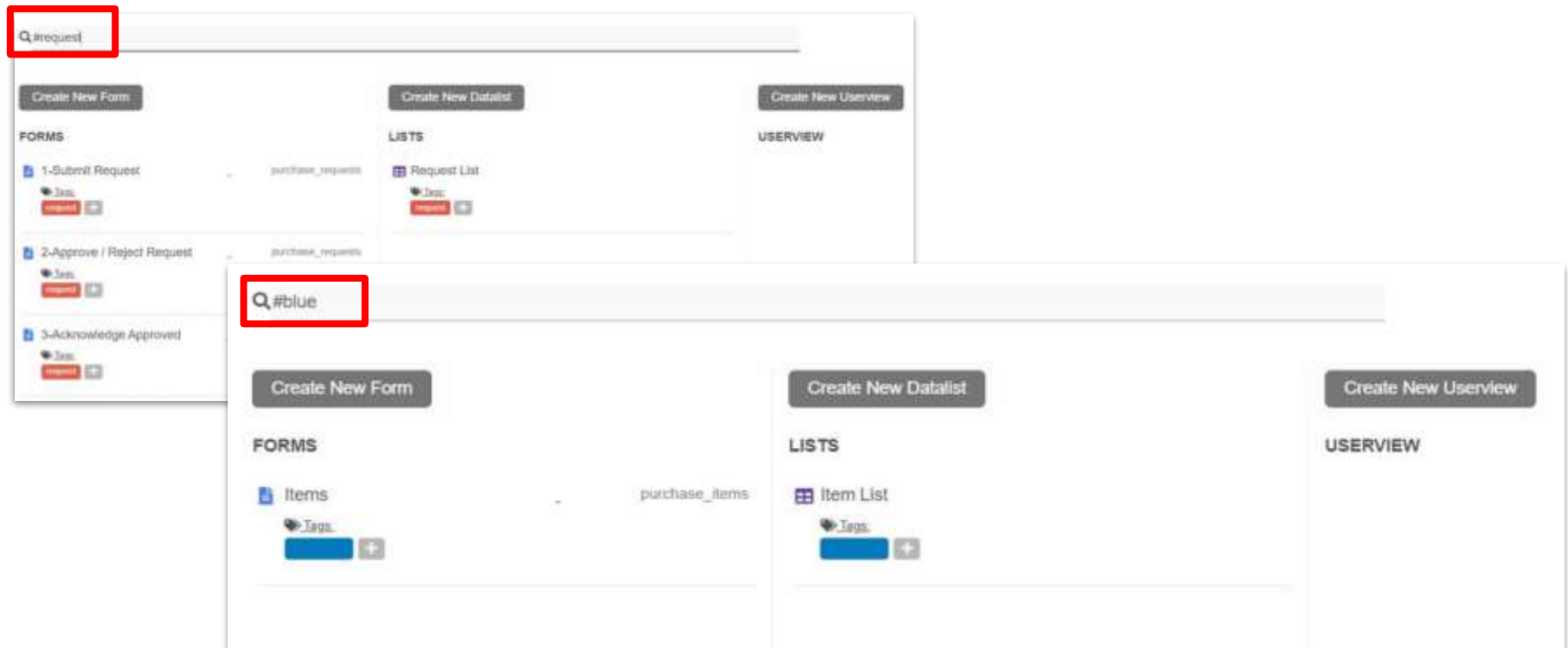
How to Add Tag?

- In **Design App**, click on **Show Tags** and click on **+** sign to add color tags and add labels to the color tags.



Use Search Filter With #tag

- Use Search filter with prefix # to search the relevant tag.
- If no text were defined in the tag, you can search filter by color name.



Beanshell Coding

- Do not use **System.out.println** as it will not be displayed in **Monitor > System Logs** or **RADS.log**
- Use *LogUtil* method instead (see **LogUtil.java**)

Examples:

`LogUtil.debug(className,message)`

`LogUtil.error(className,throwableException,message)`

`LogUtil.info(className,message)`

`LogUtil.warn(className,message)`

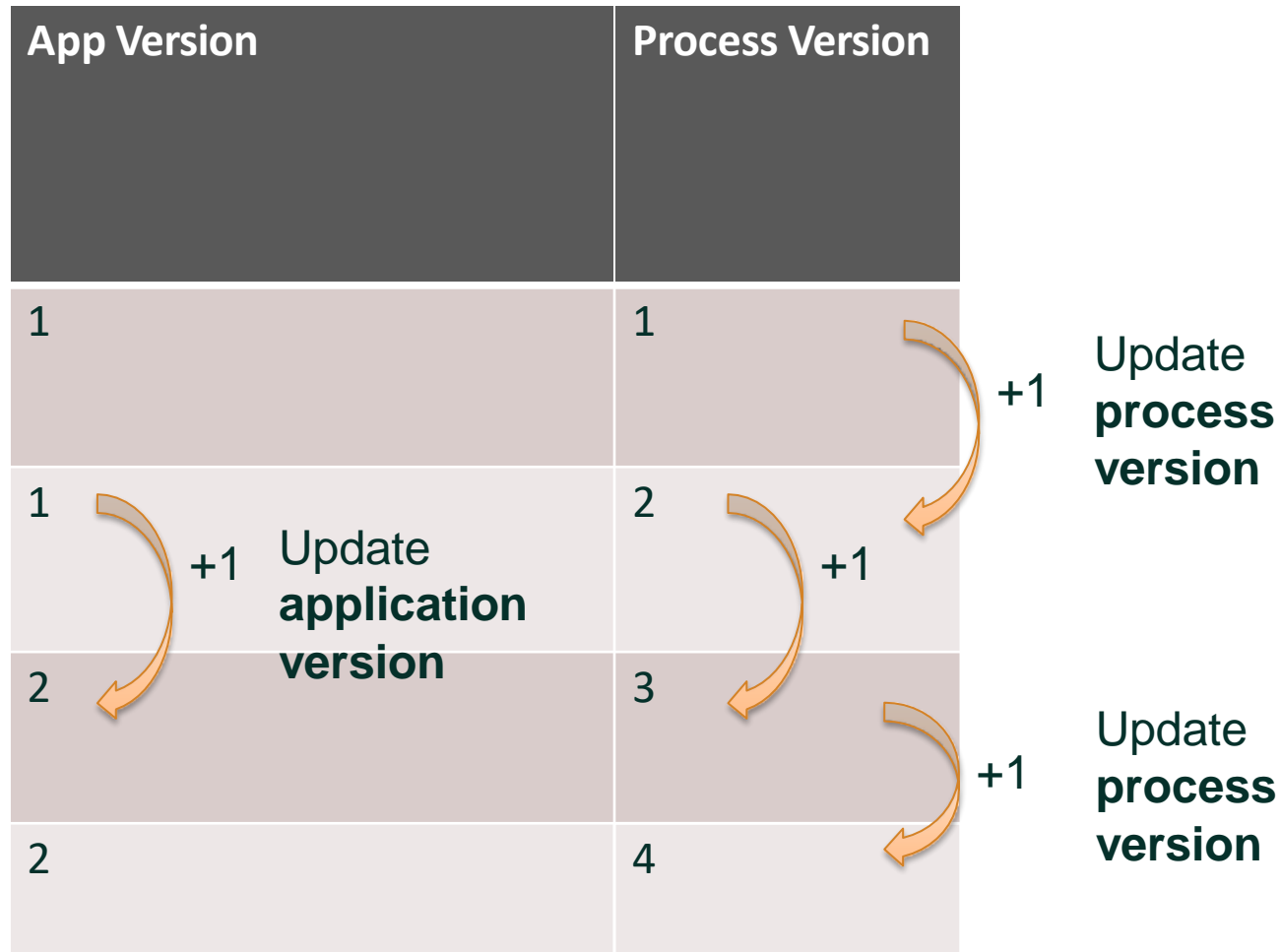
Reference:

<https://docs.rads.purwana.net/Managing+Log+Files>

Version Control - App Versioning

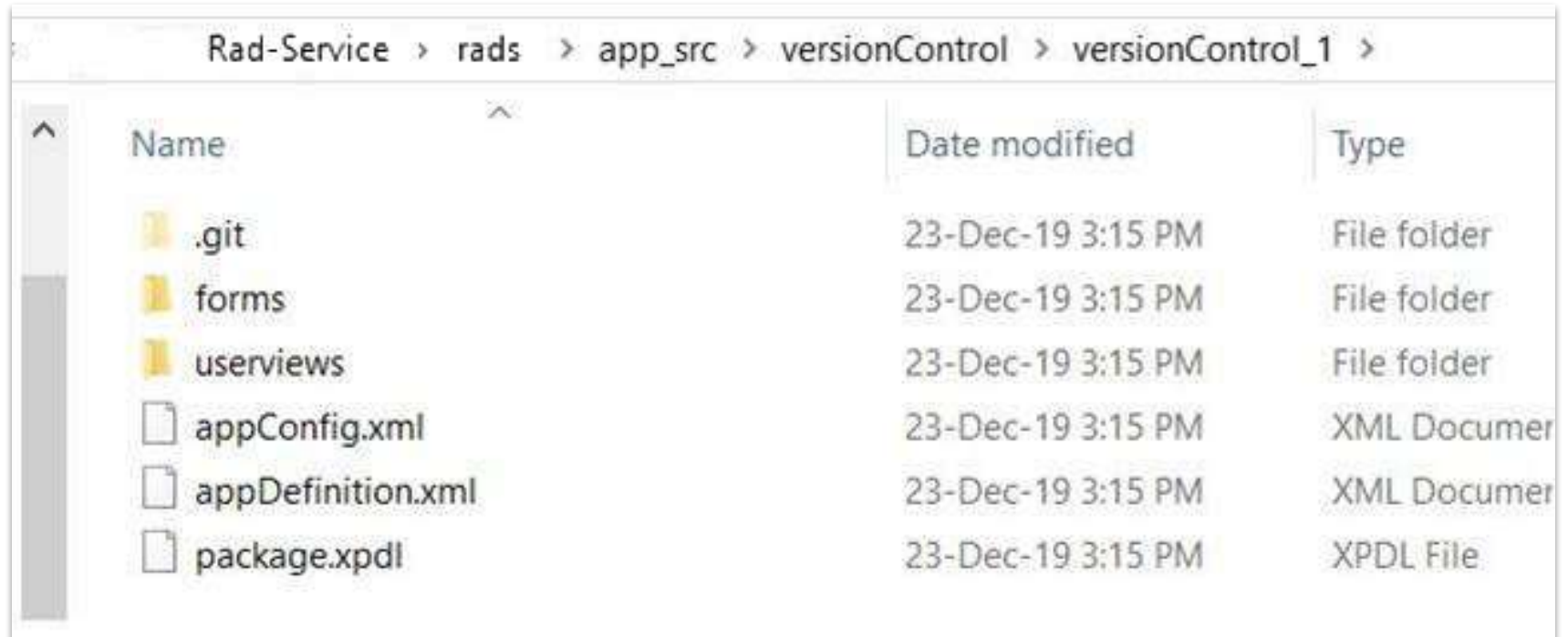
- Between each **development** cycle, it is recommended to create a new App version before moving on to the next iteration.
- Plan your release to the production.
- **Beware:**
 - Each “New Version” and “Import App” would increase the App Version count in your environment.
 - Production server’s app version **!=** Dev server’s app version.

Version Control



Version Control - GIT

- RADS DX has in-built Git version-control system.
- You may use it to track changes made to your app.
- Find out more in Module 12.






























Rad-Service > rads > app_src > versionControl > versionControl_1 >		
Name	Date modified	Type
.git	23-Dec-19 3:15 PM	File folder
forms	23-Dec-19 3:15 PM	File folder
userviews	23-Dec-19 3:15 PM	File folder
appConfig.xml	23-Dec-19 3:15 PM	XML Document
appDefinition.xml	23-Dec-19 3:15 PM	XML Document
package.xpdl	23-Dec-19 3:15 PM	XPDL File

Version Control - GIT

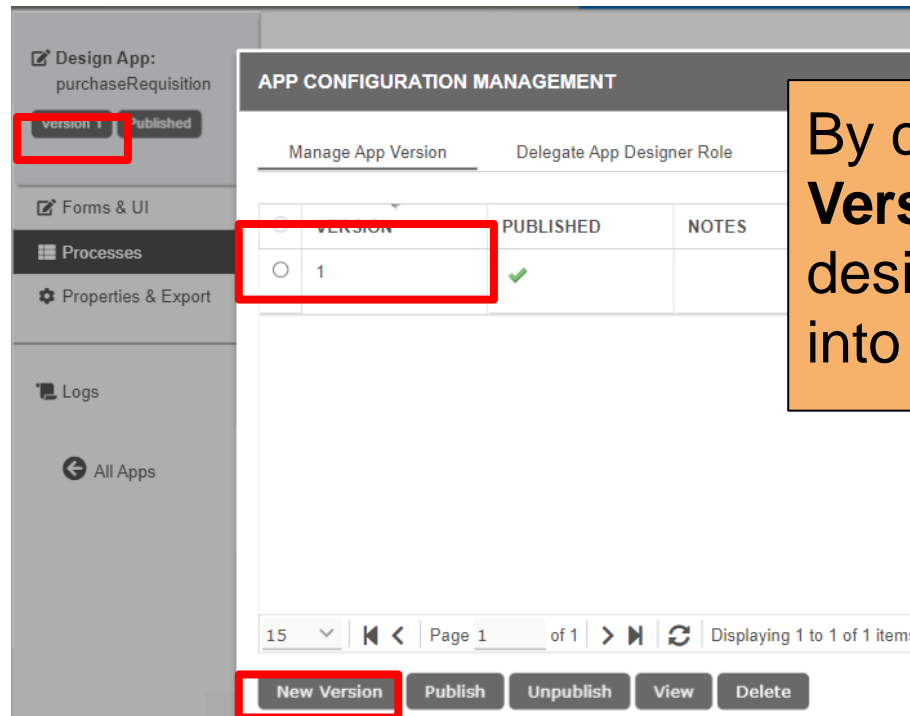
- Ensure to use **named user** during development, it will make it more meaningful to track changes

Commits

Search commits 		 All branches 	
Author	Commit	Message	
  clark	80ac032	Update app definition training. Up...  training_3	
  clark	1ec58da	Add form request. Add list list_requ...  training_3	
  clark	3807a2f	Initial commit for training_3  training_3	
  cat	c5dec58	Add list list_request. Add userview ...  training_2	
  cat	1def325	Add form request. Update app con...  training_2	
  cat	c2382cb	Initial commit for training_2  training_2	
  BCC	fad7997	Update form request.  training_1	
  BCC	d6d4554	Add form request.  training_1	

How to Update Application Version?

1. App Control Panel > Versions > Select version > New Version.



By creating a **New Version**, the App design will be **cloned** into the new version.

Online Reference:

<https://docs.rads.purwana.net/App+Versioning+and+Publishing>

How to Update Application Version?

2. Import App

- By importing the app into a RADS server, the Application Version will **increase by 1** over the existing version already in the server.

What does this means?

When you are dealing with the same app across different RADS servers, you may end up with different app version in each server but with exact same app design.

App Version Across Different Servers

- When you are dealing with the same app across different RADS servers, you may end up with different app version in each server but with exact same app design.

Development Server

Production Server

The image shows two side-by-side screenshots of the RADS application interface, comparing the 'Development Server' (left) and 'Production Server' (right). Both interfaces show the 'Design App: Purchase Requisition' section. In the Development Server, the 'Version 3' button is highlighted with a red box. In the Production Server, the 'Version 3' button is also highlighted with a red box. A red arrow points from the 'Version 3' button in the Development Server to the 'Version 3' button in the Production Server, with the word 'SAME' written above it. Below the buttons, the 'Notes' tab is selected, showing a list of version history. In the Development Server, the first note is 'version 3 2019102' followed by a list of changes. In the Production Server, the first note is also 'version 3 2019102' followed by the same list of changes. A red box highlights the first note in both versions, and a red arrow points from the Development Server's note to the Production Server's note, with the text 'AS NOTED HERE' written below it.

Development Server

Production Server

SAME

AS NOTED HERE

Keeping Track of App Design Across Different Servers

- With the nature of increment of the last app version when an app is imported in, it is imperative to keep track of the “real” app version (app design).
- Make use of **Notes** in app’s properties.

The screenshot displays the RAD Service interface for a 'Purchase Requisition' app. The left sidebar contains navigation options: 'Design App: Purchase Requisition' (with 'Version 8' and 'Published' buttons), 'Forms & UI', 'Processes', 'Properties & Export' (highlighted), 'Performance', and 'Logs'. The main area features three buttons at the top: 'Add Environment Variable', 'Add Message', and 'Add Resource'. Below these are tabs for 'Notes', 'Environment Variable', 'Message', and 'Resource'. The 'Notes' tab is active, showing a list of version history entries:

Notes	Environment Variable	Message	Resource
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			

The notes list includes the following details:

- version 3 2019102
-
- Added deadline to escalate and send reminder
- version 2 20191018
-
- Added email tool
- Added datalist and userview for end user to access
- version 1 20191015
-
- Created form and process for purchase requisition

Process or Data

- Do you really need a Process?
- Are you merely doing CRUD?

Defaults

- To minimize maintenance, one may consider to use...
 - Plugin Default Properties.

The screenshot displays the RAD Service interface. On the left sidebar, the 'Properties & Export' menu item is highlighted with a red box. The main content area features a top navigation bar with buttons: 'Add Environment Variable', 'Add Message', 'Add Resource', 'Set Plugin Default Properties' (highlighted with a red box), and 'Export'. Below this, a tabbed interface shows 'Plugin Default Properties' as the active tab. A search bar is present above a table. The table has two columns: 'PLUGIN NAME' and 'PLUGIN DESCRIPTIC'. One entry is visible: 'Process Data Collector' with the description 'Save process data into app_report_* tables for reporting purposes'.

	PLUGIN NAME	PLUGIN DESCRIPTIC
<input type="checkbox"/>	Process Data Collector	Save process data into app_report_* tables for reporting purposes

About Plugin Default Properties...

- It's good to.... only apply/add in changes after each plugin is tested individually, to steer clear of configuration mistake(s) or bugs.

Defaults

- To minimize maintenance, one may consider to use...
 - Environment Variable, to factorize frequently used values

The screenshot displays the RAD Service interface. On the left is a sidebar with navigation options: 'Design App: Purchase Requisition' (with 'Version 2' and 'Not Published' buttons), 'Forms & UI', 'Processes', 'Properties & Export' (highlighted), 'Addon Builders', and 'Performance'. The main area has a top bar with buttons: 'Add Environment Variable', 'Add Message', 'Add Resource', 'Set Plugin Default Properties', and 'Export'. Below this is a tabbed interface with 'Notes', 'Environment Variable' (selected), 'Message', 'Resources', and 'Plugin Default Properties'. A search bar is present above a table. The table has columns for a checkbox, 'ID', 'VALUE', and 'REMARKS'. It lists three environment variables: 'company_email' with value 'noreply@companyabc', 'company_name' with value 'Company ABC', and 'company_number' with value '012-3456789'.

<input type="checkbox"/>	ID	VALUE	REMARKS
<input type="checkbox"/>	company_email	noreply@companyabc	Company Email
<input type="checkbox"/>	company_name	Company ABC	Company Name
<input type="checkbox"/>	company_number	012-3456789	Company Number

Refer to our hash variable KB to learn how to use environment variables

Defaults

- To minimize maintenance, one may consider to use...
 - Platform-level email settings.

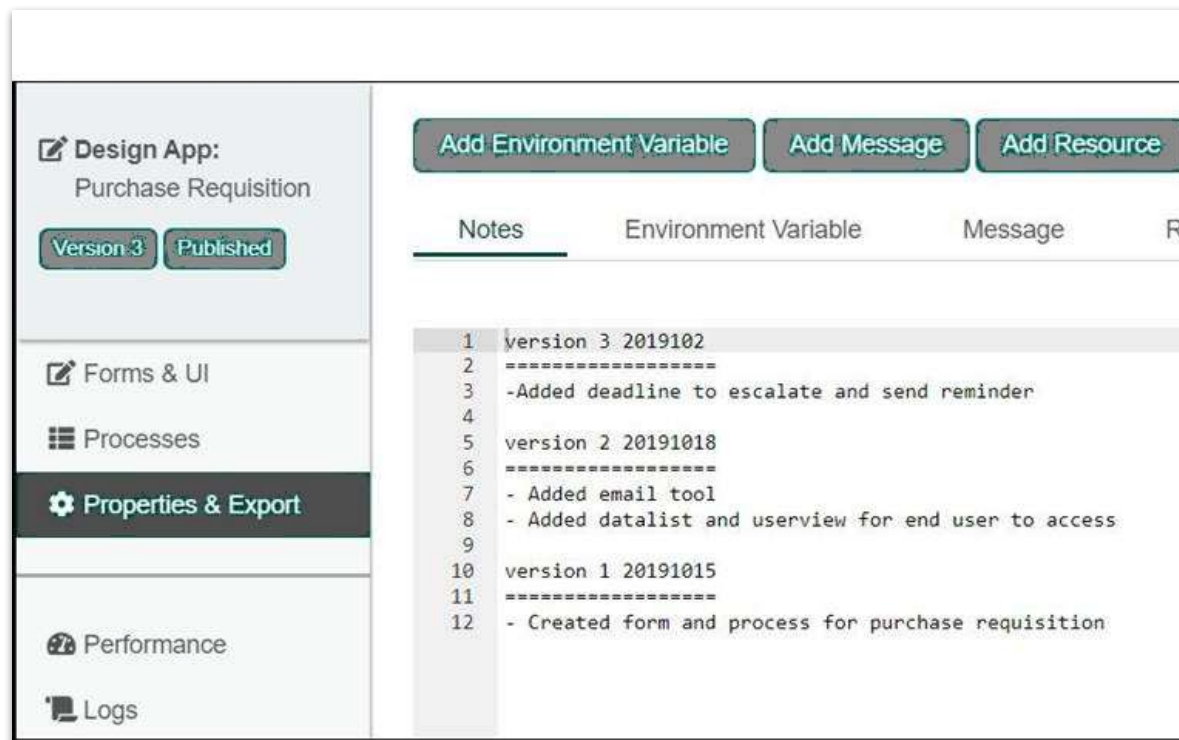
The screenshot displays the 'System Settings' configuration page for RAD Service. The left sidebar contains a list of settings categories: 'System Settings', 'General Settings' (highlighted with a red box), 'Datasource & Profile Settings', 'Directory Manager Settings', 'Manage Plugins', and 'Manage Messages'. The main content area is titled 'SMTP SETTINGS' and contains a form with the following fields:

SMTP SETTINGS	
Host	smtp.sample.com
Port	587
Security	TLS
Username	sender@sample.com
Password
From Email Address	sender@sample.com

A red rectangular box highlights the entire 'SMTP SETTINGS' form area. At the bottom of the form is a 'Submit' button.

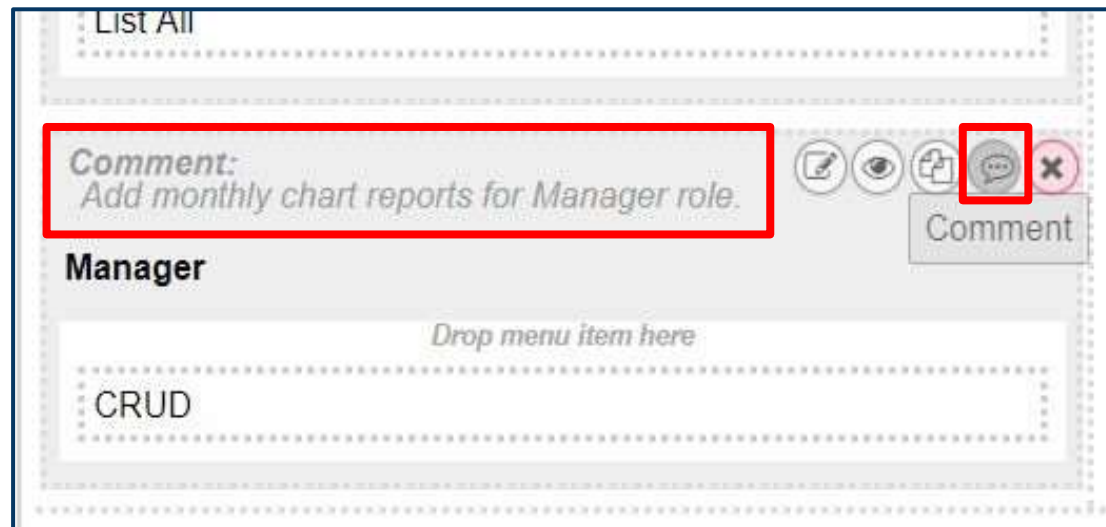
Notes

- Use this feature to jot down important notes for your apps.
- Example: keep track for app development changes.



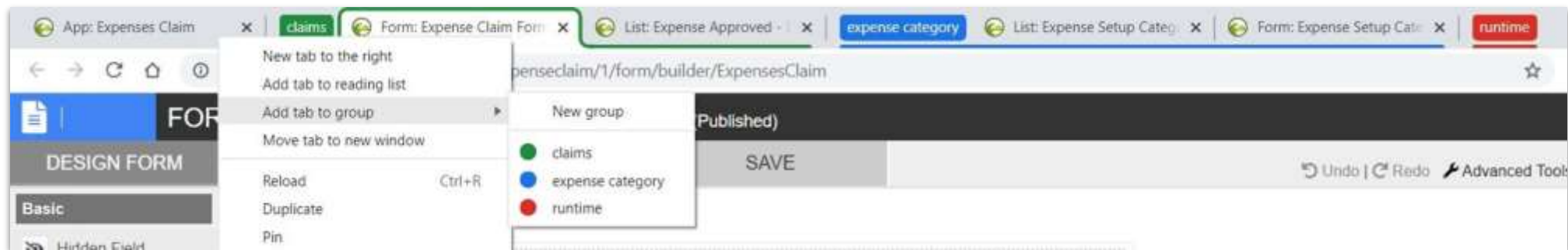
Comments

- Use this feature to comment on a form section/userview category. This can aid comprehension in a collaborative development team.
- Example: take note of the intention of a section or category/pending edits/notable info/etc.



Chrome Browser - Add Tab to Group Function

- If you use chrome browser, try use the add tab to group feature to stay organized, avoid confusion and open relevant tabs only



A horizontal bar with a black left half and an orange right half.

Chapter 2

Performance

About Performance...

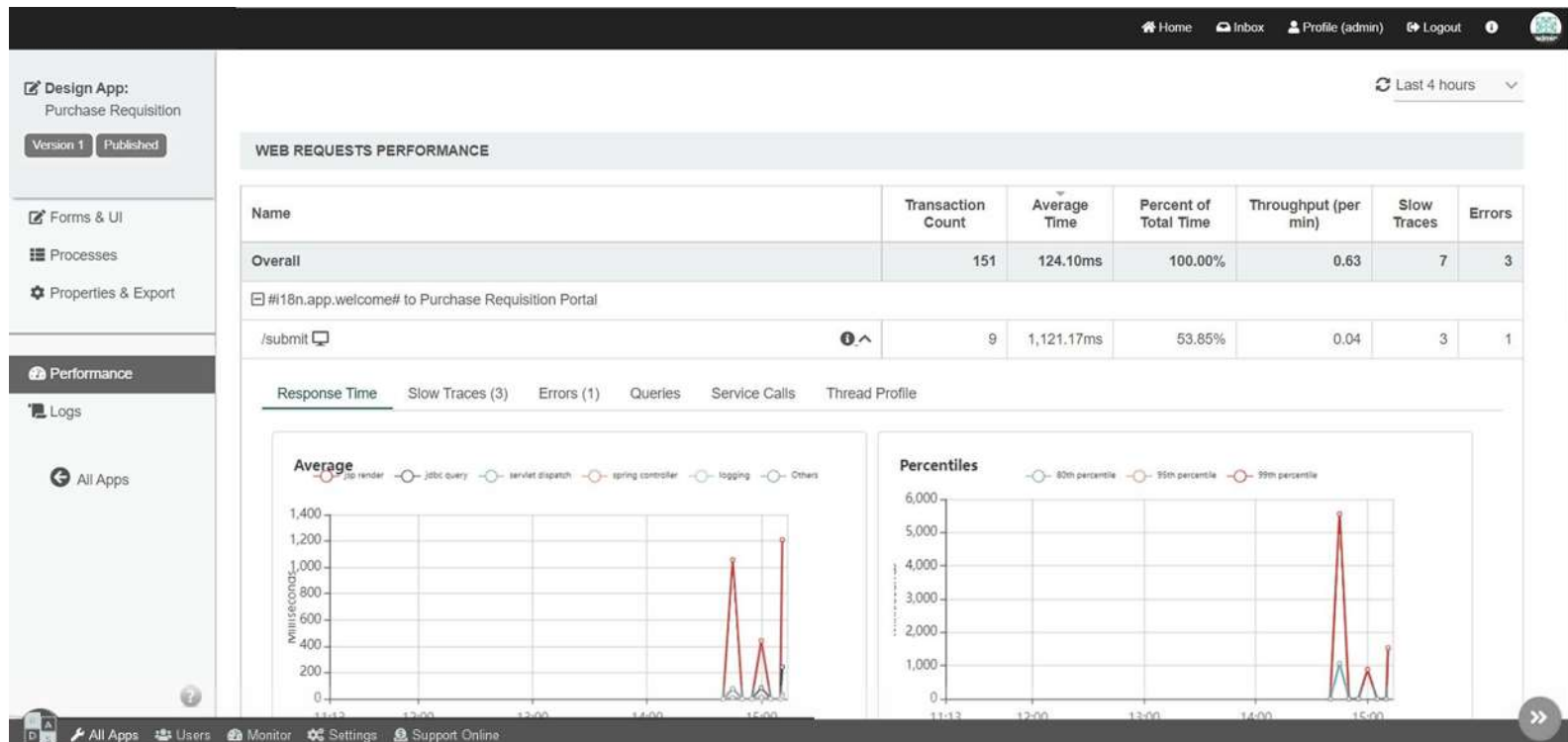
- Performance varies **wildly** from one to another depending on factors such as:-
 - Total number of users.
 - Maximum expected concurrent users.
 - Number of apps running on the platform.
 - Complexity of each of the apps.
 - Amount of data generated in each app.
 - Network infrastructure.
 - etc...

Server sizing

- Since each app differs to one another:
 - There's no one-size-fits-all
 - If its too small = poor performance
 - if its too big = waste of resources
- Run stress test / load test to measure its capability
- Check APM & Performance Analyzer (DB connection leaks) to ensure it can sustain the load and does not crash the app / database / server

Performance Monitoring

- Any resource usage (e.g. JDBC connections in Beanshell scripts or plugins, IO, etc) must be closed in a try-finally block to prevent leaks.
- Monitor using **APM** to monitor usage patterns across different apps.



App Development DEVOPS style

- Use APM to continuously monitor slow traces and errors during development
- Continuously monitor existing pages for performance improvements and regressions.

Scripting

- Scripting
 - Beanshell scripts are good for **rapid** development, but for large deployments, it may be slow.
 - Consider developing/porting to a plugin.

Integration

- Any integration (e.g. SQL queries, REST calls, etc) may be slow, so optimize them.

Database Indexing

- Form Data are stored with the prefix “**app_fd_**”.
- Columns are saved in **text/varchar** to maintain flexibility.
- Add indexing as needed, depending on use case.
- MySQL/MariaDB related: Enable slow query logging to track and improve slow queries.

Caching

- Since v6, all userview elements can be cached to reduce resource-intensive server/database operations (for non time-sensitive elements only).
- BE CAREFUL when selecting caching scope.
- Visit <https://docs.rads.purwana.net/Performance+Improvement+with+Userview+Caching> to find out more.

More...

- Visit

<https://docs.rads.purwana.net/Deployment+Best+Practices>

for more deployment best practices and tips.

Module Review

1. Application Building Best Practices
2. Performance

Stay Connected With RADS

- rads.purwana.net
- <https://github.com/radservice/rad-community>